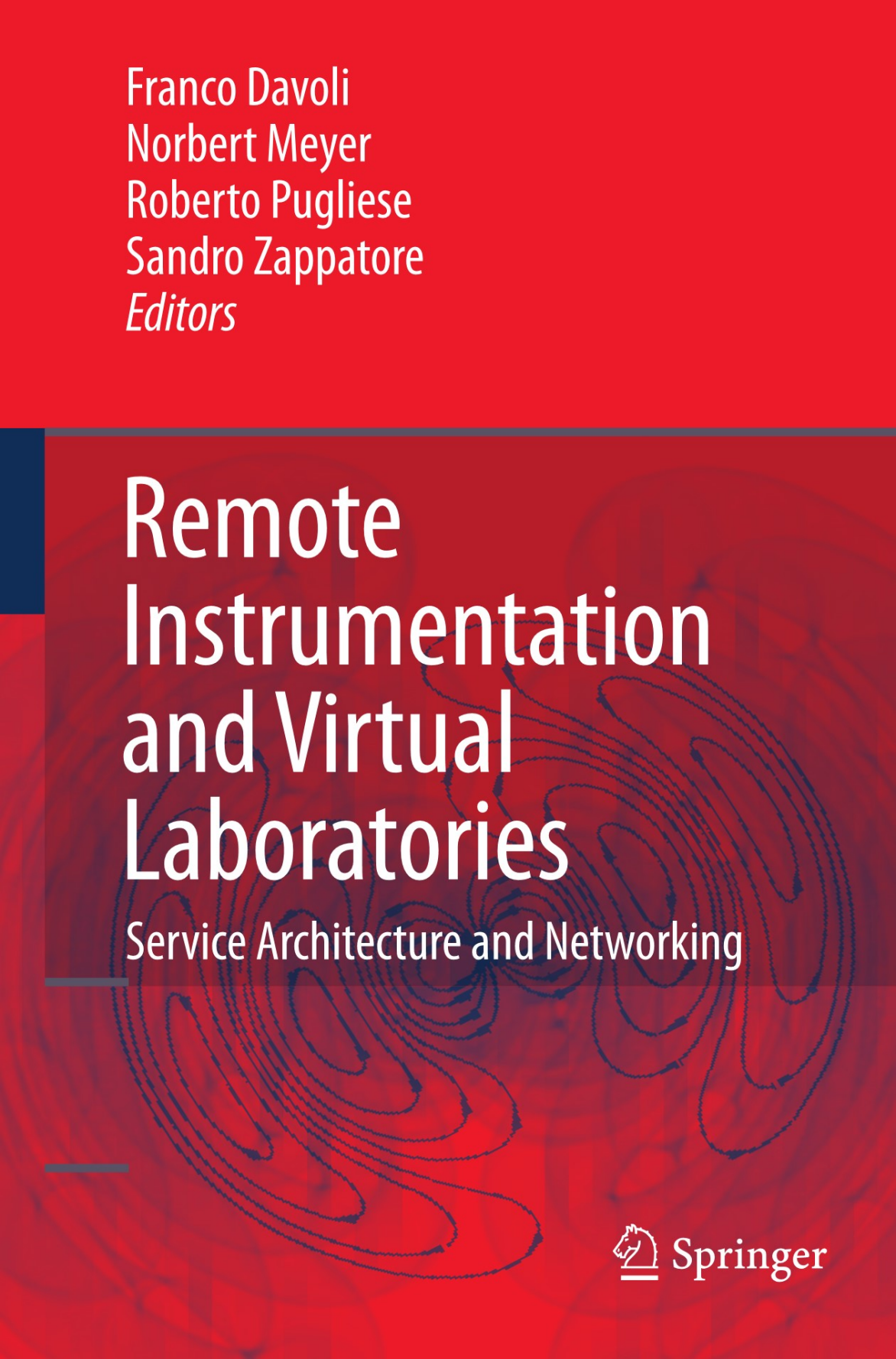
Franco Davoli
Norbert Meyer
Roberto Pugliese
Sandro Zappatore
Editors

Remote Instrumentation and Virtual Laboratories

Service Architecture and Networking



Springer

Remote Instrumentation and Virtual Laboratories

Franco Davoli · Norbert Meyer · Roberto Pugliese · Sandro Zappatore
Editors

Remote Instrumentation and Virtual Laboratories

Service Architecture and Networking

Springer

Editors
Franco Davoli
University of Genoa
DIST
Via Opera Pia, 13
16145 Genova
Italy
franco@dist.unige.it

Norbert Meyer
Poznań Supercomputing &
Networking Center
ul. Noskowskiego 10
61-704 Poznań
Poland
meyer@man.poznan.pl

Roberto Pugliese
Sincrotrone Trieste S.p.A.
Strada Statale
14 km 163.5 in Area Science Park
34012 Basovizza
Trieste
Italy
roberto.pugliese@elettra.trieste.it

Sandro Zappatore
University of Genoa
DIST
Via Opera Pia, 13
16145 Genova
Italy

ISBN 978-1-4419-5595-1 e-ISBN 978-1-4419-5597-5
DOI 10.1007/978-1-4419-5597-5
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010921156

Preface

eScience is generally recognized as a complex of activities that require highly intensive computation on huge data sets in a large-scale networked distributed environment. It embraces such diverse disciplines as particle and high-energy physics, meteorology, astronomy, astrophysics, geo-science, and biology, just to mention a few. Another essential component of such activities is the capability of real-time interaction and collaboration among many scientists working on a specific problem or data set worldwide. However, there is a third major aspect that is sometimes less emphasized in this context, which is related with the use of specific instrumentation – the pieces of equipment physically producing the data that pertain to a certain experiment. Remotely accessing instruments, configuring their parameters, setting up a "measurement chain" – possibly distributed among different laboratories, running the experiment, generating and collecting the data are all preliminary actions to any further activities regarding data analysis and interpretation. System-level science and effectively running the complex workflows that modern science requires are made possible only by integrating instruments in the computing and data treatment pipeline.

In turn, large data sets and computationally intensive operations of eScience involve the use of Grid computing paradigms and their middleware. Bringing instrumentation into this framework raises new problems and challenges. What new middleware components and functionalities are needed to expose instruments as any other resources in the eInfrastructure? Which abstractions can better represent the multiplicity of scientific instruments and laboratory equipment? How and to what extent does the distributed environment affect measurement precision? These and other issues arise that require attention and answers. Even more than with other aspects in distributed systems, dealing with remote instrumentation to set up real and usable Remote Instrumentation Services (RIS) involves the multi-disciplinary interaction among computer science, networking, and metrology.

This is the third book in this series that attempts to put together such different points of view in the field of Remote Instrumentation. The first one (*Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*, Springer, New York, 2006; ISBN 0-387-29811-8) was less focused on the integration of instruments and laboratories in the Grid and rather more concentrated on networking and measurement aspects. The second one (*Grid-Enabled Remote*

Instrumentation, Springer, New York, 2008; ISBN 978-0-387-09662-9) set forth the architectural and middleware aspects, though not neglecting the other relevant dimensions. In the present volume, we continue pursuing this point of view, with greater emphasis on the eInfrastructure in support of RIS and its features.

All books stemmed from a series of workshops on Distributed Cooperative Laboratories, which have converged under the name INGRID ("*Instrumenting*" the Grid), all held so far in Italy in different locations. The contributions in this book are extended and revised versions of presentations at INGRID 2008 (`http://www.ingrid08.cnit.it`), which took place in Lacco Ameno, Ischia, Italy, from April 9 to 11, 2008. Since the beginning, the INGRID Workshops were organized within the framework of a number of different European research projects. Among others, GRIDCC (Grid enabled Remote Instrumentation with Distributed Control and Computation), int.eu.grid (Interactive European Grid), g-Eclipse, CoreGRID, EDGeS (Enabling Desktop Grids for e-Science), EXPReS (Express Production Real-time e-VLBI Service), EGI (European Grid Initiative), DORII (Deployment of Remote Instrumentation Infrastructure), and RINGrid (Remote Instrumentation in Next-generation Grids) provided some of the results that are described
herein.

The material in the book is organized in seven parts, each containing a number of articles. Part I, Remote Instrumentation Services, presents an overview of the activities of the RISGE (Remote Instrumentation Services in Grid Environment) Research Group (`http://forge.ogf.org/sf/projects/risge-rg`) of the OGF (Open Grid Forum), followed by a description of the Instrument Element (IE) – one of the main components to provide Grid-enabled instrumentation, and of some specific use cases in high-energy physics, telecommunication measurement equipment, and genomics.

Part II, Grid Infrastructure, Services, and Applications, after a short presentation of the structure and scope of the EGI, deals with Grid-specific functionalities that are relevant also to RIS, besides other services: automatic service deployment, job scheduling, Quality of Service (QoS) provisioning, resource management, global services across multiple Grids, desktop Grids, soft real-time constraints. The part presents also two applications in electron microscopy and simulation of a biophysical phenomenon.

Part III is entirely dedicated to Interactivity Services, an important aspect to provide the operator with the capability of online controlling the instrumentation and the experiment execution. Several aspects are considered, from the coexistence of parallel and interactive jobs to interactive data visualization, among others.

Part IV touches upon some Supporting Services. These include middleware independence, context-aware service composition, synchronization, resource discovery based on Specific Service Agreements (SLAs), QoS across multiple networking domains, and the use of portals in a specific application environment.

Part V addresses some large-scale instrumentation for radio astronomy, namely e-VLBI (electronic Very Large Baseline Interferometry) and Cosmic Rays Detection, treating both computational and networking aspects.

In Part VI, Metrology Issues are considered in some detail. They include wireless networking and synchronization in distributed measurement systems, as well as the description of some educational applications in remote instrumentation.

Finally, Part VII is devoted to Wireless Sensor Networks for Measurement. Wireless Sensor Networks (WSNs) are a fundamental component of Remote Instrumentation Services and constitute the basis for data acquisition in many distributed applications. The contributions in this part cover aspects in signal processing, data aggregation, energy saving, and integration of WSNs in the Service-Oriented Architecture.

Franco Davoli
Norbert Meyer
Roberto Pugliese
Sandro Zappatore

Acknowledgments

We gratefully acknowledge the support of the many persons and institutions that contributed to the success of the third Workshop on Distributed Cooperative Laboratories (INGRID 2008) and to this book. The European Commission, through the Directorate General for Information Society and Media (DG-INFSO), and in particular the Research Infrastructures Unit, has funded most of the projects that supported the research lines covered by this book. Our thanks and appreciation go to the keynote speakers, Prof. Ian Foster, Argonne National Laboratories and the University of Chicago, USA, and Dr. Maria Ramalho-Natario, DG-INFSO. Thanks are due also to the session organizers (*Jesus Marco de Lucas, IFCA-CSIC, Spain; T. Charles Yun, JIVE, The Netherlands; Luigino Benetazzo, University of Padova, Italy; Marcin Płóciennik, PSNC, Poland; Dieter Kranzlmueller, LMU Munich, Germany; Per Öster, CSC, Finland; Tatsuya Suda, UCI, USA; Marcin Lawenda, PSNC, Poland*), Program Committee members, authors and presenters, as well as to Clotilde Fertini-Canepa for the logistic organization, to Stefano Vignola and Carlo Caligaris for carefully checking and formatting the manuscript, and to the Springer editorial staff. Last but not least, we are grateful to Agilent Technologies, Selex Communications, and SIR S.r.l. for their sponsorship of the event.

Contents

Contributors

D. Adami CNIT Research Unit, Department of Information Engineering, University of Pisa, Italy, d.adami@iet.unipi.it

G. Alfano Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129, Torino, Italy, giuseppa.alfano@polito.it

O. Andrisano WiLAB, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy, o.andrisano@ieee.org

J. Astalos Institute of informatics, Slovak academy of sciences, Bratislava, Slovakia, astalos.ui@savba.sk

K. Baba Cybermedia Center, Osaka University, Japan; Graduate School of Information Science and Technology, Osaka University, Japan; National Institute of Information and Communication Technology, Japan, baba@cmc.osaka-u.ac.jp

A. Bagnasco DIBE – University of Genoa, 16145 Genova, Italy, andrea.bagnasco@unige.it

Z. Balaton MTA SZTAKI, Computer and Automation Research Institute, Hungarian Academy of Sciences, H-1528 Budapest, Hungary, balaton@sztaki.hu

M. Baumgartner GUP Linz, Joh. Kepler University, Linz, Austria, mb@gup.jku.at

L. Benetazzo Department of Information Engineering, University of Padova, 35131 Padova, Italy, luigino.benetazzo@unipd.it

L. Berruti CNIT – University of Genoa Research Unit, 16145 Genova, Italy, luca.berruti@cnit.it

M. Bertocco Department of Information Engineering, University of Padova, 35131 Padova, Italy, matteo.bertocco@unipd.it

G. Borges Laboratòrio de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal, goncalo@lip.pt

P. Buschiazzo DIBE – University of Genoa, 16145 Genova, Italy, paolo.buschiazzo@unige.it

C. Callegari Department of Information Engineering, University of Pisa, Italy, christian.callegari@iet.unipi.it

I. Campos Instituto de Física de Cantabria, IFCA, CSIC-UC, Santander, Spain, iscampos@ifca.unican.es

L. Carlino DIBE – University of Genoa, 16145 Genova, Italy, luca.carlino@unige.it

F. Castejón Instituto de Biocomputación y Física de Sistemas Complejos. Universidad de Zaragoza, 50009 Zaragoza, Spain; Laboratorio Nacional de Fusión – Asociación Euratom/Ciemat, 28040-Madrid, Spain, francisco.castejon@ciemat.es

C.F. Chiasserini Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129, Torino, Italy, chiasserini@polito.it

M. Ciglan Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia, marek.ciglan@savba.sk

A. Clematis IMATI-CNR, Via De Marini 6, 16149 Genova, Italy, clematis@ge.imati.cnr.it

B. Coghlan Trinity College Dublin, Dublin, Ireland, brian.coghlan@tcd.ie

A. Conti WiLAB, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy; ENDIF, University of Ferrara, Italy, a.conti@ieee.org

A. Corana IEIIT-CNR, Via De Marini 6, 16149 Genova, Italy, corana@ieiit.cnr.it

D. Cordero Centro de Supercomputación de Galicia, Santiago de Compostela, Spain, dcordero@cesga.es

D. Cretoni Thales Alenia Space Italia S.p.A., Roma, Italy, daniele.cretoni@thalesaleniaspace.com

A. Curri Sincrotrone Trieste S.C.p.A., Strada Statale per Basovizza 14, 34012 Trieste, Italy, alessio.curri@elettra.trieste.it

D. D'Agostino IMATI-CNR, Via De Marini 6, 16149 Genova, Italy, daniele.dagostino@ge.imati.cnr.it

P. Daponte Department of Engineering, University of Sannio, Corso Garibaldi 107, 82100 Benevento, Italy, daponte@unisannio.it

D. Dardari WiLAB, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy, ddardari@ieee.org

S. Date Graduate School of Information Science and Technology, Osaka University, Japan, date@ime.cmc.osaka-u.ac.jp

K. Daumiller Forschungszentrum Karlsruhe, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany, kai.daumiller@ik1.fzk.de

M. David Laboratòrio de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal, david@lip.pt

F. Davoli CNIT – University of Genoa Research Unit, 16145 Genova, Italy, franco.davoli@cnit.it

J. Marco de Lucas Instituto de Fisica de Cantabria, IFCA, CSIC-UC Santander, Spain, marco@ifca.unican.es

A. Del Linz Sincrotrone Trieste S.C.p.A., Strada Statale per Basovizza 14, 34012 Trieste, Italy, andrea.dellinz@elettra.trieste.it

T. Delaitre Centre of Parallel Computing, University of Westminster, gemlca-discuss@cpc.wmin.ac.uk

N. Dias Laboratòrio de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal, ndias@lip.pt

K. Dichev High-Performance Computing Center Stuttgart, Stuttgart, Germany, dichev@hlrs.de

D. Drótos Department of Automation, University of Miskolc, H-3515 Miskolc, Egyetemv'aros, Hungary, drdani@mazsola.iit.uni-miskolc.hu

T. Eickermann Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany, t.eickermann@fz-juelich.de

Z. Farkas MTA SZTAKI Computer and Automation Research Institute, 1518 Budapest, zfarkas@sztaki.hu

E. Fernández Departament d'Arquitectura de Computadors i Sistemes Operatius, Universitat Autónoma de Barcelona, Barcelona, Spain, enol@aomail.uab.es

C. Fernández Centro de Supercomputación de Galicia, Santiago de Compostela, Spain, carlosf@cesga.es

G. Ferrari Wireless Ad-hoc and Sensor Networks (WASN) Laboratory, Department of Information Engineering, University of Parma, I-43100, Parma, Italy, gianluigi.ferrari@unipr.it

A. Ferrero Dipartimento di Elettrotecnica, Politecnico di Milano, Milano, Italy, alessandro.ferrero@ieee.org

M. Figlerowicz Institute of Bioorganic Chemistry PAS, Noskowskiego 12/14, 61-704 Poznań, Poland, marekf@ibch.poznan.pl

E. Floros GRNET, Athens, Greece, efloros@grnet.gr

F. Freitag Computer Architecture Department, Polytechnic University of Catalonia, Jordi Girona, 08034 Barcelona, Spain, felix@ac.upc.edu

W. Frings Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany, w.frings@fz-juelich.de

K. Fujii School of Information and Computer Science, University of California, Irvine, California, USA, kfujii@ics.uci.edu

G. Gamba Department of Information Engineering, University of Padova, 35131 Padova, Italy, giovanni.gamba@unipd.it

L. García-Tarrés Centro de Supercomputación de Galicia, Santiago de Compostela, Spain, lino.garcia@icmab.es

H. Gemmeke Forschungszentrum Karlsruhe, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany, gemmeke@hpe.fzk.de

V. Gianuzzi DISI – Università degli Studi di Genova, Via Dodecaneso 35, 16146 Genova, Italy, gianuzzi@disi.unige.it

P. Gibbon Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany, p.gibbon@fz-juelich.de

S. Giordano Department of Information Engineering, University of Pisa, Italy, s.giordano@iet.unipi.it

J. Gomes Laboratòrio de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal, jorge@lip.pt

M. Gotta Thales Alenia Space Italia S.p.A., Roma, Italy, monica.gotta@thalesaleniaspace.com

F. Gottifredi Thales Alenia Space Italia S.p.A., Roma, Italy, franco.gottifredi@thalesaleniaspace.com

M. Grammatikou Network Management & Optimal Design Laboratory (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece, mary@netmode.ntua.gr

D. Grimaldi DEIS, University of Calabria, 87036 Rende (CS), Italy, grimaldi@deis.unical.it

S. Habbinga Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany, s.habbinga@fz-juelich.de

A. Hammad Steinbuch Centre for Computing, Forschungszentrum Karlsruhe, Karlsruhe, Germany, ahmad.hammad@kit.edu

L. Handschuh Institute of Bioorganic Chemistry PAS, Noskowskiego 12/14, 61-704 Poznań, Poland; Departament of Hematology, Poznań University of Medical Sciences, Szamarzewskiego 84, 60-569 Poznań, Poland, luizahan@ibch.poznan.pl

M. Hardt Steinbuch Centre for Computing, Forschungszentrum Karlsruhe, Karlsruhe, Germany, marcus.hardt@iwr.fzk.de

P. Heinzlreiter GUP Linz, Joh. Kepler University, Linz, Austria, heinzlreiter@gup.jku.at

E. Heymann Departament d'Arquitectura de Computadors i Sistemes Operatius, Universitat Autònoma de Barcelona, Barcelona, Spain, e.heymann@cc.uab.es

T. Jejkal Forschungszentrum Karlsruhe, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany, thomas.jejkal@ipe.fzk.de

P. Kacsuk MTA SZTAKI, Computer and Automation Research Institute, Hungarian Academy of Sciences, H-1528 Budapest, Hungary, kacsuk@sztaki.hu

S. Kafetzoglou Network Management & Optimal Design Lab. (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece, skafetzo@netmode.ntua.gr

G. Kecskemeti Laboratory of Parallel and Distributed Systems, MTA-SZTAKI, kecskemeti@sztaki.hu

S. Kenny Trinity College Dublin, Dublin, Ireland, stuart.kenny@cs.tcd.ie

A. Kertész MTA SZTAKI Computer and Automation Research Institute, H-1518 Budapest, Hungary, keratt@inf.uszeged.hu

T. Kiss Centre for Parallel Computing, School of Informatics, University of Westminster, UK, kisst@wmin.ac.uk

M. Knauer Innoopract GmbH, Karlsruhe, Germany, mknauer@innoopract.com

D. Kollia Network Management & Optimal Design Lab. (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece, dimkollia@netmode.ntua.gr

A. Kopmann Forschungszentrum Karlsruhe, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany, kopmann@ipe.fzk.de

H. Kornmayer NEC Laboratories Europe, St. Augustin, Germany, kornmayer@it.neclab.eu

D. Kranzlmüller Ludwig-Maximilians-Universitaet (LMU) Muenchen & Leibniz Supercomputing Centre (LRZ), Germany, kranzlmueller@ifi.lmu.de

N.G.H. Kruithof Joint Institute for VLBI in Europe (JIVE), Postbus 2, 7990 AA Dwingeloo, The Netherlands, nico@nghk.nl

P. Lason Academic Computer Centre CYFRONET AGH, Krakow, Poland, p.lason@cyfronet.pl

M. Lawenda Poznań Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland, lawenda@man.poznan.pl

T. Lippert Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany, th.lippert@fz-juelich.de

C. Loomis Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud 11, Orsay, France, loomis@lal.in2p3.fr

J. López Centro de Supercomputación de Galicia, Santiago de Compostela, Spain, jlopez@cesga.es

A. López Instituto de Física de Cantabria, IFCA, CSIC-UC, Santander, Spain, aloga@ifca.unican.es

D. Lopez-Bruna Instituto de Biocomputación y Física de Sistemas Complejos. Universidad de Zaragoza 50009 Zaragoza, Spain; Laboratorio Nacional de Fusión – Asociación Euratom/Ciemat 28040-Madrid, Spain, daniel.lopezbrunaciemat.es

V. Maglaris Network Management & Optimal Design Laboratory (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece, maglaris@netmode.ntua.gr

M. Marastoni Pride s.p.a., I-20151, Milano, Italy, m.marastoni@pride.it

D.C.P.M. Marchal University of Amsterdam (UvA), Kruislaan 403, 1098 SJ Amsterdam, The Netherlands, dmarchal@science.uva.nl

M. Marchenko Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia, mam@osmf.sscc.ru

J. Marco Instituto de Física de Cantabria, IFCA, CSIC-UC, Santander, Spain, marco@ifca.unican.es

R. Marco Instituto de Física de Cantabria, IFCA, CSIC-UC, Santander, Spain, rmarco@ifca.unican.es

C. Marinos Network Management & Optimal Design Laboratory (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece, cmarinos@netmode.ntua.gr

A.Cs. Marosi MTA SZTAKI Computer and Automation Research Institute, 1518 Budapest, atisu@sztaki.hu

J.P. Martins Laboratòrio de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal, martinsj@lip.pt

H.-J. Mathes Forschungszentrum Karlsruhe, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany, hermann-josef.mathes@ik.fzk.de

P. Medagliani Wireless Ad-hoc and Sensor Networks (WASN) Laboratory, Department of Information Engineering, University of Parma, I-43100, Parma, Italy, paolo.medagliani@unipr.it

A. Merlo DISI – Università degli Studi di Genova, Via Dodecaneso 35, 16146 Genova, Italy, merlo@disi.unige.it

N. Meyer Poznań Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland, meyer@man.poznan.pl

F. Mighela Telemicroscopy Laboratory, Sardegna Ricerche, Pula (CA), Italy, francesca.mighela@diee.unica.it

G. Molera Metsähovi Radio Observatory, TKK, FIN-02540 Kylmälä, Finland, gofrito@kurp.hut.fi

M. Montecelo Laboratòrio de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal, mafm@lip.pt

M. Nakagawa Cybermedia Center, Osaka Univeristy, Japan; Graduate School of Information Science and Technology, Osaka University, Japan; National Institute of Information and Communication Technology, Tokyo, Japan, m-nakagawa@ist.osaka-u.ac.jp

L. Navarro Computer Architecture Department, Polytechnic University of Catalonia, Jordi Girona, 08034 Barcelona, Spain, leandro@ac.upc.edu

K. Nawrocki Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Warsaw, Poland, nawrocki@fuw.edu.pl

A. Nordio Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129, Torino, Italy, alessandro.nordio@polito.it

M. Noro National Institute of Information and Communication Technology, Tokyo, Japan, noro@ais.cmc.osaka-u.ac.jp

K. Nozaki Cybermedia Center, Osaka Univeristy, Japan; Graduate School of Information Science and Technology, Osaka University, Japan, nozaki@cmc.osaka-u.ac.jp

P. Orviz Instituto de Física de Cantabria, IFCA, CSIC-UC, Santander, Spain, pablo.orviz@gmail.com

P. Öster CSC – IT Center for Science Ltd., FI-02101 Espoo, Finland, per.oster@csc.fi

R. Ottoboni Dipartimento di Elettrotecnica, Politecnico di Milano, Milano, Italy, roberto.ottoboni@polimi.it

A. Padee Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Warsaw, Poland, apadee@ire.pw.edu.pl

M. Pagano Department of Information Engineering, University of Pisa, Italy, m.pagano@iet.unipi.it

S. Papavassiliou Network Management & Optimal Design Laboratory (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece, papavass@mail.ntua.gr

C. Perra 2DIEE, University of Cagliari, Cagliari, Italy, cperra@diee.unica.it

M. Płóciennik Poznań Supercomputing and Networking Center, Poznań, Poland, marcinp@man.poznan.pl

M. Pospieszny Poznań Supercomputing and Networking Center, Poznań, Poland

V. Pouli Network Management & Optimal Design Laboratory (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece, vpouli@netmode.ntua.gr

M. Prica Sincrotrone Trieste S.C.p.A., Strada Statale per Basovizza 14, 34012 Trieste, Italy, milan.prica@elettra.trieste.it

T. Prokosch GUP, Johannes Kepler University, Altenbergerstr. 69, A-4040 Linz, Austria, thomas.prokosch@gup.uni-linz.ac.at

R. Pugliese Sincrotrone Trieste S.C.p.A., Strada Statale per Basovizza 14, 34012 Trieste, Italy, roberto.pugliese@elettra.trieste.it

A. Quarati IMATI-CNR, Via De Marini 6, 16149 Genova, Italy, quarati@ge.imati.cnr.it

S. Rapuano Department of Engineering, University of Sannio, Corso Garibaldi 107, 82100 Benevento, Italy, rapuano@unisannio.it

J.M. Reynolds Instituto de Biocomputación y Física de Sistemas Complejos. Universidad de Zaragoza 50009 Zaragoza, Spain; Departamento de Fisica Teórica, Universidad de Zaragoza 50009 Zaragoza, Spain, jmrb2002@gmail.com

M. Riedel Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany, m.riedel@fz-juelich.de

D. Rodriguez Instituto de Física de Cantabria, IFCA, CSIC-UC, Santander, Spain, drodrig@ifca.unican.es

F. Rodríguez-Haro Polytechnic University of Catalonia, Computer Architecture Department, Jordi Girona, 08034 Barcelona, Spain, frodrigu@ac.upc.edu

H. Rosmanith Interactive European Grid Project, GUP Linz, Austria, hr@gup.unilinz.ac.at

N.V. Ruiter Institute for Data Processing and Electronics (IPE), Forschungszentrum Karlsruhe, Postfach 3640, 76021 Karlsruhe Germany, nicole.ruiter@ipe.fzk.de

A.M. Scapolla DIBE – University of Genoa, 16145 Genova, Italy, scapolla@unige.it

M.A. Senar Departament d'Arquitectura de Computadors i Sistemes Operatius, Universitat Autònoma de Barcelona, Barcelona, Spain, miquelangel.senar@uab.es

S. Shimojo Cybermedia Center, Osaka Univeristy, Japan; Graduate School of Information Science and Technology, Osaka University, Japan; National

Institute of Information and Communication Technology, Tokyo, Japan, shimojo@cmc.osakau.ac.jp

B. Silva Laboratòrio de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal

L. Skital Academic Computer Centre CYFRONET AGH, Krakow, Poland, l.skital@cyfronet.pl

R. Soloperto WiLAB, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy, raffaele.soloperto@unibo.it

A. Sona Department of Information Engineering, University of Padova, 35131 Padova, Italy, alessandro.sona@unipd.it

P. Stępniak Institute of Bioorganic Chemistry PAS, Noskowskiego 12/14, 61-704 Poznań, Poland

R. Stotzka Forschungszentrum Karlsruhe, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany, rainer.stotzka@kit.edu

A. Streit Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany, a.streit@fz-juelich.de

M. Stroiński Poznań Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland, stroins@man.poznan.pl

M. Stümpert Institute for Scientific Computing, Forschungszentrum Karlsruhe, Germany, mathias.stuempert@iwr.fzk.de

T. Suda School of Information and Computer Science, University of California, Irvine, California, USA, suda@ics.uci.edu

M. Sutter Forschungszentrum Karlsruhe, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany, michael.sutter@ipe.fzk.de

A. Tarancón Instituto de Biocomputación y Física de Sistemas Complejos. Universidad de Zaragoza 50009 Zaragoza, Spain; Departamento de Fisica Teórica, Universidad de Zaragoza 50009 Zaragoza, Spain, tarancon@unizar.es

G. Terstyanszky Centre of Parallel Computing, University of Westminster, UK, gemlca-discuss@cpc.wmin.ac.uk

M. Uunila Metsähovi Radio Observatory, TKK, FIN-02540 Kylmälä, Finland, minttu@kurp.hut.fi

R. Valles Instituto de Biocomputación y Física de Sistemas Complejos, Zaragoza, Spain, rvalles@bifi.es

E. Varriale Thales Alenia Space Italia S.p.A., Roma, Italy, enrico.varriale@thalesaleniaspace.com

C. Veiga Centro de Supercomputación de Galicia, Santiago de Compostela, Spain

J.L. Velasco Instituto de Biocomputación y Física de Sistemas Complejos. Universidad de Zaragoza 50009 Zaragoza, Spain; Departamento de Fisica Teórica, Universidad de Zaragoza 50009 Zaragoza, Spain, velasco@unizar.es

S. Vignola CNIT – University of Genoa Research Unit, 16145 Genova, Italy, stefano.vignola@cnit.it

J. Volkert Johannes Kepler University Linz, GUP Linz, Austria, volkert@gup.uni-linz.ac.at

J. Wagner Metsähovi Radio Observatory, TKK, FIN-02540 Kylmälä, Finland, jwagner@kurp.hut.fi

J. Węglarz Poznań Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland, jan.Weglarz@cs.put.poznan.pl

W. Wislicki Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Warsaw, Poland, wislicki@fuw.edu.pl

F. Wolf Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany; Department of Computer Science, RWTH Aachen University, 52056 Aachen, Germany, f.wolf@fzjuelich.de

M. Zapf Institute for Data Processing and Electronics (IPE), Forschungszentrum Karlsruhe, Postfach 3640, 76021 Karlsruhe Germany, michael.zapf@ipe.fzk.de

S. Zappatore CNIT – University of Genoa Research Unit, 16145 Genova, Italy, sandro.zappatore@unige.it

Part I
Remote Instrumentation Services

Chapter 1
Open Grid Forum Research Group: Remote Instrumentation Services in Grid Environment – Overview of Activities

M. Płóciennik and R. Pugliese

Abstract This chapter presents the activities of the Open Grid Forum Remote Instrumentation Services in Grid Environment Research Group. The main goal of this research group is to bring together different approaches in defining remote access to scientific instruments and analyse use cases that integrate scientific instruments with Grids. This group brings together people that are involved in the field of remote instrumentation and Grid, in order to document experiences, present best practices, and work on informational documents that would describe different use cases and a "model use case". The group is currently in the phase of collecting use cases and identifying a "model use case". First, results and an example use case are presented in order to illustrate the future directions of the research.

Keywords Remote instrumentation · Grid · Open grid forum · Instruments · Remote access · Interactivity · Visualization · Model use case · Experimental facilities · Accelerators · Synchrotron · Free electron lasers

1 Introduction

The main goal of this chapter is to introduce the work of the Open Grid Forum Remote Instrumentation Services in Grid Environment Research Group (OGF RISGE-RG). This worldwide research group brings together various existing approaches in defining remote access interfaces to instruments and unique laboratory equipment. One of the foreseen outcomes of the group work is the document describing use cases that present requirements for integrating scientific instruments with Grids.

In this chapter we present goals and scope of the group, past and current activities, research projects involved in the activities, and some future actions. We also present first results of this on-going activity – an example use case, as well as some considerations on the "model use case".

M. Płóciennik (✉)
Poznan Supercomputing and Networking Center, Poznań, Poland
e-mail: marcinp@man.poznan.pl

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_1,

2 Main Goals OF OGF RISGE-RG

The main goals of the group as well as its scope have been defined within the OGF charter [1]. "RISGE-RG explores issues related to the exploitation of Grid technologies for conducting and monitoring measurement tasks and experiments on complex remote scientific equipment. The main purpose of this research group is to bring together various existing approaches in defining remote access interfaces to sophisticated laboratory equipment and to come up with use cases that can dictate the requirements for integrating scientific instruments with the Grid. As such, it mostly concerns the steering and monitoring of instrument resources, although more typical problems such as user access and authorization are also in scope. The advances of grid technologies in areas such as interactivity, visualization, and quality of service provisioning play an important role in accessing remote devices; therefore, the description of suitable service-level terms is highly relevant".

Work of this group is focussing on providing an overview of existing solutions as well as on indicating some requirements and future directions of e-infrastructure functionality development to facilitate access to instruments.

This group brings together people that are involved in the field of remote instrumentation and grid, in order to document experiences, present best practices, and work on informational documents that would describe different use cases and would define a "model use case". Such document could be later used by other Working Groups (WG) that are taking care of some aspects of use cases. Depending on the results, the group will take into account the possibility of establishing a new WG to standardize the relevant capabilities.

3 Past and Current Activities

The group was established during the OGF20 meeting in Manchester, May 2007, where it had the first Birds of a Feather (BoF) session. The following European projects were initially involved in creating the group: RINGrid, int.eu.grid, GRIDCC, g-Eclipse, DEGREE, Edutain@Grid, BalticGrid, and VLab. The session attracted almost 50 participants representing different geographical and research areas. The group defined the scope and goals, and agreed to write a charter. During the official process of the group creation other projects from the field of Remote Instrumentation and Grids joined the group. The first official group session, after OGF approval, took place during OGF21 in Seattle, October 2007. The group defined there a roadmap and agreed on formal aspects like templates, logo, and the mailing list.

During next sessions in 2008, OGF 22 in Cambridge, MA (USA), and INGRID 2008, Ischia (Italy), as well as using the mailing list, the group discussed the structure of the OGSA (open grid services architecture)-style use case template. A

number of best practices were presented, including “Service Oriented Architectures for Remote Instrumentation”, “RINGrid: Evaluation of Remote Instrumentation Infrastructures”, “Grid-enabled Remote Instrumentation with Distributed Control and Computation”, “Ocean Observatory Initiative and Grid Computing”, “The EGEE Infrastructure and Remote Instruments”, “Interactive controlling and monitoring of applications in grid environment”, “Interactive grid framework standardization in Eclipse environment”, and “Visualization in grid environment”. Based on the provided template essential information coming from projects involved in the group was collected and published on the project website.

The following research projects are represented in the group’s mailing lists:

- GRIDCC – Grid-enabled Remote Instrumentation with Distributed Control and Computation (`http://www.gridcc.org`),
- RINGrid – Remote Instrumentation in Next-Generation Grids (`http://www.ringrid.eu`),
- int.eu.grid – Interactive European Grid line (`http://www.interactive-grid.eu`),
- g-Eclipse (`http://www.geclipse.org`),
- AUGERACCESS – Integrating Auger Observatory and European Research Institutions into a world-wide Grid (`http://www.augeraccess.net`),
- VLab – Virtual Laboratory (`http://vlab.psnc.pl`),
- Edutain@Grid (`http://www.edutaingrid.eu`),
- Integrated e-Infrastructure for Facilities (`http://www.e-science.stfc.ac.uk/facilities/`),
- An Open Network for Robotic Telescopes,
- NORIA – Network for Ocean Research, Interaction, and Application,
- OMF – Observatory Middleware Framework,
- SENSORS: Ocean Observing Network Infrastructure (`http://www.mbari.org/rd/sensors/sensors.htm`),
- DORII – Deployment of Remote Instrumentation Infrastructure (`http://www.dorii.eu`),
- CIMA – The Common Instrument Middleware Architecture (`http://www.instrumentmiddleware.org/`),
- BalticGrid (`http://www.balticgrid.eu`),
- EGEE – Enabling Grids for E-science (`http://www.eu-egee.org/`),
- MMRA – Minnesota Microprobe Remote Access (`http://probelab.geo.umn.edu/remote.html`),
- Looking (`http://lookingtosea.ucsd.edu`),
- CYCLOPS – Cyber-Infrastructure for Civil protection Operative Procedures (`http://www.cyclops-project.eu`),
- DEGREE – Dissemination and Exploitation of Grids in Earth Science (`http://www.eu-degree.eu`),

- NICTA Open SensorWeb Architecture (`http://nicta.com.au/research/projects/nicta_open_sensorweb_architecture`),
- ARCHER (`http://archer.edu.au`)

The list includes representatives of remote instrumentation-related user communities (like DORII, AUGERACCESS, NORIA), remote instrumentation-focused projects (like GRIDCC, RINGrid, VLab, observatory middleware framework, CIMA), grid middleware projects (like g-Eclipse), and grid infrastructure projects (like int.eu.grid, BalticGrid, EGEE). These projects come from such geographical locations as Europe, the USA, and Australia. The research group is continuously inviting other interested projects from the field of remote instrumentation and grids and is open for any kind of collaboration.

The group is also involved in horizontal OGF discussions like the Workshop on Access Paradigms, where the use case coming from the RISGE-RG has been presented and discussed. Representatives of the RISGE-RG follow the work of other groups and communities within OGF, like the Grid Visualization and Steering community.

The group has started the process of collecting use cases from members and other groups of interest in order to build a common use case, i.e. the least common denominator or a superset of all use cases. This is going to be described in a common use case OGSA-style document, which will allow identifying requirements and necessary capabilities. This could be also preceded by establishing a working group to standardize the relevant capabilities.

4 Example Use Case – Remote Operations of Experimental Facilities

As an example of such use case we present the “remote operations of experimental facilities”. This refers to experimental research done in the Synchrotron Radiation Facility ELETTRA in Trieste.

Remote operations of experimental facilities like accelerators, synchrotrons, and free electron lasers, but also experimental stations, involve planning of the operations, maintenance and troubleshooting, repair of delicate equipment, understanding and pushing performance limitations, performing commissioning and setups, and various routine operations. All these activities are based on large amounts of information, normally accessible only at the site where the facility is located.

In this scenario operators of the experimental setup and remote experts, through sophisticated software, called virtual control room, can collaboratively operate the experimental setup both in a routine and in a troubleshooting scenario.

All the high-level software and workflows are suitable to be implemented using Grid technologies. One example is the implementation of feedback workflows, where a high level of integration of instruments and traditional Grid resources is required. In the feedback, parameters of the setup are sent via Grid to a remote

computing centre, where the feedback correction code is executed and the corrections are fed back to the equipment thus improving some working parameters of the controlled devices.

4.1 Customers

The potential customers are operators of the experimental facility and the remote experts, who are scientists and experimental scientists working at these research laboratories. Actors involved in this use case are software developers, scientists (the experts), and operators of the experimental equipment. The developers can perform all the activities of the scientists and also deploy software and define the collective behaviour (workflows and scripts). The scientist can perform all the activities of the operators, plus control and monitoring of the equipment. The operators can define the operations' parameters, run the workflows involved in the operations, and monitor the evolution of the workflows. The operators can access and visualize historical data, for example, when they need to do post-mortem analysis. In specific situations (for example for troubleshooting) they need to collaborate with experts to understand where the problem is and how to solve it. They can also run simulations and then compare the simulated results with historical data and possibly understand how to improve the performance of the equipment.

4.2 Scenario

The resources involved in the remote operations are instruments, sensors, and detectors interfaced with the equipment. This is represented in the scenario by instrument elements (IE). Data acquired need to be stored in storage resources (SE) and processed by computing resources (CE). The user interacts with the involved resources and collaborates with the other users via user interfaces – in this case, web portals called Virtual Control Rooms. The following diagram describes how these components interact during operations workflows:

1. User controls actuators
2. Gyroscopes acquire data, which is then stored
3. User starts simulation
4. The simulated data is stored
5. Acquired and simulated data is compared
6. Results and stored data are visualized by the end users

The whole documentation of this specific use case [2] includes also presentation of the following: involved resources, use case simulations analysis, functional and non-functional requirements (like security considerations), and other related work.

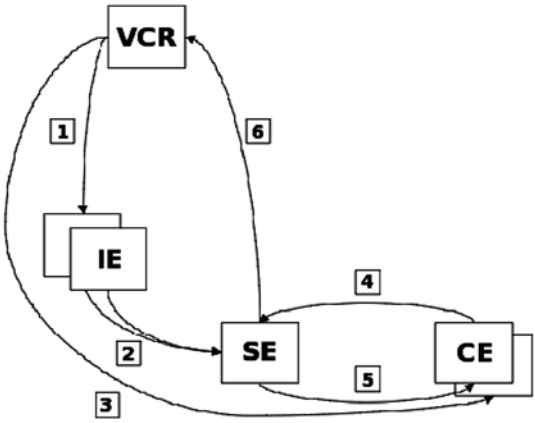


Fig. 1.1 Interaction among components during operations workflows

5 Model Use Case

As already mentioned one of the main goals of the RISGE-RG is to define a model workflow, common denominator of different collected use cases, and describe it in a common use case OGSA-style document.

The work on defining a model use case has already started within the EU RINGrid project [3]. One of the results of the RINGrid project was the "model workflow" that represents various experiments. They have presented a least common denominator of collected (by the project) use cases.

The model workflow defined by the RINGrid project consists of the following steps:

1. Schedule the experiment. This implies advance reservation of the instruments.
2. Enable the experiment (preparatory actions). It concerns actions like retrieving and preparing the input for the experiment, switching on the instrument to perform infrastructure checks, and to configure this infrastructure appropriately in order to accept the produced data. In the presented example use case this refers to step 1.
3. Define access policies. This step allows joining the experiment, either on-site or through some video-conferencing facility transmitting still/moving images. In some cases, it refers to granting permission to specific people in order to control the experiment.
4. Provide the input. The input stands for real substance, or digital information. In the presented example use case this refers to step 2.
5. Calibrate the instrument. It means the adjustment of instrument properties to minimize the expected error margin.
6. Execute the experiment. This essential activity allows controlling the instrument to perform measurements and monitor the relevant parameters. Usually during

the execution the outputs are being placed into the pre-defined data storages. In the presented example use case this refers to step 3 – starting the simulations, and to step 4 – storing the simulated data.
7. Evaluate experimental output. This step can concern actions like visualization, scientific calculations, further simulations and analysis of the results, macro- and microscopic examination of samples. In the presented example use case this refers to step 5 when the simulated and acquired data are being compared and to step 6 that means visualization of stored data.
8. If needed, loop to step 3. This loop is required in case of multi-run experiments. It is also useful when performing complex actions like calibrations or in case of experiment failure.
9. Clean-up activities. This step can include actions like switching off instruments, maintenance operations, deletion of input and output. In some cases other activities are needed, which allow the next person to perform an experiment.

There are two compulsory steps in the workflow – execution of an experimental operation (measurement, observation, etc.) and the post-processing of each such operation. Some of the steps in the workflow are compulsory, some other (that could be irrelevant to the specific scenario) are optional.

The whole model use case in details as well as the requirements are described in the RINGrid deliverable [4].

Work of RISGE-RG is referring to this model use case description as a very good base for defining a more general workflow that will take into account also other different use cases coming from the group. The first analysis shows that a number of the use cases, like the described Remote Operations of Experimental Facilities, perfectly fit into the "model use case" defined by RINGrid without extending it. There are, however, some sophisticated cases that require defining new actions in workflow description, and that is our current work.

6 Conclusions

In this chapter we have presented the main activities of the OGF research group RISGE-RG. This group includes worldwide representatives of Remote Instrumentation-related user communities as well as Grid middleware and infrastructure projects. As defined by the RISGE-RG charter, the group is documenting experiences, presenting best practices, and working on informational documents that would describe different use cases in common OGSA-style. Based on these collected use cases and results of the RINGrid project, the group would work on proposing a "model use case". First, results of this on-going activity have been presented in this chapter – one of the collected use cases as well as considerations on the "model use case" coming from the RINGrid project.

References

1. OGF RISGE–RG Charter. http://www.ogf.org/gf/group_info/charter.php?review&group=risge-rg
2. Remote Operations of Experimental Facilities. http://forge.gridforum.org/sf/docman/do/downloadDocument/projects.risge-rg/docman.root.use_cases/doc15295
3. RINGrid project. http://www.ringrid.eu/
4. RINGrid Report on requirements verification. http://www.ringrid.eu/public/deliverables/RINGrid-WP6-D6_2-2008-04-20-1-GRN-Report_on_requirements_verification.pdf

Chapter 2
Adapting the Instrument Element to Support a Remote Instrumentation Infrastructure

M. Prica, R. Pugliese, A. Del Linz, and A. Curri

Abstract GRIDCC was a project funded by the European Commission with the goal to extend the grid by integrating instruments and sensors with traditional computing and storage resources. This chapter describes how the GRIDCC's instrument element concepts have been implemented to support the challenging requirements of the DORII project and of a usable remote instrumentation infrastructure. The new Instrument Element has been designed in order to be easy to use for both the application engineers and the scientists who are the final users of both the instruments and the grid infrastructure.

Keywords Grid · Middleware · Remote operations · Remote instruments · Sensors · Online processing

1 Introduction

Current grid technologies offer unlimited computational power and storage capacity for scientific research and business activities in heterogeneous areas all over the world. Thanks to the grid, different virtual organizations can operate together in order to achieve common goals. As the technology of the classical grid infrastructure (typically composed of computing and storage elements) matured, the attention shifted toward the actual sources of data: the instruments and sensors. A significant effort worldwide is being put in providing a closer interaction between various types of instruments accessible from the grid on the one hand and the traditional grid infrastructure on the other hand. The goal is to provide an end-to-end grid environment for scientific activity, connecting the points of data collection with the points of data analysis. The GRIDCC [13] project proposed and realized a new grid component called the Instrument Element (IE) [20, 10, 21] that provides the traditional grid with an abstraction of real instruments and grid users with an interactive interface to control them.

M. Prica (✉)
Sincrotrone Trieste S.C.p.A., Strada Statale per Basovizza 14, 34012 Trieste, Italy
e-mail: milan.prica@elettra.trieste.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_2,

A new implementation of the Instrument Element (IE2) will be among the building blocks of DORII [6]. DORII (deployment of remote instrumentation infrastructures) is a 30-month project that started in February 2008 and is funded by the European Commission under the seventh framework program. It seeks to deploy a testbed for remote instrumentation, building on the experience of several preceding projects like RINGrid [27], GRIDCC, g-Eclipse [11], or int.eu.grid [16]. For the Instrument Element, DORII will mark the passing step from the prototype implementation to production quality software. The IE2 integrates easily to the Open Geospatial Consortium (OGC) [24] reference model and particularly to the sensor web enablement principles thanks to its flexible and extensible architecture. Also, the IE2 will be included in the Italian Grid Infrastructure [14].

2 Related Work

In September 2004, the GRIDCC project was launched by the European Union. Its main goal was to exploit grid opportunities for secure and collaborative work of distributed teams to remotely operate and monitor scientific equipment. GRIDCC also focused on adoption of the grid's massive memory and computing resources for storing and processing data generated by such equipment. Moreover, the GRIDCC middleware has been deployed on a few pilot applications with the aim to validate it and show its use and effectiveness, both in real contexts and in testbed environments. These applications have been selected in strategic sectors such as follows: (1) the remote control and monitoring of a large particle physics experiment [5]; (2) the remote operation of an accelerator facility [7, 25]; (3) the remote control and monitoring of a widely sparse network of small, but numerous, power generators (a real power grid); (4) the landslide hazards monitoring; (5) the ensemble limited area of forecasting meteorology; and (6) the device farm for the support of cooperative distributed measurements in telecommunications and networking laboratories.

The instrument element (IE) is a concept unique to the GRIDCC project. It is an abstraction of the instrument (or group of instruments) into a standard interface, which can be used within the rest of the GRIDCC architecture. The term instrument is used in this context to define a piece of equipment that needs to be initialized, configured, operated (start, stop, standby, resume, application-specific commands), monitored, or reset.

2.1 Motivations for the New Implementation

The 3 year long experience of the GRIDCC project indicated a set of requirements for the Instrument Element middleware. First, ease of use is a fundamental feature for the adoption of the IE framework by its target communities. In particular, attaching devices to the middleware must be kept as simple and intuitive as possible. Second, the security issues must be addressed properly. IE should provide an

effective multi-user environment with a locking mechanism that prevents concurrent access to operations that alter the instrument state. The locking mechanism should allow the reservation of an instrument or a group of instruments for the duration of an experiment. IE middleware must support the grid operations in the most transparent way possible for the final users, while taking care of all the necessary steps to allow safe connection to the grid world. Effective handling of the alarms and the events triggered by the instruments is another highly desirable feature. The orginal IE lacked a few of these features, most seriously the concurrency control mechanism. Besides, connecting new devices required a very complex and hardly intuitive procedure. All of the above suggested a radically different design approach for the IE middleware and thus the new implementation.

3 Design Approach

Despite the fact that the IE2 is a completely new implementation of the GRIDCC's Instrument Element concept, the existing IE WSDL [15] interface has been kept unchanged in order to assure the compatibility with the rest of the GRIDCC middleware, in particular with the virtual control room (VCR) [26]. VCR is a portal-based client for the GLite [12] Grid and GRIDCC instruments that integrates a set of collaboratory tools in support of team work. Minor WSDL changes will likely occur in the near feature to allow for additional features and will require some modifications on the VCR side as well. Just as the original IE, the IE2 runs in the Apache's Tomcat [3] container as an Axis [2] web service. IE2 is entirely implemented in Java. Unlike the old version, it does not use database back-end for storing configuration data and registering new devices. Instead, the framework uses the local XML files for storing such information. Another major difference is the user interface: the old IE had both a web-service interface for remote clients like the VCR and a proper web interface. The latter, however, could provide only remote access to the instrument, but was lacking the support for the grid security. With the new IE, we kept just the web-service interface. Currently, the VCR is the only front-end to the IE2, but other clients (e.g., g-Eclipse) might follow soon.

The interface to a single instrument is called instrument manager (IM). It is a protocol adapter that allows the middleware to talk to the physical instrument or, more precisely, its control system. The major novelty regards the IM creation process, which is described in detail in the next section.

Two locking mechanisms are provided: implicit locking of the IM access occurs automatically during the execution of commands that trigger state change on the instrument or modify the values of its parameters or attributes. Explicit locking of an instrument for a certain time period is triggered by a user command, e.g., to perform a set of operations on the instrument. The explicit locking is also the base for integration with the reservation service.

Instruments can be grouped in logical units called contexts, and contexts can be further grouped in other contexts creating a tree-like structure. Near future

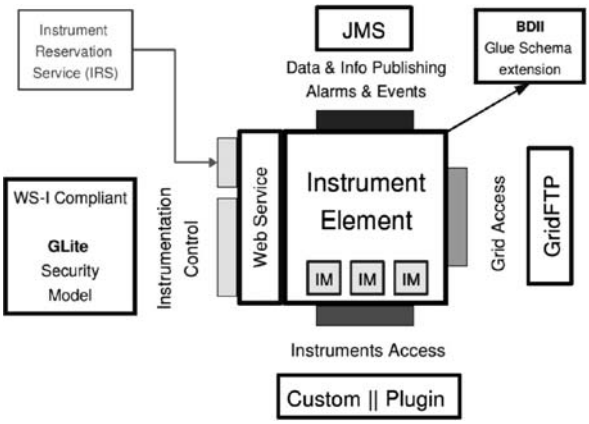


Fig. 2.1 Architecture of the Instrument Element 2

development will certainly include the locking of the entire context in order to support reservation of a set of instruments needed for complex experiments. A simple schema of the IE architecture can be seen in Fig. 2.1.

User authentication is performed using the GLite security model. The IE exports a web-service interface for delegation of the client's VOMS proxy certificates. Certificates are used for both user authentication and authorization. The client, in our case the VCR, delegates the user proxy certificate to the IE where the certificate is used for the user authentication. The VOMS attributes may be used for more fine-grained control of the user authorization. The proxy certificates are further used for accessing the grid, in particular the storage elements. The framework supports the grid operations providing a utility that allows instrument managers to store their output to a grid storage element transparently. All a developer of an instrument manager has to do is to invoke from the command implementing code the utility methods specifying only the basic parameters like the name of the file to be copied and its location on a storage element. The middleware performs the authentication behind the scenes and copies selected files using the GridFTP protocol.

Java Message Service (JMS) [17] allows for asynchronous reception of the messages from the instrument thus eliminating the need for the client polling. In Fig. 2.2 the difference between the JMS monitoring and regular time-interval polling for an attribute can be seen. All four graphs show the output (distance measure) from a proximity sensor. JMS monitoring shows its advantage over client polling both when the variable changes its value frequently (top graph) and in the opposite case when the variable value change occurs seldom (third graph). In the same situations, the regular time-interval client polling either looses intermediate values (second graph) or keeps polling the constant value (bottom graph), thus wasting resources. Another important use of the JMS in the IE2 framework is for signaling alarms and events to the users. IE2 has been tested with the Narada Brokering [23] and the IBM's proprietary implementation, Reliable Multicast Messaging (RMM) [28], but it should work with any provider conforming to the JMS standard.

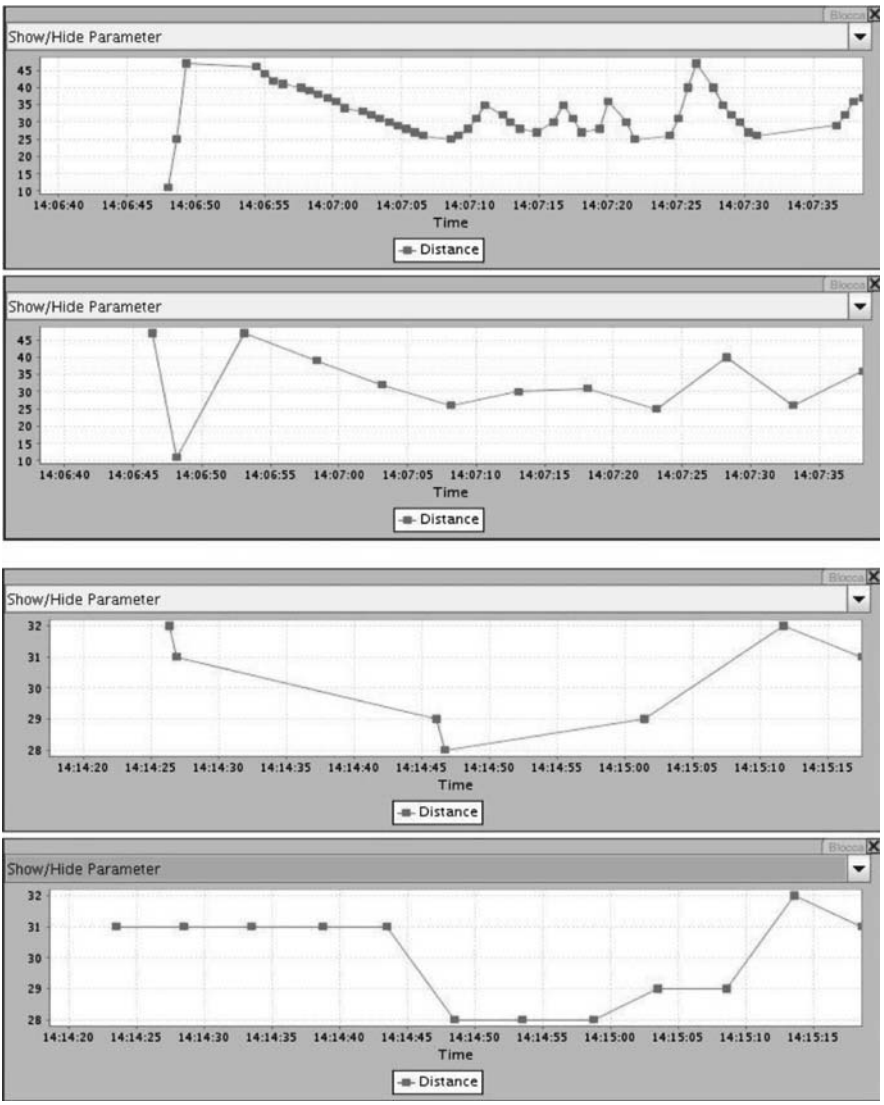


Fig. 2.2 Attribute value polling vs. the JMS monitoring

A centralized information system stores the end point addresses of the published Instrument Elements together with some additional information like status and operative parameters regarding those IEs. Information is kept consistent using the BDII [4] information system adopted by the LCG [18].

The new IE architecture makes the development of the generic plug-ins for control systems much simpler. We provide such a plug-in for TANGO [30], an object-oriented distributed control system based on CORBA, born at ESRF [8] and being

actively developed as a collaborative effort among four institutes: ESRF, SOLEIL [29], ALBA [1], and ELETTRA.

4 Implementation Issues

4.1 Implementing an Instrument Manager

Instrument manager is the actual protocol adapter used to access remotely an instrument. The first step in creating a new manager is writing the XML file that fully describes the instrument. This XML file contains the list of possible instrument states that together define the state machine, together with the details about the commands, attributes, and parameters. Next, for each command, attribute, or parameter, the developer must write the implementing class that extends the ones provided by the framework. Most of the time it suffices to implement a single method for each such class.

The instrument's XML descriptor must conform to the document type definition (DTD) file. Advanced IDE tools like Eclipse provide substantial aid in building and validating the file against DTDs.

Setup of four attributes is required to configure an IM: *name*, *implementation*, *id*, and the *initialStatus*. *Implementation* field contains the fully qualified name of the main class of the IM. *InitialStatus* indicates the state in which the instrument manager should be when first created. Note that the state of the instrument manager is not the same as the state of the instrument itself. An instrument may be already switched on and running while the IM is in the *off* state. (In such case, the meaning of the IM's *off* state might be disconnected from the instrument, while *on* state would mean successfully connected to the instrument.) The IE internally identifies each IM by a unique *id*. Such an *id* is also used to assign an IM to a context and to represent the IM on the client side. *Name* attribute has similar function and is used only internally in the IE, thus it is redundant and likely shall be removed in the future. An example IM and its state machine configuration may be seen in XML Descriptor 2.1.

Algorithm 2.1 IM and State Machine definition

```
<instrumentManager      name="Thermostat"
                        implementation="it.trieste.elettra.uos.test.impl.
                             Thermostat"
                        id="InstrumentManagerID" initialStatus="off">
<stateMachine>
        <status statusName="on"/>
        <status statusName="off"/>
        <status statusName="error"/>
        <status statusName="any"/>
</stateMachine>
...
</instrumentManager>
```

The IM – implementing class must extend the framework's *InstrumentManager* class. It is the class that actually connects to the physical instrument. All other classes that implement commands, attributes, and parameters use this one as the common link.

4.2 Attributes

Attributes are instrument (sensor) variables. Attribute values are normally set through commands, but since in some control systems (e.g., Tango) they may be set directly, we allow for that option as well. The outcome of setting an attribute is identical to running a command that alters that attribute value.

To configure an instrument attribute, the user must specify three mandatory fields: *name*, *description*, and the *enableInStatus*. All remaining fields are optional. The *enableInStatus* defines the state in which the attribute is active (and thus may be monitored) and the unique attribute name. The *name* identifies the attribute in the framework and in the user interface. Attribute's *description* is intended as a help for the final users. The optional *implementation* field contains the fully qualified name of the implementing class. If omitted, the framework will attempt to construct the implementing class name from the instrument manager implementation directory and the attribute name. Another optional field is *subscribable*: if set to *true* each variation of the attribute value registered by the IM will trigger a JMS message. *Lockable* indicates whether the access to the attribute may be locked. *Accessibility* field indicates the type of the attribute, read-only, write-only, or read and write. Read-only attributes cannot be locked. The last optional field is the *unit* of measurement. An example of attribute configuration may be seen in XML Descriptor 2.2.

Nested element *policy* defines how the instrument manager reads the attribute value. *Direct* means that each read is triggered by the user request, while *polling* periodically (*time* value in milliseconds) reads the value. Also, there is a possibility for the IM developer to define a custom policy extending a framework class. Nested element *value* defines the value type of the attribute. Handled types are *string*, *short*, *integer*, *long*, *float*, *double*, *calendar*, *vector*, and *enumeration*.

The implementing class must extend the framework's *Attribute* and implement the getter and setter methods for the reading and writing of the attribute's value on the physical instrument (in most cases, the setter will be a no-operation method, since physical instruments often do not allow for setting of the attributes).

XML Descriptor 2.2 Attribute definition

```
<attribute      enableInStatus="on" name="CurrentTemp"
                description="Read_the_temperature_value_from_the_sensor."
                implementation="TempSensor" subscribable="TRUE" unit="Degree_C">
        <doubleValue value="0.0" />
        <policy>
                <polling time="10000" />
        </policy>
</attribute>
```

4.3 Parameters

Parameters regard instrument settings only. Common practice is to set parameters directly, but they may be set also through commands. The procedure required to configure a parameter is almost identical to the one seen for the attributes. Configuring a parameter is the same as configuring an attribute, except that the *subscribable* and *lockable* fields are omitted. Also the required nested elements are the same as for the attributes. An example of parameter configuration may be seen in XML Descriptor 2.3.

The implementing class must extend the framework's *Parameter* and implement the getter and setter methods for the reading and writing of the parameter's value on the actual device.

XML Descriptor 2.3 Parameter definition

```
<parameter      enableInStatus="off" name="StratingTemp"
                description="The target temperature"
                implementation="" unit="Degree C">
        <doubleValue value="19.5" />
        <policy>
               <polling time="10000" />
        </policy>
</parameter>
```

4.4 Commands

Commands include both the state transitions and the regular commands that do not trigger a state change in the instrument. From the implementation point of view, the two are just the same. As with attributes and parameters, the implementation of a command should be written in a class that extends the one provided by the middleware, while the configuration is performed in the instrument deployment descriptor.

There are four mandatory fields that must be specified when configuring a command. The first one is the command *name*. The remaining three deal with the instrument state. The field *initialStatus* defines the state in which the command may be triggered, *finalStatus* is the uniquely defined state that results from the command execution. The *errorStatus* is the state to which the instrument moves in case an error occurs during the execution. Optional fields are *description*, *implementation*, and *lockable*. Nested elements to commands are command input parameters. Each command may have an arbitrary number of *commandParameters*. Commands configuration may be seen in XML Descriptor 2.4.

Command parameters also have a couple of required fields (*name* and *description*) and a couple of optional ones like the *unit* of measurement and a field that states whether that parameter is a *mandatory* input to the command.

For the attributes, parameters or command parameters that are numeric or calendar types, it is possible to define the range of admissible values (*min*, *max*) used

both by the IE2 and the VCR. An attempt of setting a variable to a value out of range would cause an exception.

XML Descriptor 2.4 Commands definitions

```
<command finalStatus="on" name="TurnOn" errorStatus="error"
     initialStatus="off" />
<command finalStatus="off" name="TurnOff" errorStatus="error"
     initialStatus="on" />
<command finalStatus="on" name="ChangeTemperature" errorStatus="error"
     initialStatus="error">
        <commandParameter name="Temperature" description="" mandatory="TRUE"
            unit="Degree C">
        <doubleValue />
        </commandParameter>
</command>
```

The *Command* class must implement the *executeCommand* method. Mostly, what it does is a simple forward of the user's call to the physical instruments API.

4.5 Deployment

Once the XML descriptor is completed and all the command, attribute and parameter classes are implemented, everything should be packaged in a jar file that must have the following manifest attribute: *InstrumentManager: DescriptorFileName.xml.* The IE ignores jars lacking such attribute. Jars are then deployed to the proper location in the Tomcat's webapp folder. Sample Ant build file provided with the framework greatly simply building, packaging, and deployment operations.

4.6 Front-End (VCR)

The VCR is currently the only available front-end to the IE2. Once registered in the Virtual Organization's BDII registry, Instrument Elements appear in the VCR's resource browser. A click on the IE link starts a https secure connection with the chosen instrument element and expands the resource browser view of the IE's tree-like structure containing contexts and instrument managers. Selected managers appear in the instrument control portlets, an example of which can be seen in Fig. 2.3. The VCR also offers a user-friendly interface to the traditional grid resources of the VO. A user can access data stored in the LFC file catalog and the storage elements or submit jobs to the workload management system by simply filling a form. Security is handled transparently for the user. First time registration into the portal requires the upload of the user's certificate into a MyProxy [22] server. For the successive logins into the portal, the user is asked for his user name and password only, while the VOMS authorization will be performed automatically. Collective use of the resources required by most applications can be programmed using the scripting capability of the VCR or by integrating external workflow management

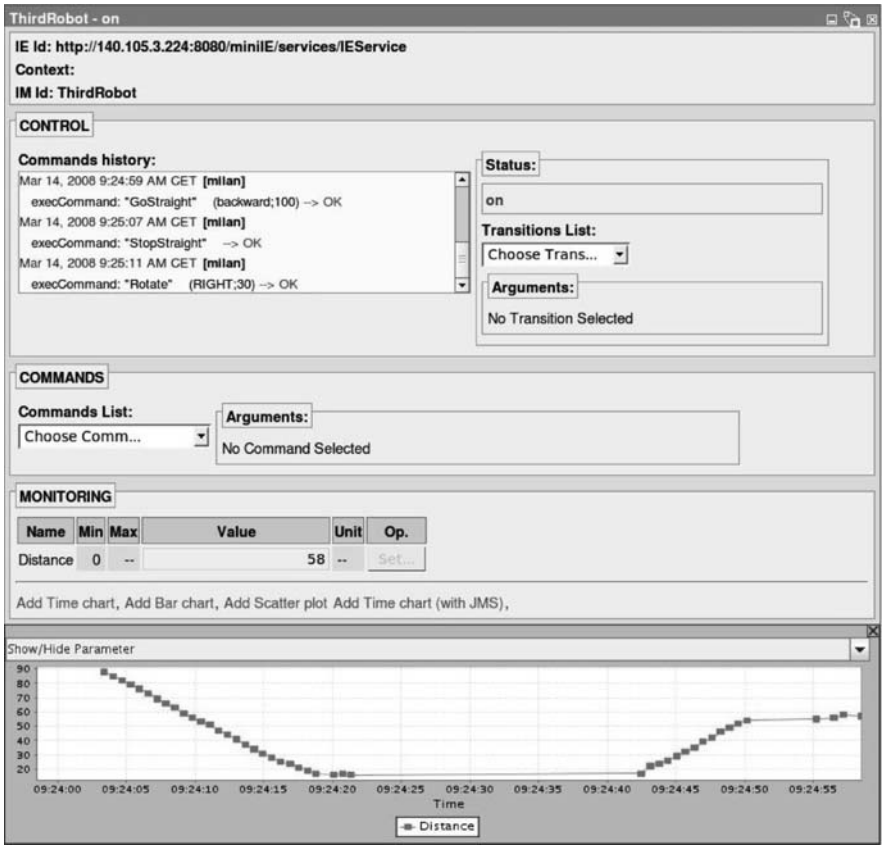


Fig. 2.3 Instrument control portlet: Lego mindstorm robot control

systems. A set of tools provided by the VCR (e-logbook, chat, integrated third party tools like Skype and EVO [9]) supports collaborative efforts.

5 Conclusions

A lesson we learned from GRIDCC is that the instrument element middleware should offer simple, straightforward procedures to the application developers in order to favor a wider adoption of the grid technologies. Otherwise, the risk is that the sheer complexity of the deployment process might mask the true benefits that this technology offers and discourage its widespread use. With the IE2 framework, we tried to make the developer's task much simpler than before.

Although the IE2 is yet to undergo extensive testing in the DORII pilot applications, we can already state that the IE2 provides a flexible solution for connecting a variety of devices to the grid. In particular, IE2 is suitable to control and monitor

both large and complex facilities such as ELETTRA accelerator, as well as simple sensors or even remotely controlled robots like the Lego Mindstorm [19].

An advanced GUI like the VCR makes a perfect front-end to the IE2 and together the two tools offer the scientific community an excellent entry point into the grid world.

References

1. ALBA – The Light Source for Spain. http://www.cells.es/
2. Apache Axis. http://ws.apache.org/axis/
3. Apache Tomcat. http://tomcat.apache.org/
4. BDII – Berkeley Database Information Index. https://twiki.cern.ch/twiki//bin/view/EGEE/BDII
5. CMS – The Compact Muon Solenoid experiment. http://cms.cern.ch/
6. DORII – Deployment of Remote Instrumentation Infrastructures. http://www.dorii.eu/
7. ELETTRA Sincrotrone Trieste. http://www.elettra.trieste.it/
8. ESRF – European Synchrotron Radiation Facility. http://www.esrf.eu/
9. EVO – The Collaboration Network. http://evo.caltech.edu/
10. E. Frizziero, M. Gulmini, F. Lelli, G. Maron, A. Oh, S. Orlando, A. Petrucci, S. Squizzato, and S. Traldi. Instrument element: A new grid component that enables the control of remote instrumentation. In *International Conference on Cluster Computing and Grid (CCGrid)*, Singapore, May 2006.
11. g-Eclipse Integrated Workbench Framework to Access the Power of Existing Grid Infrastructures. http://www.geclipse.eu/
12. GLite: Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite/
13. GRIDCC – Grid Enabled Remote Instrumentation with Distributed Control and Computation. http://www.gridcc.org/
14. INFN Grid – The Italian Grid Infrastructure. http://grid.infn.it/
15. Instrument Element VIGS WSDL. http://gladgw.lnl.infn.it:2002/rcms/services/IEService?wsdl
16. int.eu.grid – Interactive European Grid Project. http://www.interactive-grid.eu/
17. JMS – Java Message Service. http://java.sun.com/products/jms/
18. LCG – LHC Computing Grid Project. http://lcg.web.cern.ch/LCG/
19. LEGO MINDSTORMS NXT. http://mindstorms.lego.com/
20. F. Lelli, E. Frizziero, M. Gulmini, G. Maron, S. Orlando, A. Petrucci, and S. Squizzato. The many faces of the integration of instruments and the grid. *International Journal of Web and Grid Services*, 3(3):239–266, 2007.
21. F. Lelli and G. Maron. Integrating instruments into the grid. *Electronic Journal*, November 2006. GRID Today, Supercomputing '06 Special Coverage, http://www.gridtoday.com/grid/1086062.html
22. MyProxy, a Globus Project that Develops Software for Managing X.509 Credentials. http://dev.globus.org/wiki/MyProxy
23. Narada Brokering Project. http://www.naradabrokering.org/
24. OGC – The Open Geospatial Consortium. http://www.opengeospatial.org/
25. M. Prica, R. Pugliese, C. Scafuri, L. Del Cano, F. Asnicar, and A. Curri. Remote operations of an accelerator using the grid. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Grid Enabled Remote Instrumentation*, Signals and Communication Technology, pp. 527–536.

Springer US, 2008. ISSN 1860-4862, ISBN 978-0-387-09662-9 (Print) 978-0-387-09663-6 (Online).
26. R. Ranon, L. De Marco, A. Senerchia, S. Gabrielli, L. Chittaro, R. Pugliese, L. Del Cano, F. Asnicar, and M. Prica. A web-based tool for collaborative access to scientific instruments in cyberinfrastructures. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Grid Enabled Remote Instrumentation*, Signals and Communication Technology, pp. 237–251. Springer US, 2008. ISSN 1860-4862, ISBN 978-0-387-09662-9 (Print) 978-0-387-09663-6 (Online).
27. RINGrid – Remote Instrumentation in Next-Generation Grids. `http://www.ringrid.eu/`
28. RMM – Reliable Multicast Messaging by IBM Research. `http://www.haifa.ibm.com/projects/software/rmsdk/index.html`
29. Synchrotron SOLEIL. `http://www.synchrotron-soleil.fr/`
30. Tango Control System. `http://www.tango-controls.org/`

Chapter 3
Performance Analysis of a Grid-Based Instrumentation Device Farm

L. Berruti, F. Davoli, S. Vignola, and S. Zappatore

Abstract A specific implementation of the instrument element (IE) architectural component of grid-enabled remote instrumentation devoted to the control of a device farm of telecommunication measurement systems is examined. The IE adopts a web service interface to transfer commands and responses related to instrument configuration parameters, but it uses a publish/subscribe mechanism for the asynchronous transfer of messages containing measurement data to be frequently displayed.

Keywords Grid · Middleware · Tele-measurement · Tele-laboratories · Instrumentation

1 Introduction

Grid services [8, 2, 7, 14] or, more generally, the service-oriented architecture (SOA) [13] represents a key feature to enable the interconnection and effective cooperation of remote instrumentation. The latter includes a very large number of devices, with potentially widely different nature, scope of application, and aims, which ultimately share the common characteristics of being dedicated to some kind of measurement on physical phenomena and of being geographically sparse and operated by distant users.

Scientific communities embraced the previously mentioned technologies and run projects involving distributed resources and using massive data sets, in the context of the so-called e-Science.

In this scenario, the possibility to access complex (and expensive) instruments located in distributed laboratories as if they were local to users should encourage the collaboration across distributed research groups and the sharing of experimental

L. Berruti (✉)
CNIT – University of Genoa Research Unit, 16145 Genova, Italy
e-mail: luca.berruti@cnit.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_3,

results. Obviously, these "shared laboratories" should also reduce waste of money by optimizing the use of instruments, together with computational and storage resources.

Relevant research projects have been working or are currently working on these topics. The GRIDCC project [12, 17] was one among the main European efforts in this direction. It was aimed at extending grid technology to the real-time access and control of instrumentation. The CIMA (Common Instrument Middleware Architecture) project [6], supported by the National Science Foundation, proposed a framework for making instruments and sensor networks accessible in a standard, uniform way, for interacting remotely with instruments and for sharing the data they produce. The RINGrid project [18] currently explored the impact of new emerging technologies to exploit remote instrumentation, in light of user requirements. At a national level, the CRIMSON and VLAB projects [15, 19] investigated Grid architectures as possible solutions for data gathering, storage, and processing in the field of telemeasurement applications. The recently launched DORII action [16] leverages the experience of these and other European projects to deploy and operate infrastructure for e-Science applications, with the direct involvement of users.

The topic related to management of measurement equipment and remote experiment driving has been faced extensively, even in scenarios not strictly related with grid environments (see, among others, [4] in the general measurement case, and [5, 1, 20] in the case of measurements for telecommunication systems).

This chapter describes a framework for tele-measurements on telecommunication systems (a "device farm" of Telecommunication Measurement Systems), developed within the GRIDCC project. Only the aspects related to the overall software architecture and to the publication of data acquired from instrumentation through the network are addressed, while other important aspects, such as authentication, authorization, resource reservation, accounting, security, and so on, are not tackled.

Moreover, the results of some performance evaluations of the ensuing test-bed are also presented: specifically, the opportunity of exploiting a Java Message Service (JMS) to efficiently exchange data between the device farm and its users will be discussed.

The chapter is organized as follows: in the next section we sketch the overall GRIDCC architecture. Section 3 illustrates how communications among GRIDCC users and main GRIDCC elements take place. Section 4 presents the experimental setup and reports performance measurements, relative to the so-called out-of-band data transfers. Finally, in Section 5 conclusions are drawn.

2 GRIDCC Overall Architecture

The overall GRIDCC architecture (described in detailed in [9]) is sketched in Fig. 3.1, in its specialization to the "device farm" application. Specifically, the figure represents the main components of a platform, devoted to provide the remote access

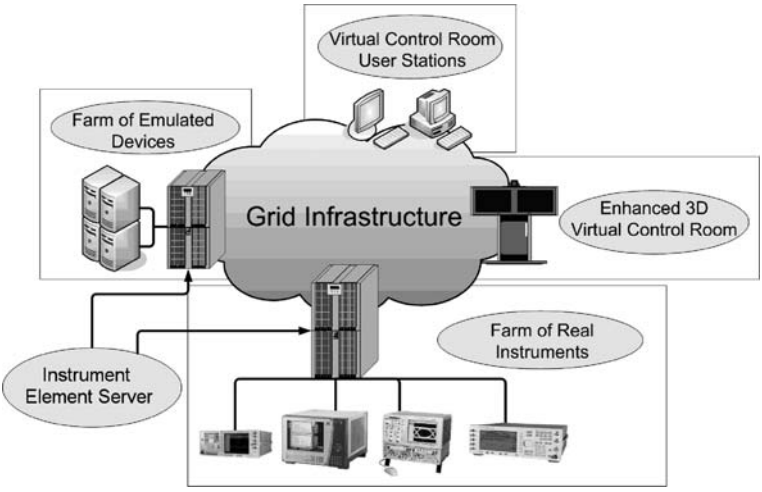


Fig. 3.1 Device farm overall architecture

to a "device farm," viz. a set of measurement devices or a complete laboratory. The key elements of such a platform are the virtual control room (VCR) and the instrument element (IE).

The virtual control room is the user interface of a GRIDCC application. Through the VCR, users can either monitor and control instruments in real time or submit workflows to the execution services, another GRIDCC element, whose functionalities are beyond the scope of this chapter. The VCR has been implemented within a grid portal framework (GridSphere [10]), in order to communicate with the underlying grid infrastructure. The final goal of the interface is to give the user the possibility to access remote instruments as if they were local to them. The term instrument element refers to a set of services, implemented as web services, which allow controlling and monitoring the remote physical instruments. Web services provide a common language to the cross-domain collaboration and, at the same time, hide the internal implementation details of accessing specific instruments.

The message and data exchanges of external entities with the IEs are based on web services (WS) standards: for instance, WSDL (Web Services Description Language) files are used to describe the services. Of course, this leads to the requirement that the IEs themselves follow these specifications. On the other hand, the communication between the IEs and the instrumentation is dependent on the installation and can be handled by any network protocol or even by a physical connection, different from one instrument to another. To this purpose, each IE includes a group of instrument managers (IMs), which actually perform the communication with the instruments. In a sense, IMs act as protocol adapters that implement the instrument-specific protocols for accessing its functions and reading its status. Since the instruments are heterogeneous in nature, there is a need to support many instrument managers, one instance for each physical instrument.

3 VCR - IE Communication

This section is devoted to illustrate how communications between the IE and a VCR take place. At first, the user has to start a specific experiment through the VCR, by issuing a series of commands in order to suitably set all the physical components involved in the experiment. These commands are sent to the IE by sequentially invoking a number of remote procedure calls (RPCs) provided by the web services associated to the IE. In turn, upon completion of each command, the IE returns a message reporting the current (new) status of the IE and, if required, some variables associated to the devices of the testbed or to the testbed itself.

For instance, in the case of a telecommunication system testbed used to study the characteristics of a certain DUT (device under test), during this phase the user must plan which instruments have to be involved in the experiments, as well as their physical connections. Then, the user configures all the devices of the testbed with the appropriate parameters/working points.

After these preliminary steps, the actual measurement phase can take place. The messages returned by the IE carry the results (e.g., an array of variables related to an oscilloscope or spectrum analyzer trace) of an experiment. The messages can be exchanged in two different modes.

In-band mode. The user issues a command (via a XML–SOAP remote procedure call) in order to get the variables that must be monitored (see Fig. 3.2a). This involves (i) the proper formatting of the command into a XML message addressed to the IE web service, (ii) the actual transmission of such a packet to the IE web service engine end point, (iii) the reception, upon completion of the command at the IE, of the XML response message, and (iv) the decoding/disassembling of the message that carries the result of the command issued. It should be noted that the IE, before sending the response, has to wait till a new steady state is reached. This assures, for instance, that the targeted device has finished to execute the invoked command(s), or that some new acquired data are available for the user, or the outputs of the device under test become steady after some stimulus. At the user end, some Applets receive the results and display them properly. According to this operational mode, an experiment consists of a sequence of excitations and responses. The user forces a new set of excitations and, in turn, the IE returns the monitored (scalar and/or array) variables, which describe the resulting system's response. Therefore, in this operational mode, a continuous reading (and displaying) of an oscilloscope trace involves the continuous sending/getting of commands/results, thus limiting the data transfer and, consequently, the refresh frequency of the trace on the virtual instrument display.

Out-of-band mode. After starting the experiment, the user may send a command to the IE in order to subscribe to the reception of a group of data (e.g., an array of variables corresponding to an oscilloscope or spectrum analyzer trace) from a certain device. Upon receiving this request, the IM controlling the device opens a channel toward a dispatcher (or broker in JMS terminology) and communicates back to the user Applet a sort of "locator" that specifies the channel used by the dispatcher (see Fig. 3.2b). Then, at the user end, the Applet connects (subscribes)

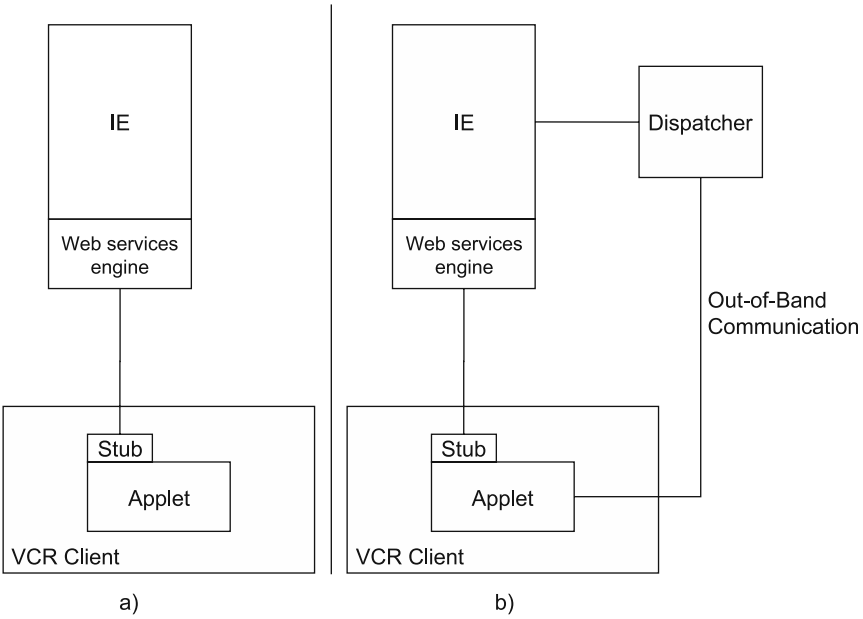


Fig. 3.2 Operational modes of communication between the user station and the GRIDCC device farm

to the dispatcher at the specified "locator" in order to automatically receive data whenever they are released by the instrument.

In other terms, while commands toward instrumentation are still issued according to the in-band mode, data acquired from instrumentation are forwarded to the dispatcher and then quickly delivered to the user. In this manner, the dispatcher takes care of the distribution of asynchronous data updates to the remote clients, independently of the presence of multicast support within the underlying network. It is worth noting that, though the same operation might be performed by the use of multiple threads at IM level, the adoption of an external dispatcher guarantees scalability.

Furthermore, data delivery on a separate, out-of-band, fast channel, significantly improves the overall reactivity of the system. A general and thorough analysis of these features is reported in [11].

Which mode is more convenient depends on many factors and on the kind of experiment itself. As a matter of fact, if the user is interested in knowing only the final state reached by a system after a certain excitation, the former operational mode is surely more appropriate; moreover, it calls for only a limited commitment of bandwidth resources.

On the contrary, if a high level of interactivity is required – for instance, in order to allow a user to havea feedback of what he or she is doing – the latter mode appears

more appropriate. Obviously, the higher level of interactivity requires more bandwidth needs, as well as increased computational effort by all the elements involved in the experiment's management.

In [3] the sustainable goodput was investigated under the in-band operational mode to acquire quasi-continuous arrays of variables corresponding to the traces of instruments, such as oscilloscopes and/or spectrum analyzers. The results, although heavily dependent on the hardware used to implement the IE and its IMs, show that the time spent by a user to get an array of 2500 doubles ranges from about 200 ms (when only a user accesses the system) to 2 s (when a group of 15 user stations are active). On the basis of these results, it is possible to affirm that whenever a medium–high level of interactivity is needed to carry out an experiment, and many client stations are involved, this operational mode does not represent an adequate solution to control a remote laboratory.

4 Performance Evaluation

A number of tests were carried out in order to evaluate the performance of data delivery via JMS in the device farm case. Moreover, JMS performance has been compared with that achieved with the IE operating in the in-band data exchange mode. The purpose of the tests is to estimate (i) how many clients the system can properly serve at the same time and (ii) the time spent to receive a specific array of variables. Obviously, the latter is strictly connected to the maximum throughput sustainable by the system in terms of data exchanged between a group of instruments and a set of users that subscribed to certain variables, describing the behavior of the experiment.

Due to the JMS implementation, a direct comparison with the case of in-band mode is impossible. The JMS employs the publish/subscribe paradigm: clients have to subscribe to a specific "topic" in order to get data. After subscription, clients automatically receive any changes on data regarding the topic. This approach completely differs from the one where only the IE manages the data communication and clients continuously poll the IE for any possible changes.

The performance evaluation was conducted by means of a virtual instrument that continuously sends data to a JMS topic as fast as possible. Let us assume that the virtual instrument generates messages (viz. arrays of variables) at the maximum rate sustainable by the network; then, clients will receive data at the maximum rate suitable by the JMS. In other words, it is possible to estimate the time spent by the JMS to serve a topic "update," by measuring the mean arrival time between two consecutive update notifications at the client end.

The performance evaluation was carried out by exploiting the experimental setup depicted in Fig. 3.3.

At the producer end, the virtual instrument continuously generates data. The JMS broker sends the data received from the virtual instrument to all the subscribed clients. Hence, the time spent by the JMS to serve each client can be estimated by varying the number of the client stations involved in the experiment.

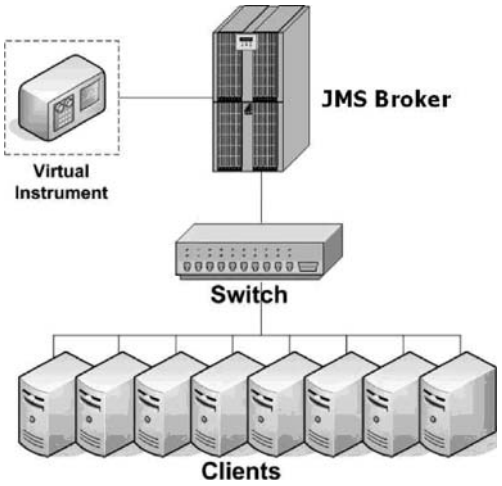


Fig. 3.3 The experimental setup used for performance evaluation

Furthermore, since the total net payload (consisting of measurement data) is known, the goodput and the related traffic load can be easily evaluated.

In more detail, the experimental tests consist of some measurement campaigns: during each campaign the inter-arrival time estimated in the presence of M client stations was obtained by averaging the times related to the reception of N consecutive messages at each station. Thereby, the results obtained for each client were further averaged, in order to evaluate the inter-arrival time of the entire campaign. For each campaign, let T_{ij} be the inter-arrival time of the *i-th* message at the *j-th* client. Then

$$T = \frac{1}{M} \sum_{j=0}^{M} \frac{1}{N} \sum_{i=0}^{N} \Delta T_{ij} \tag{3.1}$$

where T is the estimated overall average inter-arrival time.

The experimental results are presented in Fig. 3.4, which shows the behavior of the inter-arrival time vs the number of clients that have subscribed the topic. Specifically, the histogram reports the averaged inter-arrival time evaluated through each campaign (the first three bars in each group of experiments) and the overall mean inter-arrival time (the last bar in each group).

For comparison purpose, Fig. 3.5 reports the response time (i.e., the time spent by a client to get the same array of variables) for different numbers of active clients, which attempt to access the data gathered by the virtual instrument. The results were obtained by using an experimental setup similar to that of Fig. 3.3 (where the JMS broker is replaced by a proper IE), and the same hardware components. Looking at the comparison is indeed more meaningful than considering the absolute values of the results, because the latter, in general, depend on the type of hardware in use.

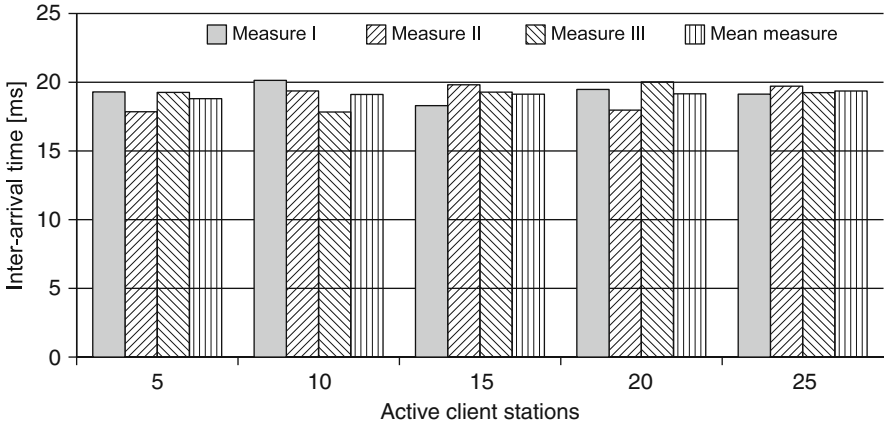


Fig. 3.4 Behavior of the inter-arrival time for different numbers of active client stations (JMS broker)

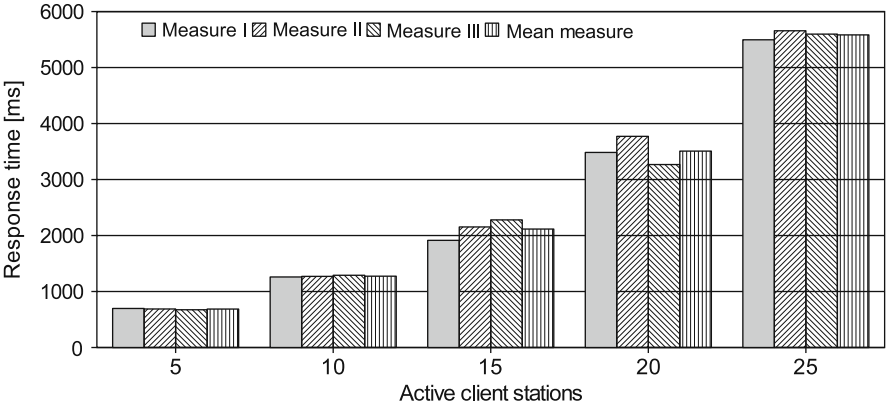


Fig. 3.5 Behavior of the response time for different numbers of active client stations (IE polling)

Observing the figures above highlights that the JMS inter-arrival time is about two orders of magnitude smaller than the response time achieved with simple use of an IE. In the case the JMS is adopted, the overall inter-arrival time seems not to be related to the number of client stations. Conversely, when the in-band operational mode is in use, the overall response time increases almost linearly as the number of active client stations ranges from 5 to 15, and then drastically increases if the number of clients is greater than 15.

The comparison between the previously mentioned operational modes clearly reveals that the JMS strongly enhances the overall performance of a device farm, thus allowing to use it for many applications, which require a medium–high level of interactivity. Furthermore, separating the control and actual data exchange functionalities can significantly increase the scalability of the system, as the latter might include a cluster of JMS brokers: in such a scenario, a JMS can serve a sole IM or a JMS can dispatch messages toward dynamic pools of subscribed clients.

Finally, a second set of tests was aimed at evaluating the behavior of the JMS at different levels of payload. More in detail, the virtual instrument was programmed in order to generate different numbers of variables for each array; specifically, 5000, 10,000, 20,000, 40,000 doubles per array.

Figure 3.6 illustrates the behavior of the inter-arrival time vs the number of active client stations at different levels of payload. It should be noted that the overall inter-arrival time increases slightly if the payload ranges from 5000 to 20,000 doubles. At the other extreme, if the payload is greater than 20,000 doubles, the response time significantly increases with the number of active client stations. The fact can be motivated by observing that, under this condition, the total traffic produced by the JMS broker approaches the physical channel capacity (a 100 Mbps Ethernet link).

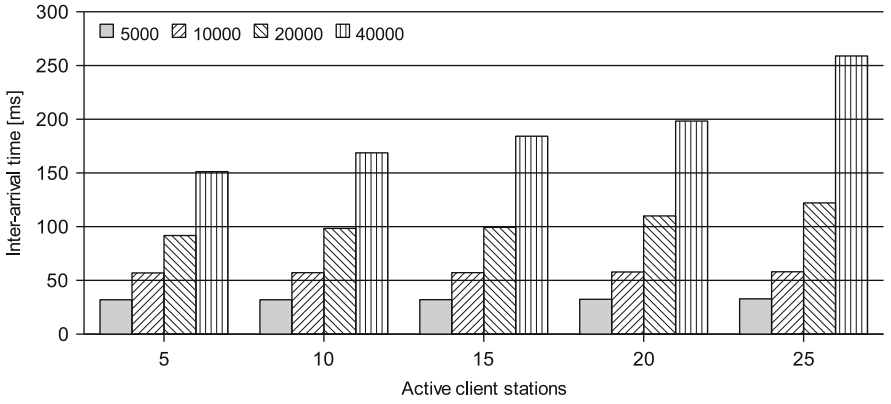


Fig. 3.6 Behavior of the inter-arrival time vs. the number of active client stations at different levels of payload

5 Conclusions

In a number of applications concerned with remote instrumentation, where access to a device farm of telecommunication measurement systems is involved, data must be frequently collected from the field and displayed on the user client stations. This is the case, for instance, of refreshing the traces of measurement devices that follow a time-varying phenomenon, which requires transmission of arrays of variables to numerous clients observing it. We have investigated the response time of an implementation of the instrument element, equipped with a publish/subscribe data transfer mechanism based on JMS. The results have been compared with those of an IE implementation that uses "in-band" data transfer, in response to a poll from the VCR, showing a relevant improvement.

Acknowledgment This work was supported by the European Commission, under the projects GRIDCC (contract no. 511382) and RINGrid (contract no. 031891).

References

1. O. Andrisano, A. Conti, D. Dardari, and A. Roversi. Telemeasurements and circuits remote configuration through heterogeneous networks: characterization of communications systems. *IEEE Transactions on Instrumentation and Measurement*, 55(3):744–753, Jun. 2006.
2. M. Baker, R. Buyya, and D. Laforenza. Grids and Grid technologies for wide-area distributed computing. *Software: Practice and Experience*, 32(15):1437–1466, Dec. 2002.
3. L. Berruti, L. Caviglione, F. Davoli, M. Polizzi, S. Vignola, and S. Zappatore. On the integration of telecommunication measurement devices within the framework of an instrumentation grid. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Grid-Enabled Remote Instrumentation*, pp. 283–300, Springer, New York, 2008. ISBN 978-0-387-09662-9.
4. M. Bertocco, S. Cappellazzo, A. Carullo, M. Parvis, and A. Vallan. Virtual environment for fast development of distributed measurement applications. *IEEE Transactions on Instrumentation and Measurement*, 52(3):681–685, Jun. 2003.
5. F. Davoli, G. Spanò, S. Vignola, and S. Zappatore. LABNET: towards remote laboratories with unified access. *IEEE Transactions on Instrumentation and Measurement*, 55(5):1551–1558, Oct. 2006.
6. T. Devadithya, K. Chiu, K. Huffman, and D.F. McMullen. The common instrument middleware architecture: Overview of goals and implementation. In *Proceedings 1st International Conference on e-Science and Grid Computing*, pp. 578–585, Melbourne, Australia, Dec. 2005.
7. I. Foster. Service-oriented science. *Science Magazine* 308(5723):814–817, May 2005.
8. I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2004. ISBN: 1-55860-933-4.
9. GRIDCC Deliverable 1.2. *The GRIDCC Architecture – An Architecture of Services for a Grid Enabled Remote Instrument Infrastructure Version 1.2*. `http://www.gridcc.org/documents/D1.2-revised.pdf`
10. GridSphere Portal Framework. `http://www.gridsphere.org`
11. F. Lelli. *Bringing Instruments to a Service-Oriented Interactive Grid*. PhD thesis, University of Venice "Ca' Foscari", Italy, 2007.
12. F. Lelli and G. Maron. The GridCC Project. In F. Davoli, S. Palazzo, and S. Zappatore, editors, *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*, pp. 269–277, Springer, New York, 2006.
13. Q. Mahmoud. *Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)*.`http://java.sun.com/developer/technicalArticles/WebServices/soa/`
14. Open Grid Forum. *Open Grid Services Architecture*, 2006. `http://www.ogf.org/documents/GFD.80.pdf`
15. Project CRIMSON (Cooperative Remote Interconnected Measurement Systems Over Networks). `http://www.prin-crimson.org`
16. Project DORII (Deployment of Remote Instrumentation Infrastructure). `http://www.dorii.eu`
17. Project GridCC (Grid Enabled Remote Instrumentation with Distributed Control and Computation). `http://www.gridcc.org`
18. Project RINGrid (Remote Instrumentation in Next-generation Grids). `http://www.ringrid.eu`
19. Project VLAB (Virtual Laboratory). `http://vlab.psnc.pl`
20. S. Vignola and S. Zappatore. The LABNET-server software architecture for remote management of distributed laboratories: Current status and perspectives. In F. Davoli, S. Palazzo, and S. Zappatore, editors, *Distributed Cooperative Laboratories: Networking, Instrumentation and Measurement*, pp. 435–450, Springer, New York, 2006.

Chapter 4
Experimental Characterization of Wireless and Wired Access in Distributed Laboratories

R. Soloperto, A. Conti, D. Dardari, and O. Andrisano

Abstract Remote access to distributed resources such as instruments is one of the most important challenges to increase the value of laboratories of excellence. Through a telemeasurement platform enabling cooperative experiments involving distributed resources, users would be able to reach the workbenches, configure instrumentation, programmable circuits, and network cameras, thus controlling the entire system under test. First, this chapter investigates different kinds of instrument connections in order to allow remote access and their performances in terms of *response time* of remote commands compliant with the IEEE488 standard. The different types of access techniques, investigated at our laboratory, are GPIB-ENET, IEEE802.3 LAN, and Wireless IEEE802.11b connections. Second, the *refresh rate* of NI LabVIEW remote virtual panels representing the instrumentation, estimated for various access techniques, is characterized.

Keywords Telemeasurement · Distributed resources · Cooperative work · Instrument connections · Wireless access

1 Introduction

In order to increase the intrinsic value of distributed workbench resources (such as instruments, programmable devices, and sensor networks) a telemeasurement platform involving geographically distributed laboratories interconnected through a heterogeneous network has been implemented at WiLab [19]. The interest for this topic has increased in the last years as evidenced by many research projects related to the concept of telemeasurement for remote resources extending their application range. Within several national projects (e.g., [10, 18, 17, 16, 4]) we enlarged

R. Soloperto (✉)
WiLAB, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
e-mail: raffaele.soloperto@unibo.it

This research is developed within EU RINGrid Project (contract number 031891) and EU DORII Project (contract number 213110).

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_4,

the telemeasurement concept in different directions related to the communication infrastructure, access typologies, configurable devices, interconnection matrix, etc., involving distributed laboratories. In [1, 14, 15] the concept of wide-sense telemeasurement has been presented, where both the instrumentation and the devices under test can be controlled remotely and cooperate to aggregate measurement results, thus obtaining a virtual instrument with enhanced capabilities. The key aspects of a performing telemeasurement platform are given by scalability, transparency, and of course data manageability, helped by good graphical virtual interfaces. Recently, some European projects [6, 5] are addressing the issue of integrating distributed instruments through the Grid paradigm.

The main goal of this chapter is to understand what is the impact of different access technologies, both wireless and wired, on the command response time and data throughput achievable, especially when heterogeneous networks are present.

Traditionally, virtual interfaces (VIs) are created through dedicated software libraries and drivers within proprietary development platforms (e.g., National Instruments LabVIEW [12] and HP VEE [8]). However, we also developed some Java interfaces to control instruments, which result to be lighter to be used also in mobile terminals, and less expensive than commercial platforms. These two solutions (NI or Java) provide to create an easy graphical programming approach, managing different commercial protocols, such as General Purpose Interface Bus (GPIB), LAN, RS232, and nowadays WLAN connections. In the last years, several activities on these issues have been developed at WiLab at IEIIT/CNR [9] and CNIT [3], University of Bologna, Italy, with interesting results (Fig. 4.1).

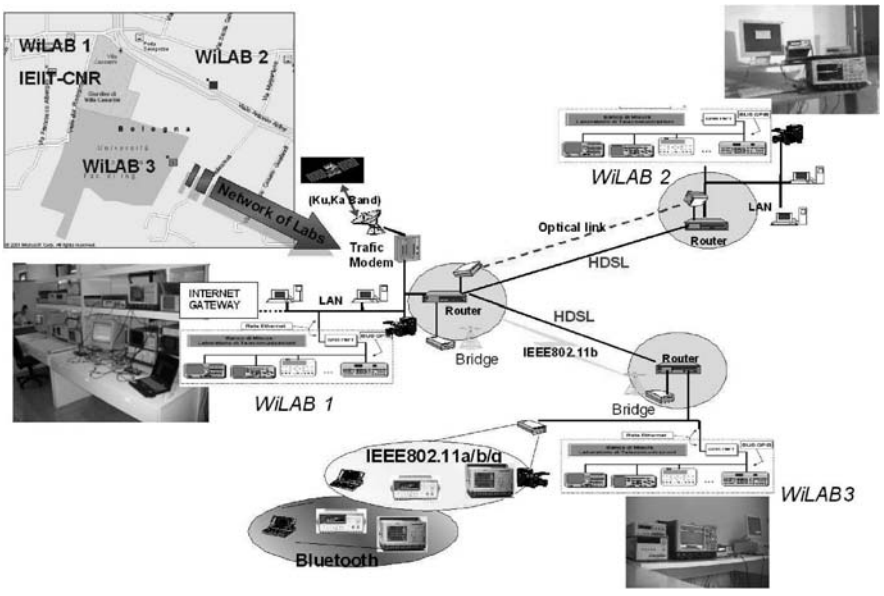


Fig. 4.1 Distributed laboratories on heterogeneous communication networks at WiLab, Bologna, Italy

The chapter is organized as follows: in Section 2 the complete WiLab architecture for distributed and cooperative telemeasurements is presented. In Section 3 all instruments connected and the modalities to measure and compare the different accesses are shown. Finally, conclusions are given in Section 4.

2 WiLab Architecture

The heterogeneousness of resources, communication, and access networks requires the design and implemenetation of a scalable architecture in which the addition of a new device in the network, represented by an instrument or programmable circuits, should be easy both in terms of network administration (no changes to the structure should be required to insert a device) and of its availability in telemeasurement experiments. Also, the whole structure should be scalable even if an entire new laboratory network is added to the system. In order to give the users the idea of an enhanced laboratory in which all the resources are aggregated increasing their functionalities, the organization of the laboratories and the management of the resources have to be transparent for users.

To reach these targets the architecture we designed and implemented is based on a hierarchical three-level structure as shown in Fig. 4.2. This structure has been already implemented in the three laboratories of WiLab and is based on (bottom–up order)

- LEVEL III: this level contains each single resource, such as instruments and programmable circuits under test connected to the network in different ways, for example, through GPIB-ENET, LAN, WLAN, RS232 connections.
- LEVEL II: this level is composed of a set of workbench slave servers each one controlling the laboratory they belong to. This level of servers (slave servers) has the role of managing the commands coming from the upper level to control the devices using the correct applications related to the particular resource required for the experience. Slave servers also answer upper-level requests with information and data provided by the controlled resources.
- LEVEL I: through a common web access the master server makes available to the users a unique interface toward all resources within the distributed laboratories. It has the role of managing all the information about the working session of the user, interacting with the slave servers to perform the experiment, retrieving information from the resources, and presenting results to the user via the VIs.

The master server is based on four different managers to fulfill several needs of a complete telemeasurement experience:

- Apache [2]: its role is to dialogue with the clients. The Apache server presents the remote user both static pages (like HTML pages with documentation of instruments and technical notes) and dynamic pages (like PHP [13] pages containing

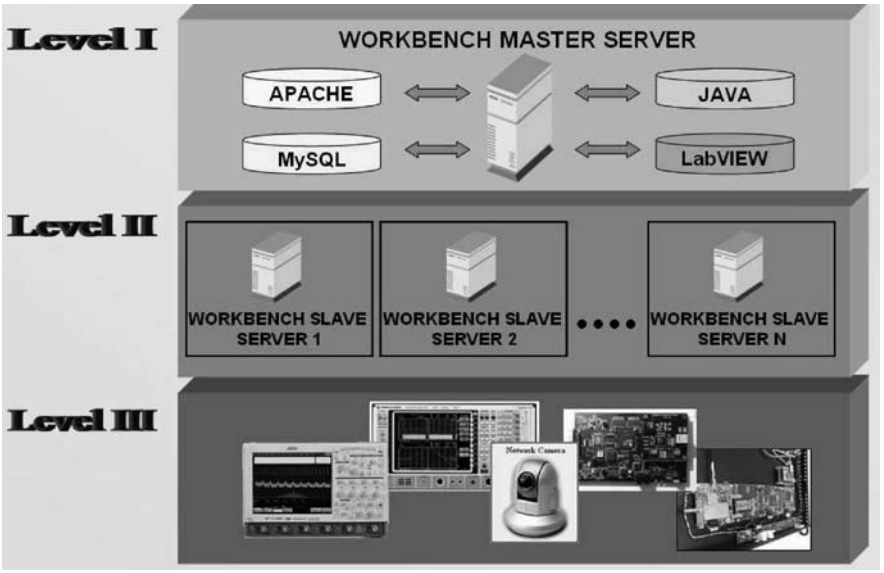


Fig. 4.2 Distributed laboratories architecture at WiLab, Bologna, Italy

information about the actual state of resources within the work session of the client). It is also able to report measurement results in the appropriate format according to clients' requests.

- MySQL [11]: its role is to save, manage, and upgrade information about the instrumentation and the programmable devices present in the distributed laboratories. For example, every time there is a request of control and the availability of the devices has to be checked. In a multiuser and distributed environment, the presence of the resource management system becomes strictly necessary to avoid conflicts among different users simultaneously accessing the same resources. For this reason a scheduling system has been implemented to allow users to register themselves in the system and then to grant the reservation of the requested set of resources.
- Java: its role is to allow the communication between the client interfaces and the controlled devices (when VIs are implemented in Java) using an XML-based protocol. It performs the translation of the device-independent commands into the specific communication protocol of the devices under control.
- LabVIEW: its role is to provide the clients with executable VIs to control the chosen instruments in case of use of previously implemented interfaces (the LabVIEW server is used for backward integration with previous platforms).

Our telemeasurement platform aims at configuring remotely more than 30 instruments of various brands and programmable devices based on digital signal processors (DSP) and field programmable gate arrays (FPGA) for telecommunication systems characterization. It is composed of digital transmission analyzer, up to 40 GHz

spectrum analyzers, vector signal generator and analyzer (with 50 MHz demodulation bandwidth), digital oscilloscopes, constellation analyzer, noise figure meters, noise and interference test set, RF channel simulators, switching matrix, up to 50 GHz signal generators, network analyzer, RF antenna sets, WLAN access points, and routers.

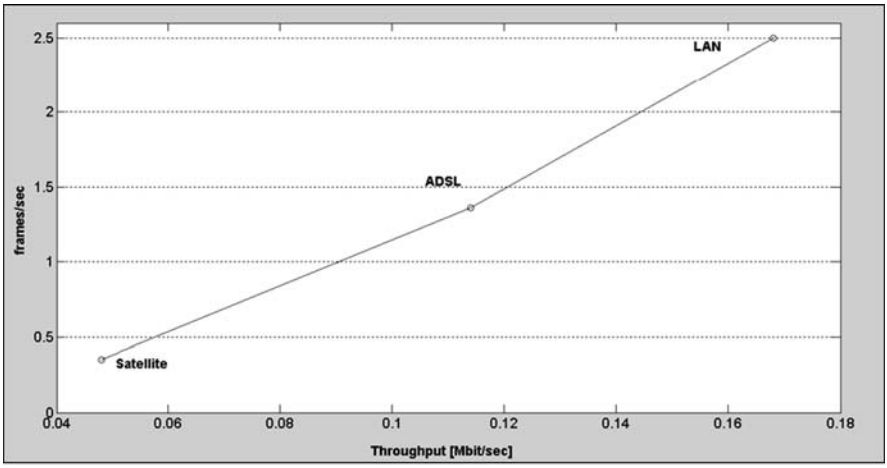
It has to be remarked that resources are heterogeneous in their functionalities and, in addition, they are also heterogeneous with respect to network connections. The next section will show how to characterize the performance of such a heterogeneous system.

3 Communication Networks Among Distributed Instruments

The evolution of wireless and wired access techniques and the standardization of communication protocols for instruments enable the possibility to control an entire workbench only using a remote workstation connected to an Internet access point. Users involved in the measurement campaign aim at downloading in few seconds the virtual panel, implemented in NI LabVIEW or Java, located on the workbench servers and controlling the real instruments. The access to the resources from users' terminals can be done through ADSL, Satellite (by using the CNIT satellite network), or LAN, while the connections between server and resources are granted by GPIB, LAN (only for last generation instruments), or WLAN, not yet adopted in current instruments but experimented in our labs. The next sections will show the two experiments realized at WiLab with the aim to characterize the user-perceived quality in terms of refresh rate of the virtual interfaces and the response time (RT, defined later) by varying the communication technology.

3.1 From Internet User to GPIB Resources

This experiment regards the refresh rate evaluation for remote VIs, representing the real instruments on workbench. In this kind of test the instrument chosen is a digital oscilloscope, in particular a LeCroy LC534A, able to communicate through the GPIB protocol. We measured the refresh rate for the LeCroy virtual panel, by varying the connections characteristics from user to master server. Results are depicted in Fig. 4.3. As shown, the Satellite channel does not permit to give the user the impression to work with a real instrument. Surely, the perceived quality does not allow some measurements, such as *persistence*, in which a minimum 1 frame per second is required. The other access techniques considered satisfy this parameter and can be considered suitable for telemeasurements. Note that the refresh rate increases as the data throughput achievable by the considered access techniques.



	Throughput (Mbit/sec)	Frames/sec
Satellite	0.048	0.35
ADSL	0.114	1.36
LAN	0.168	2.50

Fig. 4.3 Evaluation in terms of refresh rate (frames/s) perceived by an Internet user, varying the access network to the workbench

3.2 Remote Access by Intranet User

In this section the issues related to the resources' connection will be discussed. Three several workbench configurations and some cross configurations are shown. They are related to measurements done at WiLab laboratories about the performance of different access networks from an intranet wireless user when instruments are interconnected through GPIB (Fig. 4.4), IEEE 802.3 Ethernet Network (Fig. 4.5), and IEEE 802.11 b/g Wireless LAN Network (Fig. 4.6). Figure 4.4 shows a possible workbench configuration, using GPIB–ENET as a bridge device between GPIB and LAN network that grants the right communication with instruments. Starting from Fig. 4.4, the single link between instrument and network has been modified, avoiding the GPIB–ENET device and connecting the LAN cable directly on the instrument using the LAN interface on the rear panel (only for last generation instruments) (Fig. 4.5).

In this case the data rate for single link is surely higher than GPIB and the experiments that will be shown later aim at determining the response time (RT) in some particular cases. Finally, our idea is depicted in Fig. 4.6 and shows how it is possible to modify the link between instruments and network, introducing the concept of *wireless resources*. Each device is connected through an access point to the intranet

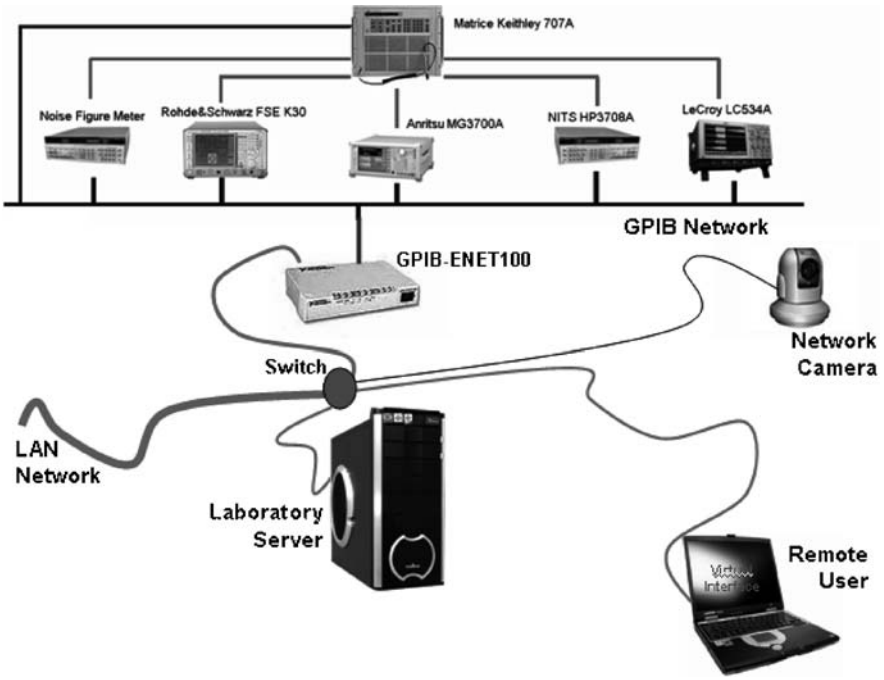


Fig. 4.4 Instrument connections through GPIB network

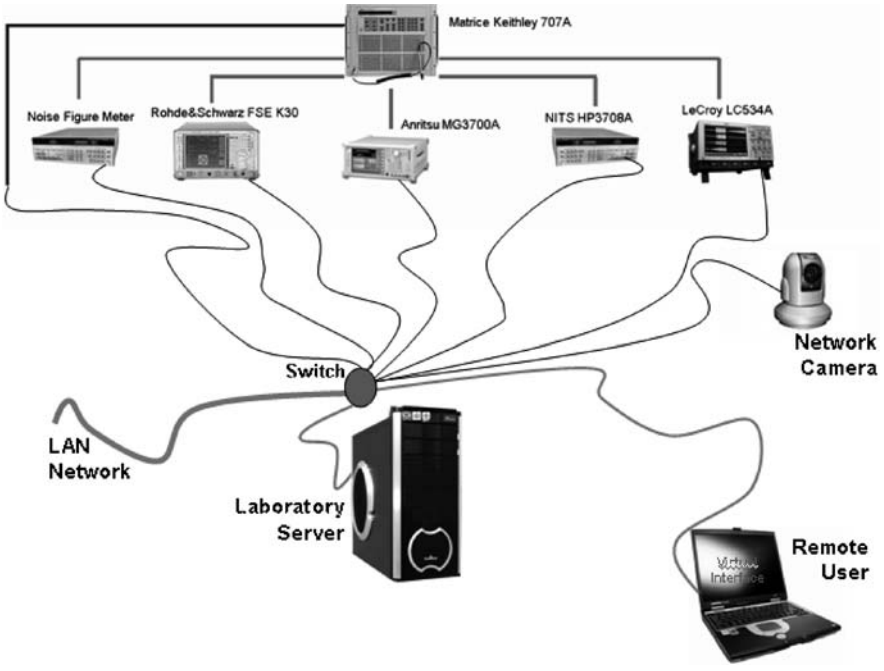


Fig. 4.5 Instrument connections through LAN network

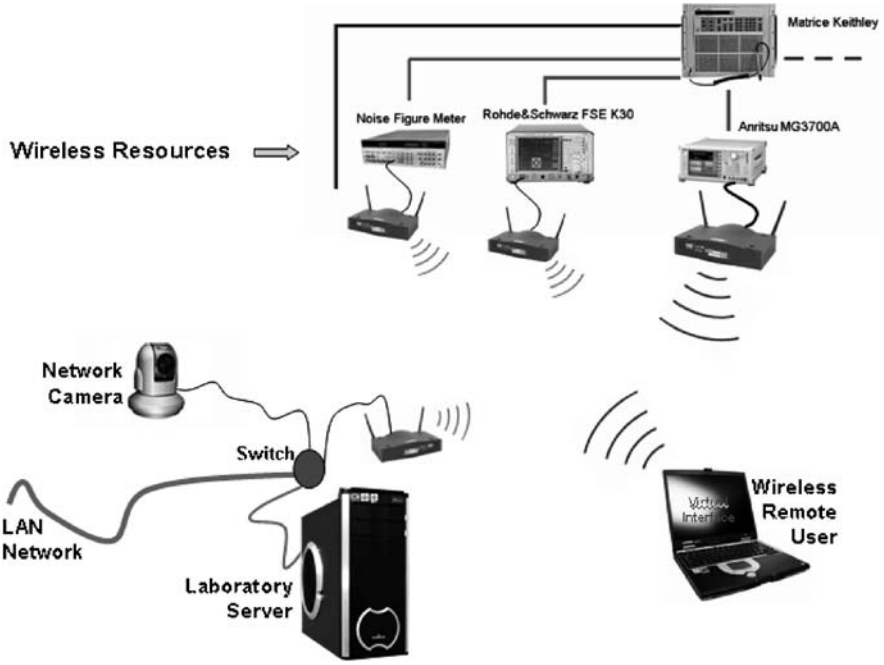


Fig. 4.6 Instrument connections through WLAN network

network and can be controlled through a wireless connection from a remote user (also wireless).

Results are reported in Fig. 4.7. It is noticeable that GPIB connections are slower than LAN or WLAN and show how the time needed to communicate with the instrument, defined as response time, increases extending the number of active instruments in the network. By the way, these concepts motivated a new communication protocol for instruments.

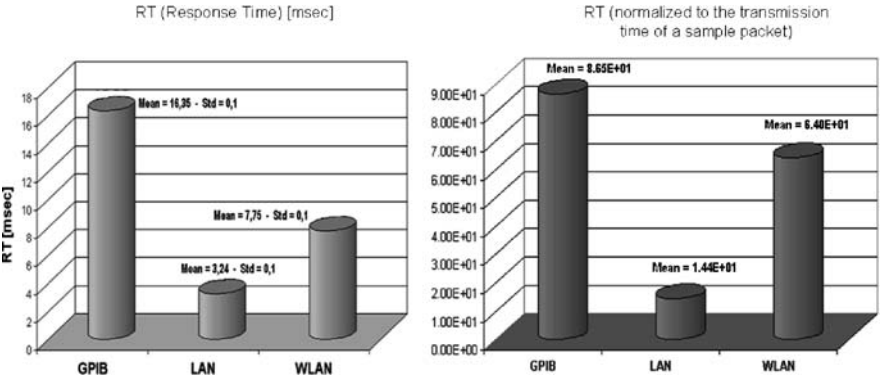


Fig. 4.7 RT in ms and RT normalized to the transmission time of a sample packet, evaluated for different access networks

In 2004, the LXI (LAN eXtensions for Instrumentation) [7] was defined as standard for test and measurement. It is the successor of GPIB, combining the advantages of Ethernet with the simplicity and familiarity of GPIB. The LXI standard has been rapidly adopted by several companies as the natural successor to GPIB.

They recognized that it was time for instruments to go beyond GPIB to make it easier for test system designers and integrators to create faster, more efficient systems. LXI reduces the time needed to set up, configure, and debug test systems, offering many advantages over GPIB-based systems. Even if no instrument belonging to WiLab is LXI compatible, some experiments considering LAN and WLAN protocols have been studied.

Description of the experiments and numerical results. Measurement performances in terms of RT for different networks, as depicted in Figs. 4.4, 4.5 and 4.6, have been experimented by fixing the same measurement device, able to receive remote commands from all network interfaces.
In particular an ANRITSU vector signal generator, MG3700A, has been chosen, in which frequency value and power level have been set to 70 MHz and −10 dBm, respectively, for all the experiments.

Thus, the RT (in ms) has been measured by sending the same remote command (so-called Query) to the instrument. Queries have been sent by a remote user connected in the intranet LAN, operating with a laptop in which O.S. Windows XP is installed and the instrument has been able to read frequency values in different connection conditions.

The first experiment regards the measurement on GPIB network, changing the number of active instruments (without generating other traffic) on the network. In Table 4.1 results obtained for the RT are shown.

Table 4.1 Time [ms] varying the number of switched-on instruments in the network

	1 instr. ON	2 instr. ON	3 instr. ON
Response time (ms)	18	34	53

These preliminary results show that the RT will increase about 18 ms for each active instrument. Then, if we want to have N instruments, the total RT needed to communicate with one of them is about N × 18 ms. In this case, other instruments used in this experiment are LeCroy LC534 Digital Oscilloscope and Rohde & Schwarz FSEK30.

To evaluate how to decrease the RT, LAN (about 100 Mbits/s), and WLAN (about 54 Mbits/s) networks have been experimented. In fact, a second experiment has been done in order to compare the different performances offered by LAN and WLAN networks versus GPIB connection (about 8 Mbyte/s), remaining in the same measurement conditions described above.

Figure 4.7 shows the results, in terms of RT (expressed in ms) and normalized RT [normalized to the transmission time of a TCP sample packet (1518 byte)], confirming the gap that exists between the data rate of LAN and WLAN network. RT is inversely proportional to the data rate: the RT in the WLAN is double that of the cabled LAN, as depicted in the figure.

4 Conclusions

In this chapter the access to remote resources and instruments through a heterogenous communication network has been characterized. Experimental results for the refresh rate and the response time in the case of different access technologies and instrument connections have been obtained with the hierarchical control architecture developed at WiLab. Our results show how the quality perceived by users, in terms of refresh rate, increases using connections with more data throughput. A comparison among different access technologies, included those based on a wireless link, has been reported. The results obtained may help choose the proper access technology as a function of the user and network requirements.

References

1. O. Andrisano, A. Conti, D. Dardari, and A. Roversi. Telemeasurements and circuits remote configuration through heterogeneous networks: Characterization of communications systems. *IEEE Transactions on Instrumentation and Measurement*, 55:744–753, Jun. 2006.
2. Apache HTTP Server Project. `http://www.apache.org`
3. Consorzio Nazionale Interuniversitario per le Telecomunicazioni. `http://www.cnit.it`
4. CRIMSON project, Cooperative Remote Interconnected Measurement Systems Over Networks.
5. EU Project DORII (Deployment of Remote Instrumentation Infrastructures). http://www.dorii.eu/dorii
6. EU Project RINGrid. `http://www.ringrid.eu/`
7. Lan eXtension for Instrument. `www.lxistandard.org`
8. Hewlett Packard. *Easy Productive Instrumentation Control*. January 1995 Edition.
9. Istituto di Elettronica e Ingegneria Informatica e delle Telecomunicazioni del CNR, Consiglio Nazionale delle Ricerche. `http://www.cnr.it`
10. MultiMedialita project – Interactive and multimedia Network to extend internet services.
11. *MySQL Database Server*. `www.mysql.com`
12. National Instrument. *LabVIEW 7 Express User Manual*. April 2003 Edition.
13. PHP web site. `www.php.net`
14. A. Roversi, A. Conti, D. Dardari, and O. Andrisano. A WEB-based Architecture enabling cooperative telemeasurements. In: F. Davoli, S. Palazzo, and S. Zappatore, editors, *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*, pp. 395–407, Springer, New York, 2006.
15. R. Soloperto, A. Conti, D. Dardari, and O. Andrisano. The WiLab telemeasurement platform for distributed resources on heterogeneous communication networks. In: F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Grid Enabled Remote Instrumentation*, pp. 253–268, Springer, US, New York, 2009.
16. SUMMIT project – Servizi Ubiquitari MultiMediali per l'Innovazione Telematica e Tecnologica. `http://summit.aster.it`
17. Teledoc2 project. `www.teledoc2.cnit.it`
18. VICom project – Virtual Immersive COMmunication. `http://www.vicom-project.it`
19. Wireless Communication Laboratory at the Faculty of Engineering, Bologna, Italy. Web site: `www.wilab.org`

Chapter 5
Virtual Laboratory and Its Application in Genomics

L. Handschuh, M. Lawenda, N. Meyer, P. Stępniak, M. Figlerowicz, M. Stroiński, and J. Węglarz

Abstract Nowadays, there is no science domain that does not use specialized software, on-line tools, and computational resources. Genomics is a new branch of science that developed rapidly in the last decade. As the genome research is very complex it must be supported by professional informatics. In a microarray field the following steps cannot be performed without computational work: design of probes, quantitative analysis of hybridization results, post processing, and finally data storage and management. Here, the general aspects of Virtual Laboratory systems are presented, together with perspectives of their implementation in genomics in order to automate and facilitate this area of research.

Keywords Virtual Laboratory · Genomics · Digital science library · Remote instrumentation

1 Introduction

There are numerous areas of science, industry, and commerce that require broad international co-operation for their success. A number of problems may be addressed by using sophisticated equipment and top-level expertise, which is often locally unavailable. Therefore, the development and dissemination of techniques and technologies that allow virtualized, remote, and shared access to industrial or scientific instruments is essential for the progress of society. The possibility of using scientific or industrial equipment independently of the physical location helps in the egalitarian attitude in using expensive facilities and for unification of communities and subsequently opens new opportunities for industry, science, and business. That is why a new branch of grid science – Remote Instrumentation Systems (RIS) – has become so important for balanced development of domains requiring expensive equipment.

L. Handschuh (✉)
Institute of Bioorganic Chemistry PAS, Noskowskiego 12/14, 61-704 Poznań, Poland,
Departament of Hematology, Poznań University of Medical Sciences, Szamarzewskiego 84, 60-569 Poznań, Poland
e-mail: luizahan@ibch.poznan.pl

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_5,

2 Virtual Laboratory

The specific representatives of the RIS systems are virtual laboratory systems (VL). One of them is PSNC Virtual Laboratory (VLab) [15], the research project developed in Poznań Supercomputing and Networking Center since the beginning of the year 2002.

In general, a VL is a distributed environment providing remote access to various kinds of scientific equipment and computational resources. In VLab users can submit their tasks in the form of a Dynamic Measurement Scenario – the sets of experiments and computational tasks of different kind. These tasks form a dependencies graph describing the connections between them and all possible flow paths; the actual flow path is determined upon run-time based on results obtained at each stage. The tasks are scheduled on a one-by-one (or group) basis, after the appropriate results from the preceding tasks are obtained. The Virtual Laboratory is not a standalone system. It was designed to cooperate with many other grid systems, providing only the purpose-specific functionality and relaying on well-known and reliable grid solutions. The most important system the VLab cooperates with is the Globus Toolkit [4] – in the scope of scheduling computational tasks, software services, and libraries for resource monitoring, discovery, and management. All computational tasks submitted in the VLab system are transferred to the Globus via the GRMS broker – an important part of the GridLab project. Among other external systems used by the Virtual Laboratory are the VNC system, DMS [8] (data management system), authentication and authorization modules.

2.1 Architecture

The Virtual Laboratory system has a modular architecture. The modules can be grouped into three main layers. The general architecture is presented in Fig. 5.1.

An Access Layer is the top layer of this structure. Basically, it contains all modules and components responsible for a VLab user access and graphical interface to the system (including a web portal) and data input interface. Below there is a Grid Layer which communicates with the external grid environment. It is responsible for user authorization and authentication, data management, and general task scheduling. Modules from this layer also handle the transfer of the computational tasks to the Globus system and gather feedback data and the computed results. The Monitoring Layer consists of lower-level modules, such as hardware dedicated schedulers, system monitoring, gathering accounting data. All the physical scientific devices are located in the Resources Layer, as well as modules responsible for their direct control.

On the Grid Environment side the most important element is the Globus system with all its components (GRMS, GridFTP, etc.). Globus also allows to execute computational tasks (batch and interactive) on a wide variety of grid applications.

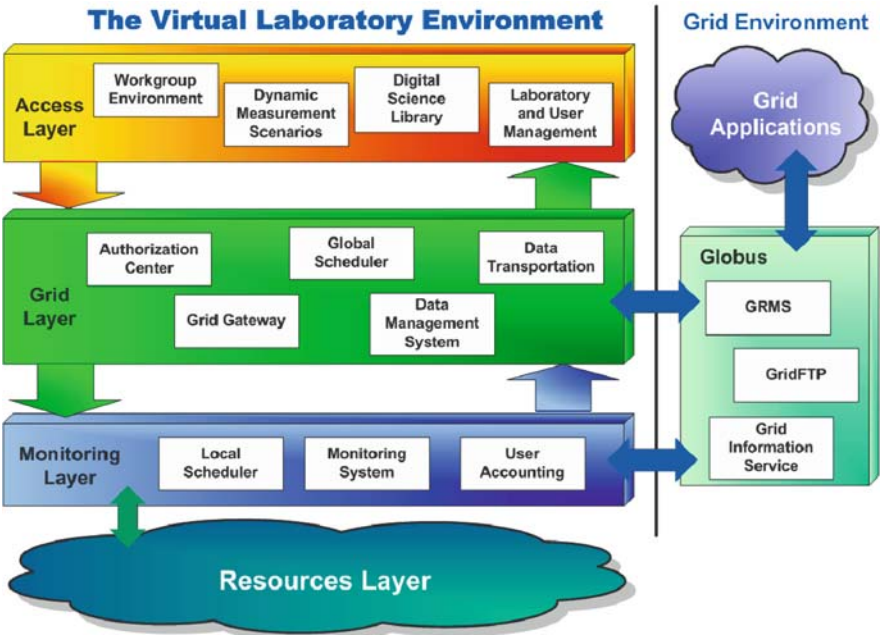


Fig. 5.1 General architecture of the virtual laboratory

2.2 The Peculiar Nature of the Virtual Laboratory Experiments

To explain how the VLab works the peculiar nature of this system must be described first. In general, there are two main kinds of tasks in the Virtual Laboratory: experimental and computational ones. The latter can be divided into regular (batch) jobs and interactive/visualization tasks (the ones performed in real time, directly by the users via the GUI). The biggest difference between those types is that in the interactive tasks the time slot reserved for running the task on a computational machine must be synchronized with user preferences, considering specific work hours, daily schedule, etc. Another aspect is the mechanism that will present the users with the graphical interface of the actual computational (or visualization) application – which is run on dynamically assigned computational server – and allow them to perform their interactive task.

Another very specific type of Virtual Laboratory tasks are the experiments. By the term experiment we mean a task scheduled to be performed on the remote laboratory equipment, available via VLab to its users. In most cases such experiments will be interactive processes, with users manipulating directly the remote equipment via a specialized control software GUI. In every science domain there can be many dependencies on external, often non-deterministic factors. The device will not be available for VLab users non-stop, but they will be shared with local researchers (usually with higher priority than the remote users). There are also maintenance periods, in which the device is unavailable. Sometimes the presence and

assistance of the device operator may be necessary – especially at the beginning of experiments.

The first scientific device incorporated into the VLab system was an NMR spectrometer. The most important problems with scheduling the NMR experiments, apart from those described above, come from the samples management. To perform an NMR experiment, an actual sample containing a chemical compound has to be delivered to the NMR spectrometer and inserted into the machine. At the task (and corresponding sample) submission point the actual NMR device has to be known and chosen, because the sample has to be sent from the remote location to the device site. The exact time of a sample arrival is not known as well, making the exact task scheduling impossible until the sample arrives.

2.3 Workflow Management

Experiments executed in the science laboratories are complex and consist of many stages. Usually it looks as follows: a scientist prepares a sample and/or input data (e.g., parameters) which will be measured/computed. Next he/she uses laboratory devices to obtain data, which are then processed by specialized software. Processing can include the visualization stage if it is needed to assess the results. In case of undesirable results some measurement stages should be repeated.

At each stage the scientist decides which way the research should go next. As we can see, the experiment process execution may consist of very similar stages in many scientific disciplines (laboratories). The given experimental process is often executed recurrently by some parameters modification. Obviously, the presented scenario is typical, but we realize that scholars can have more sophisticated needs.

Thus, we can create a graph, which describes the execution path specified by the user. Nodes in this graph correspond to experimental or computational tasks. Edges (links) correspond to the path the measurement execution is following. In nodes we have a defined type of application and its parameters. In links the passing conditions which substitute decisions made by the user are defined. These conditions are connected with applications, and let us determine if the application is finished with the desired results (if not, the condition is not met). This execution graph with specified nodes and links (with conditions) is called the Dynamic Measurement Scenario (DMS). The term “dynamic” is used, because the real execution path is determined by the results achieved on each measurement stage and can change.

To properly define the Dynamic Measurement Scenario the system has to have knowledge about available applications and connections they are enabled to create. In case of the laboratory type, where the processing software is well known, it seems to be easy (an example shown in Fig. 5.2). The issue will be more complex when we want to define the DMS model for a wider range of virtual laboratories.

To solve this problem we have defined a special language – Dynamic Measurement Scenario Language (DMSL), which determines the following parameters: names of the connected applications, a condition (or conditions) connected with

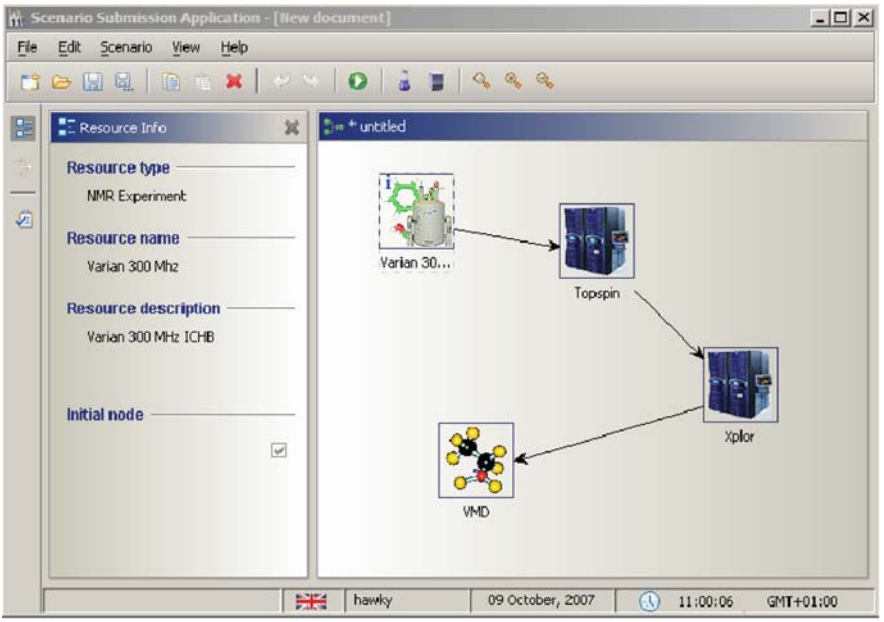


Fig. 5.2 An example workflow in scenario submission application

a given path, an additional condition which has to be met to pass a given path (e.g., when special conversion between applications is needed) and, finally, a list of input or output files for applications.

An expertise from a particular discipline is necessary to define rules for DMS. It can be done by a laboratory administrator together with a domain expert.

The DMS schema must be designed by the VLab administrator before it is available for users. Creating a new Dynamic Measurement Scenario needs a special attitude. We assume that some stages will be performed by the computer science engineer and scientist from a scientific discipline DMS concerns, and some by the application.

The design of the DMS consists of the following stages:

- analyzing the application;
- preparing the connection diagram;
- describing additional dependencies in the connection diagram;
- generating applications and links description;
- describing the measurement scenario.

All stages are tightly connected between themselves and should be executed in the presented sequence. The first three phases should be performed, as we mentioned before, by the VLab administrator and engaged scientist. The last two can be done by a special application, which takes into consideration user's rights and information introduced by the user. Next, we will analyze each stage of DMS designing.

2.4 Digital Library

The crucial component in most typical RIS (and VL) systems is a module responsible for data storage and management. Performing remote experiments on laboratory devices in a multi-user environment is related to the necessity of storing results' data which could be next used on the post-processing and visualization stage. This requirement is especially true for experiments where a certain number of devices need to work in parallel, e.g., in e-VLBI experiments.

The functional requirements that DSM systems should meet are not limited to the storage and management. These systems should provide their users with a possibility to publish electronic papers and present the digital content. This kind of tool is called Digital Science Library (DSL).

The DSL, which is implemented in the VLab system, is created on the basis of the Data Management System (DMS), which was developed for the PROGRESS project. The Digital Science Library should allow to store and present data used by the virtual laboratories. This data can be input information used for scientific experiments, as well as the results of performed experiments. Another important aspect is the capability of storing various types of publications and documents, which are often created in the scientific process. This type of functionality, which is well-known to all digital library users, is also provided by the DSL.

The main functional assumptions of the Digital Science Library can be described in reference to the reasons for founding the Digital Library of Wielkopolska (Wielkopolska Biblioteka Cyfrowa – WBC) developed by Poznań Supercomputing and Networking Center. Some of the most important assumptions are as follows:

- digital collections should be unique;
- digital library software should make it possible to significantly extend its usefulness;
- digital library software must cooperate with the library integrated systems and, in particular, allow using the catalogue database.

Referring to the first assumption, the Digital Science Library will serve the students and scientists by storing the results and computational data of the experiment. There is a need to digitalize the research in a form further called a digital publication. The second assumption is related to the usefulness that should be provided by the digital library in comparison with the existing libraries. A digital library is not supposed to constitute a separate unit or to be in competition with the existing libraries. It should complete and broaden the overall library usefulness, expanding its universality and availability on the one hand, and on the other allowing the librarian (or library administrator) to adapt the digital library functionality to its readers' (users) and administrator's needs.

In this context, the DSL exemplifies the digital library which, in assumption, will serve the scientific unit providing the selected instrument (e.g., NMR spectrometer) through the virtual laboratory. Scientists should have the possibility of collecting and accessing results of their experiments and computations. The collected

resources related to a defined institution and the provided instrument should have a teaching character, too.

The third issue concerns a semantic description of the stored resources, i.e., metadata. It allows searching the content of the library. The DSL realizes these assumptions in a complete manner – the publications can be described by any metadata set, including those renowned as quasi-standards (e.g., Dublin Core). The issue of the scientist's experiment results is similar; in this case, the available functionality is broadened by the possibility of storing the NMR data.

The functionality of the digital libraries. Over the last several years a few fundamental features which the modern digital library should provide have been established. The list below presents the most important ones:

- Categorization – the ability to group the related publications;
- Catalogue description – detailed information about the digital entries in the library;
- Searching – the possibility of looking up the library content with the user-defined criteria (paper search or the experiment results search);
- Browsing – visualization of the search results in the user-friendly interface.

Convenient and widespread access to publications. There are many scientists who find it necessary to publish their work results, materials, etc., in the digital library and share them with other users. However, to make it possible, the basic functionality described in the previous chapter is not sufficient. The Digital Science Library software has to be equipped with some additional features adequate to the requirements of the scientific environment. The list of such additions for the publication of documents and experiment results is presented below:

- **Widespread access to the published papers** – Internet access to the digital library resources has to be assured;
- **Lastingness** – a publication made available should never change its form (i.e. the file format) or the localization (i.e. network localization);
- **Version management** – the possibility of keeping the document in many versions and managing them;
- **Access management** – resource access protection. The access cannot be granted to the valuable, non-published resources or experiment results;
- **Copying protection** – protection from unauthorized resource copying;
- **Notifications** – the user should be notified about the changes of the library content;
- **Credibility** – the published document should not be changed by non-trusted or unauthorized users.

Many existing digital libraries (e.g., WBC) can be characterized by the common qualities described above.

The Digital Science Library is build on the basis of the Data Management System. This implies the list of features which are not so common in the digital libraries environments, but on the other hand are valuable and significant to the scientific

environment. DMS is a distributed virtual file system, extended by the data storing on the Hierarchical Storage Management (HSM).

2.5 The Example Diagrams

After gathering all data which are designed for putting them into the database, the use case diagrams should be prepared for most frequent scenarios. One of them is discussed below.

There we can distinguish the main actor on the basis of demands defined for the Digital Science Library of Nuclear Magnetic Resonance Spectroscopy. It is presented in Fig. 5.3. As an actor we can consider the system placed in the presentation or the indirect (middle) layer, on condition that it is able to communicate over the SOAP protocol.

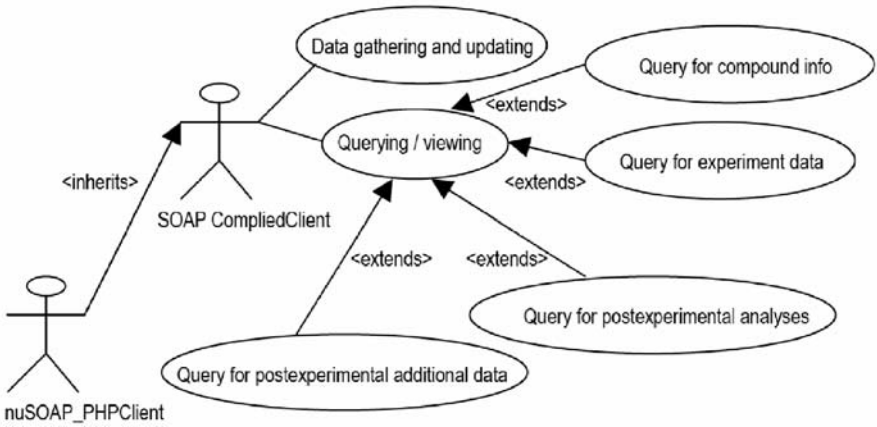


Fig. 5.3 Use case diagram

The following diagram presents the two actors. The first – SOAPComplied-Client – is identified with a certain pattern of actor with granted rights for executing the specified operations and provided with the ability of intercommunication with the library over the SOAP protocol. The nuSOAP (nuSOAP_PHPClient) is the second actor in the diagram. It can be used to create an access service for the NMR Library. In this case the access service is based on WWW pages and the PHP programming language.

The diagram shows some main examples of using the DSL–NMR library system:

- Data gathering and updating – enables introducing and updating information related to the chemical compound and the experiment data.
- Querying/viewing – enables retrieving information on a chemical compound, experiment and its results, and the analysis executed on the basis of that data. The

type of searching out the data depends on the needs. A standard list of requests consists of

- Query on compound info – condition: existence of information concerning re-searched chemical compound; searching out its attribute;
- Query on experiment data – condition: existence of experiment data; searching out the attributes of an experiment, e.g., type of spectrum, sequence of impulses, spectrometer type, solvent, measurement temperature;
- Query on post-experimental analysis – condition: existence of the analysis description; searching out of a chemical compound according to the value of chemical shift, coupling constant;
- Query on post-experimental additional data – condition: existence of the research documentation, e.g., reference on authors of experiments, for research papers documenting the received results.

3 Genomics

Genomics is one of the disciplines which can definitely benefit from the RIS system. Observed within the last 10 years, rapid development of genomics was driven by elaboration of new technologies, especially DNA sequencing and microarray technologies. Contrary to genetics, which focuses on individual genes, genomics concentrates on the whole genomes. Consequently, it attempts to answer the questions concerning the entire genome, e.g., what is the size and content of the genome, how it is regulated or how the expression of a particular gene influences all others genes. Although all cells forming an organism contain the same genetic information, their size, shape, and function can be distinctly different. This is possible since in each kind of cells a different set of genes is expressed. To release the information encoded in dsDNA, individual genes are transcribed into RNA, which then serves as a template for the protein synthesis. Accordingly, the present status of any cell can be determined by analyzing the overall RNA or protein content, called transcriptome and proteome, respectively. Thus, by applying DNA microarrays or other techniques used in functional genomic, one can precisely distinguish cells coming from different tissues, cells being at the different stage of development, or cells derived from healthy and pathologically changed organs.

3.1 Microarray Experiment Execution

A DNA microarray is a set of DNA molecules, called probes, spotted on a solid surface, for example, glass slide, in an ordered fashion [14, 13]. Each probe is complementary to a specific fragment of one gene. At present it is possible to manufacture DNA microarrays containing over six millions of probes located in 1 cm^2 [3]. Consequently, the expression of all genes can by analyzed simultaneously with a single microarray. At present many companies offer standardized, ready to use

microarrays [3, 6]. However, they are rather expensive and not available for each organism, especially when its genome is not sequenced. Alternatively, it is possible to design probes and produce home-made microarrays, e.g., dedicated to a particular research project. This approach is more flexible and economic.

The microarray experiment scheme is presented in Fig. 5.4. Construction of a microarray is a separate step that can be avoided when we use commercial microarrays. Another challenging task is preparing a sample to study with a microarray. First, all mRNA molecules (primary products of gene expression) are extracted from the target samples. Then mRNAs are converted into complementary single-stranded DNA molecules (cDNA) in the process called reverse transcription. During this stage cDNAs are also labeled with fluorescent dyes (e.g., cyanines). Sometimes, when the amount of biological material is very limited, cDNA amplification is required.

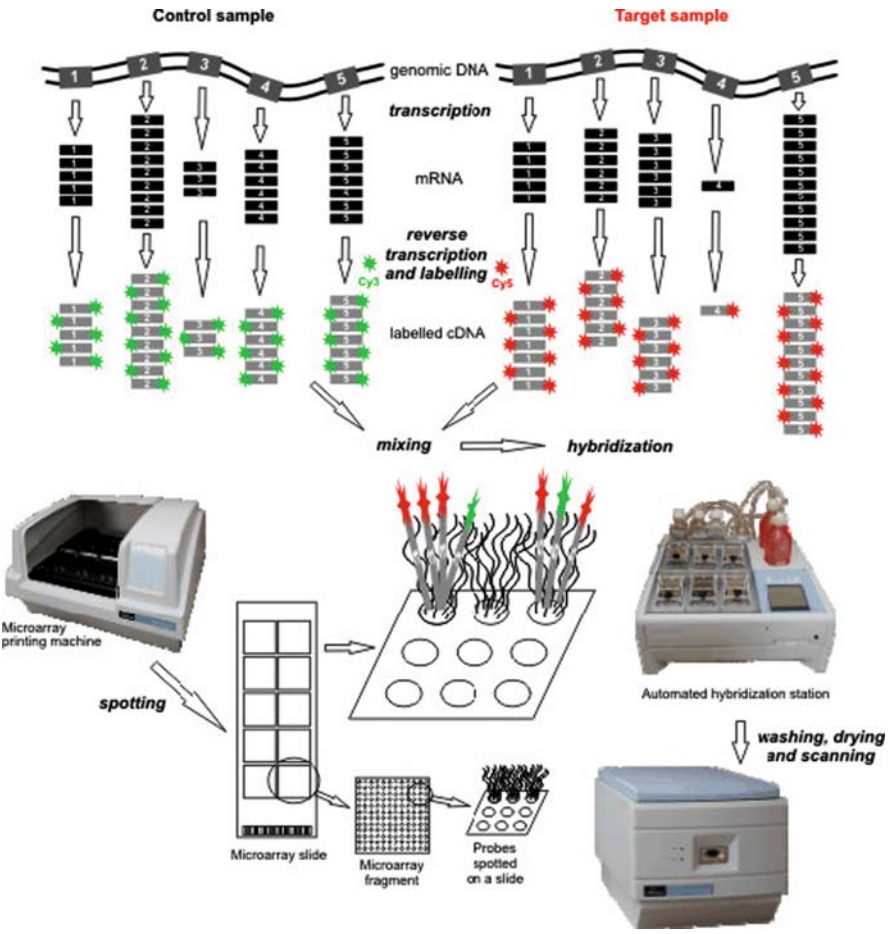


Fig. 5.4 Microarray experiment scheme

In case of two-color experiment, two samples – the control and the studied one – are prepared simultaneously. Each sample is labeled with a different dye (e.g., the control sample with Cy3, the examined sample with Cy5). Both samples are subsequently mixed and incubated overnight with the same microarray. During this stage, fluorescently tagged cDNAs hybridize with complementary probes. Theoretically each type of cDNA should only hybridize with one specific probe. Then the microarray is washed, dried, and scanned with two lasers. Each of them selectively detects signals coming from one of the applied fluorescent dyes. The intensity of the detected signal is directly proportional to the number of hybridized cDNA molecules, thus it also reflects the level of corresponding gene expression. As a result, two images are generated, the first one showing the level of gene expression in the tested and the second one in the control sample. The differences in the activity of the individual genes as well as the whole genomes can be superficially determined by superimposing both images. Usually, however, a more detailed statistical analysis is required. During the quantitative analysis images are converted to the numerical text data file. Raw data are subsequently submitted to the preprocessing – normalization procedures – and so-called low-level analysis, e.g., gene filtration, detection of differential expression. Normalization procedures minimize discrepancies between slides and allow for comparison of the results obtained from different experiments – high-level analysis, encompassing clustering of the samples and probes, correlation network modeling, construction of gene ontology trees and biochemical pathway, etc. [16, 11, 9, 12, 10]. Apart from many commercial programs dedicated to the analysis of micorarray data and less abundant free software, the most popular and flexible platform is Bioconductor in R environment [1]. Bioconductor is a still developing software project, offering a rich collection of tools (packages) useful for analysis and comprehension of genomic data. The results of analysis performed in Bioconductor can be deposited as *pdf*, *jpg*, or *bmp* reports.

3.2 Genomic Virtual Laboratory

There are numerous examples that DNA microarrays can be very useful in both basic and applied research, especially in medicine. It was demonstrated that DNA microarray-based analysis of medical samples allows doctors for fast and accurate diagnostics, selection of the optimal treatment method and reliable prediction of the results of applied therapy [2, 7, 5].

Unfortunately, common usage of DNA microarrays-based techniques is currently rather difficult. There are at least two major limitations: cost of the sophisticated equipment and a small number of appropriately educated specialists. Thus, two possible solutions can be applied in order to overcome the difficulties:

- specialized genomic centers;
- virtual laboratories.

Genomics centers are still more popular. In the USA and Western Europe such centers often function as integral parts of clinics, conducting the research projects as well as providing diagnostic services. In Poznań there is a Regional Genomics Center (RGC), a consortium founded by the Institute of Bioorganic Chemistry, University of Medical Sciences and Poznań University of Technology. It is a unique Polish laboratory equipped with a complete set of instruments necessary for the production, hybridization, scanning, and analysis of the commercial or home-made microarrays. Thanks to collaboration of scientists representing different branches of science (biology, biotechnology, medicine, and informatics) more complex and professional analyses can be performed there.

The genomic virtual laboratory is a better solution than a specialized genomics center as it provides common and parallel access for users that cannot directly collaborate with a specialized center. Theoretically, it should be possible to prepare the sample, send it by post and online control the instruments and programs that can be used for analysis of this sample. Unfortunately, not every step of a microarray experiment can be fully automated – some instruments still need to be manually operated. Probably more microarray tools will be web-available in the future. Nowadays, at least computational work connected with microarray probes design and analysis of microarray results can be automated. The rest of the experimental procedure has to be performed by a specialist. For example, a hospital which is not equipped with sophisticated devices and does not employ specialists trained in microarray analysis, can send a medical sample to a virtual laboratory, where the sample will be processed, and then monitor the process of a microarray-based analysis. At the end the obtained results can be confronted with the analogical data generated earlier with the samples of the same type.

As presented in Fig. 5.5, following scanning a microarray image (deposited in a *tiff* file) is quantitated using, e.g., Spotfinder or commercially available programs – ScanArray Express, Spot, GenePix, ImaGene. Raw data in a text file (*csv*, *gpr*, or similar) are then submitted to normalization, filtration, and high-level analysis, using sophisticated software (e.g., R Bioconductor, Multi, Experiment Viewer, Gene Spring, Genowiz). Result reports are saved as text or graphic files (e.g., *pdf*, *jpg*). Data transfer within programs usually demands file conversion. For example, *gpr* files generated by ScanArray Express do not follow the GenePix standards and are not automatically recognized by Bioconductor or by MIDAS (TIGR). Similarly, an output file generated by MIDAS is recognized by related MeV (Multi Experiment Viewer, TIGR), but not by Bioconductor.

3.3 Experiment Description

Measurement in a virtual laboratory starts from the post-processing stage. A microarray image can be stored in a digital library and next processed by particular applications according to the workflow diagram (Fig. 5.6).

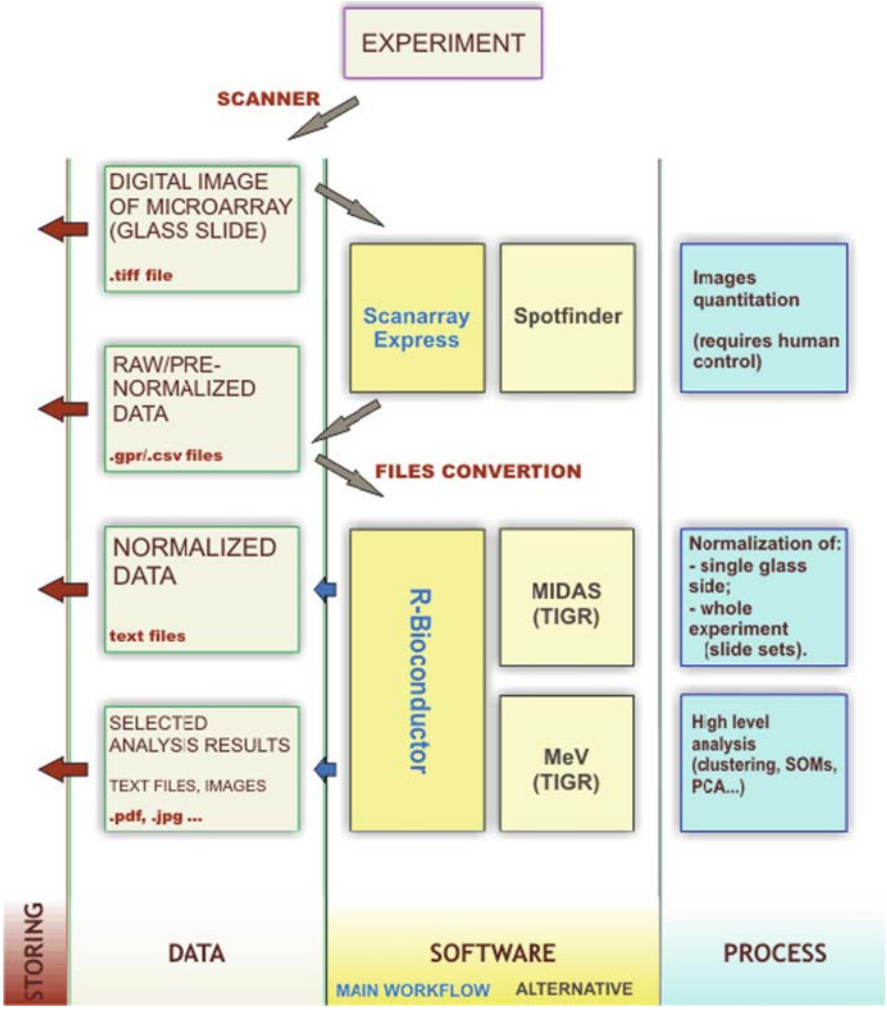


Fig. 5.5 Diagram presenting experiment execution in the virtual laboratory of genomics

The added value in this solution is automation of the quantitation procedure. End users can focus only on their research field – analysis – and draw conclusions from genomic data. All technical aspects related to server and software installation, creation of a new user account, setting up access rights, etc., are hidden from users. Another positive aspect is the possibility of storing all result data in a digital science library in an organized way. Output data from each stage are stored in it. It gives a possibility to go back in the experiment procedure and repeat any phase of research. Moreover, users can share experiment data among them and work together on a given project.

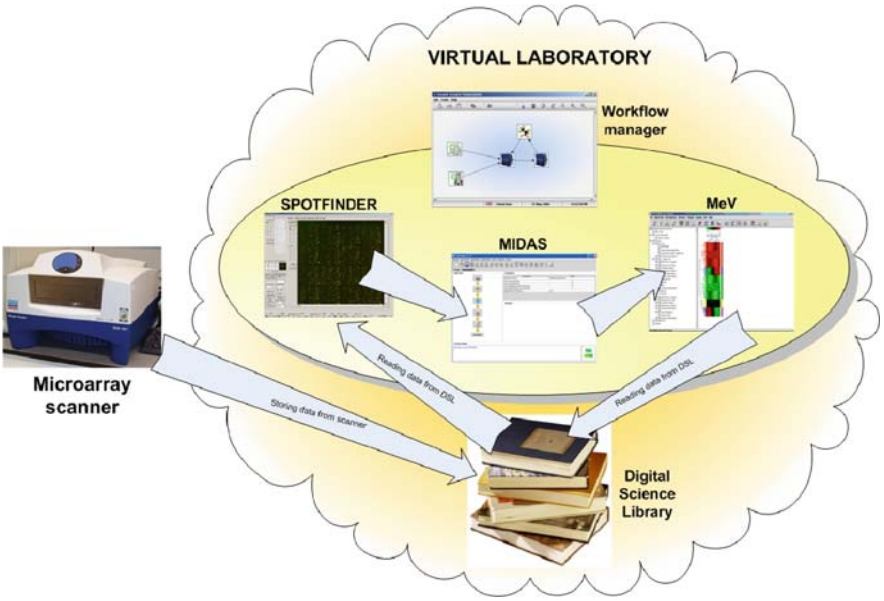


Fig. 5.6 Outlook of virtual laboratory of genomics

Finally, it is worth highlighting an aspect of simplicity in a virtual laboratory. The end user operates only on graphical applications with an intuitive interface. It could be very important especially for people without technical background.

4 Conclusions

Remote Instrumentation Systems become more and more popular among users wishing to use very rare and expensive scientific devices remotely. These systems, apart from remote access, provide an environment which facilitates experiments conducted on many stages.

Here, the general presumption of the virtual laboratory is presented, showing how it was applied in an NMR field, and the implementation of this idea in functional genomics is discussed. The main advantage of this system will be supporting users in a widely understood post-processing phase. It is assumed that processing scenarios are fully automated and aided by virtual laboratory software. One of the most important modules in this system is a workflow management tool, which allows to define the analysis path. Another one is the Digital Science Library used for output data storing and management. The DSL system also supports teamwork by sharing experimental results among different groups of users.

The purpose of future work will be developing a detailed project of the Virtual Laboratory of Genomics system, which will support and automate all activities related to processing data after the measurement phase.

Acknowledgment The work was supported by the grant from the Polish State Committee for Scientific Research No. PBZ-MniI-2/1/2005 to M.F.

References

1. Bioconductor project. `http://www.bioconductor.org/`
2. L.A. Garraway and W.R. Sellers. Array-based approaches to cancer genome analysis. *Drug Discovery Today*, 2(2):171–177, 2005.
3. D. Gershon. More than gene expression. *Nature*, 437:1195–1198, 2005.
4. Globus Toolkit. `http://www.globus.org/toolkit/`
5. T. Haferlach, A. Kohlmann, S. Schnittger, M. Dugas, W. Hiddemann, W. Kern, and C. Schoch. Global approach to the diagnosis of leukemia using gene expression profiling. *Blood*, 4:1189–1198, 2005.
6. D.N. Howbrook, A.M. van der Valk, M.C. O'Shaugnessy, D.K. Sarker, S.C. Baker, and A.W. Lloyd. Developments in microarray technologies. *Drug Discovery Today*, 8(14):642–651, 2003.
7. O. Margalit, R. Somech, N. Amariglio, and G. Rechavi. Microarray-based gene expression profiling of hematologic malignancies: basic concepts and clinical applications. *Blood Reviews*, 19:223–234, 2005.
8. PROGRESS – Polish Research on Grid Environment for SUN Servers. `http://progress.psnc.pl/English/index.html`
9. J. Quackenbush. Microarray data normalization and transformation. *Nature Genetics Supplement*, 32:496–501, 2002.
10. B.I.P. Rubinstein, J. McAuliffe, S. Cawley, M. Palaniswami, K. Ramamohanarao, and T.P. Speed. Machine learning in Low-level microarray analysis. *ACM SIGKDD Explorations Newsletters*, 5(2, m14), 2003.
11. G.K. Smyth and T.P. Speed. Normalization of cDNA microarray data. *Methods*, 31:265–273, 2003.
12. G.K. Smyth, Y.H. Yang, and T.P. Speed. Statistical issues in cDNA microarray data analysis. *Methods in Molecular Biology*, 224:111–36, 2003.
13. V. Trevino, F. Falciani, and H.A. Barrera-Saldaña. DNA microarrays: a powerful genomic tool for biomedical and clinical research. *Molecular Medicine*, 13:527–541, 2007.
14. S. Venkatasubbarao. Microarrays – status and prospects. *Trends Biotechnology*, 22:630–637, 2004.
15. Virtual Laboratory PSNC. `http://vlab.psnc.pl/`
16. Y.H. Yang, S. Dudoit, P. Luu, D.M. Lin, V. Peng, J. Ngai, and T.P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variations. *Nucleic Acids Research*, 4, e15, 2002.

Part II
Grid Infrastructure, Services, and Applications

Chapter 6
The European Grid Initiative (EGI)

Towards a Sustainable Grid Infrastructure

D. Kranzlmüller, J. Marco de Lucas, and P. Öster

Abstract Grid computing is a technology for distributed storage and processing of data, providing virtualised access to large-scale computing and other networked resources. Over the past decade, European states and the European Commission have invested heavily in grid technologies and Europe has reached a leading position in grid development and deployment. Moreover, Europe has also been highly active in accelerating the corporate and social usage of grids. Today's European grid infrastructure already supports a wide range of applications that are being developed, from testbed projects to efforts in infrastructure deployment and management. In addition, many European countries have initiated national grid projects and e-Science projects.

Keywords Grid computing · Grid technologies · e-Infrastructures · Sustainability · European grid infrastructure · National grid initiatives

1 Introduction

Already today, many research communities and application domains rely on grids. A new challenge for the European research community is to satisfy the need to protect the existing services and the investments made by both the user communities and funding organizations. The cyclic-based project funding for e-Infrastructures has been crucial in reaching the present high level of grid developments. However, this is not sufficient anymore, and if Europe wants to maintain its leading position on the global science arena, there is an urgent need to ensure a reliable and adaptive support for all sciences. Today's grid users range from grid enthusiasts to large research collaboration projects involving major organisations and institutions that are heavily dependent on grids. Tomorrow's grid users, especially the latter, are already asking for a long-term perspective for their investments (Fig. 6.1). Therefore,

D. Kranzlmüller (✉)
Ludwig-Maximilians-Universitaet (LMU) Muenchen & Leibniz Supercomputing Centre (LRZ), Germany
e-mail: kranzlmueller@ifi.lmu.de

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_6,

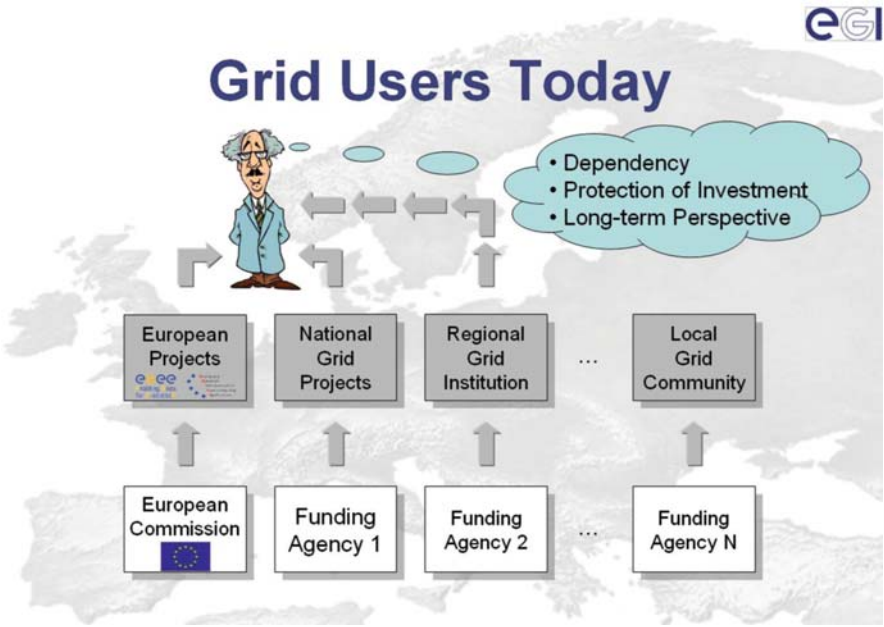


Fig. 6.1 Grid users today

the creation of a sustainable grid infrastructure is the logical next step in the development.

The idea of sustainability is supported widely within the whole grid community. The e-Infrastructure Reflection Group (e-IRG) [1] published a recommendation in December 2005 announcing that "The e-IRG recognises that the current project-based financing model of grids e.g., EGEE [5] (Enabling Grids for e-sciencE), DEISA (Distributed European Infrastructure for Supercomputing Applications) [4] presents continuity and interoperability problems, and that new financing and governance models need to be explored – taking into account the role of national grid initiatives as recommended in the Luxembourg e-IRG meeting".

The sustainability has also been strongly appointed by the European Commission. Viviane Reding, Commissioner of the European Commission, stated at the EGEE'06 conference that, "for grids we would like to see the move towards long-term sustainable initiatives less dependent upon EU-funded project cycles". This can be seen in the financing trends of the European Commission's different funding programmes. The 5th Framework Programme concentrated on broad-scale testbeds while the 6th Framework Programme encouraged for a production of quality facilities. The 7th Framework Programme opens a new page by supporting the sustainable grid and data-based e-Infrastrutures. This should also foster the emergence of new organizational models to consolidate a sustainable approach to e-Infrastructures, in particular in the domain of grids and data repositories. Also new

service provisioning schemes should be more neutral and open to all user communities and resource providers.

The European grid initiative (EGI) [2] will enable the next leap in research infrastructures by defining and installing the EGI Organization (EGI.org) for the support of all sciences in the European Research Area (ERA). The national grid initiatives (NGI) will form the core of the EGI. The EGI.org, whose headquarters will be located in Amsterdam, will coordinate and operate a common multi-national, multi-disciplinary and multi-technology grid infrastructure. The organization will enable and support international grid-based collaboration, provide support and added value to NGIs and liaise with corresponding infrastructures outside Europe.

The goal of the EGI Design Study (EGI_DS) [6] project is to create a conceptual setup and operation of a new organizational model of a sustainable pan-European grid infrastructure. The project evaluates use cases for the applicability of a coordinated effort, identifies processes and mechanisms for establishing the EGI, defines the structure of a corresponding body and initiates the construction of the EGI. The EGI_DS has 9 principal partners and is supported today by 42 national grid initiatives. The project was launched in September 2007 and will last until the end of November 2009. According to current plans, the EGI.org will start its operations in 2010.

2 Example: The Spanish National Grid Infrastructure

Several European countries have launched, or are about to launch, their national grid initiatives. NGIs will have a key role within the future EGI, as they are the building blocks of the new sustainable grid infrastructure. Each NGI should be a recognised national body that serves as a single point of contact in the country. Responsibilities between NGIs and the EGI will be split to be federated and complementary. NGIs will be responsible for the activities at the national level. They should mobilise national funding and resources, ensure an operational national grid infrastructure, support user communities and contribute and adhere to international standards and policies into the national legislation of the country. The role of the EGI will be to harmonise the national initiatives at the European level and to create a uniform grid infrastructure to support science and research (Fig. 6.2).

The first step for the Spanish national grid initiative took place in early 2003. Approximately 20 Spanish research groups with interest in grid technologies, several of them participating in the DATAGRID and CROSSGRID projects, launched a networking activity called IRISGRID. The objective was to analyse, define and foster the consolidation of a scientific community of users, infrastructure providers, application and middleware developers with interest on grids. This new community started to promote the concept of a Spanish NGI for the Spanish Ministry of Science. In 2006 Vicente Hernandez was nominated by the Ministry as first official responsible of the Spanish NGI. The official financial support started through e-Science Network in 2007 and the funded NGI activities in 2008. A special

Fig. 6.2 From National grids to the future European grid

characteristic for the Spanish network is a creation of a strong collaborative and cooperative link with the Portuguese NGI through the IBERGrid agreement.

Spanish institutions participating in grid research and development projects form the core component of the Spanish NGI. There are approximately 20 resource provider centres with 2400 cores and 340 direct users involved in operations. The close cooperation with the Spanish Supercomputing Network (RES) comprises several Spanish research centres operating with a common supercomputing infrastructure. The close links between supercomputing and grids are one of the most important challenges of the Spanish Network for e-Science. It is forecasted that a wide collaboration in terms of middleware, applications and even resources will be developed in the near future.

Scenarios similar to the Spanish one can be found in many of the European countries. Early adopters of grid technology have initiated or participated in EU and other projects. Through the different projects production-level reliable services have evolved. The quality services have attracted the interest from different research communities. Users explore the potential of the new infrastructure and start to count on it for their research. With a growing and depending user community, coordination of funding, policies and operation becomes necessary on a national level and national coordinating projects and eventually NGIs are formed.

3 European Grid Infrastructure and Large-Scale Research

The European Strategy Forum on Research Infrastructures (ESFRI) [9] has 35 different research infrastructures on its roadmap. In the present update procedure (year 2007–2008) this number will grow even further. The newly proposed infrastructures only are in the order of 2 billion Euro of construction cost. Many of the proposed infrastructures are in a preparatory phase, planning for their construction. The need for ICT supportive functions is investigated. Common keywords are data management, computing and simulation, grids, authentication and authorisation, network capacity and software development. These needs are common for many of the projects, something also recognised by the European e-Infrastructure Reflection Group (e-IRG). The group emphasised in a recent report to the European Commission the need for horizontal ICT functions common to the ESFRI projects, horizontal functions often referred to as e-Infrastructure for e-Science.

To a certain extent the e-Infrastructure itself is entering the ESFRI roadmap, but so far in the specific form of high-performance computing facilities of world-leading class, an area previously left for the USA and Japan. But, with today's research challenges (for example, in climate research, biomedicine and fundamental sciences) the need for leading high-performance computing facilities has been recognised by the European Commission and a number of countries, leading to the EU project Partnership for Advanced Computing in Europe (PRACE) [7] project. More might come and for the update of the ESFRI list two new proposals for horizontal ICT infrastructures or e-Infrastructures have been proposed by Finland, "Infrastructure for Preservation of Unrevealed Scientific Data" (IPURE) and "European Software Services Network for Large-Scale research Facilities" (ESSN). Both these proposals are answers to a general need by large-scale research projects for timely and reliable e-Infrastructures.

Grids and grid technology are crucial for large-scale research. Many research communities are already dependent on grids and expect that today's production grids with DEISA, EGEE, BalticGrid [3], SEe-Grid [8] and many more will continue and develop with a long-term perspective becoming a persistent research infrastructure in the same way as the European research network (GEANT). Of the user communities, the high-energy physics community, with its Large Hadron

Fig. 6.3 The EGI logo

Collider (LHC) experiment, has for example based the whole simulation and analysis of experimental data on a world-wide network of computing and storage resources, an infrastructure operated by several grid projects, with EGEE and OSG (in the USA) being the largest. In the European landscape of different computing resources and computing needs, the proposed EGI Organization (EGI.org) can provide the coordination and operation of a common pan-European e-Infrastructure facilitating the researchers' access and sharing of all kinds of resources on the Internet.

References

1. e-Infrastructure Reflection Group (e-IRG). http://e-irg.eu
2. EGI_DS project vision paper. http://www.eu-egi.eu/vision.pdf
3. EU FP7 Project BalticGrid Second Phase (BalticGrid-II). http://www.balticgrid.org/
4. EU FP7 Project Distributed European Infrastructure for Supercomputing Applications (DEISA). http://www.deisa.eu/
5. EU FP7 Project Enabling Grids for E-sciencE (EGEE). http://www.eu-egee.org/
6. EU FP7 Project European Grid Initiative Design Study (EGI_DS). http://www.eu-egi.eu/
7. EU FP7 Project Partnership for Advanced Computing in Europe (PRACE). http://www.prace-project.eu/
8. EU FP7 Project South Eastern European Grid-enabled e-Infrastructure development (SEE-Grid). http://www.see-grid.org/
9. The European Strategy Forum on Research Infrastructures (ESFRI). http://cordis.europa.eu/esfri/

Chapter 7
Virtual Appliances: A Way to Provide Automatic Service Deployment

G. Kecskemeti, P. Kacsuk, T. Delaitre, and G. Terstyanszky

Abstract Manual deployment of an application usually requires expertise both about the underlying system and the application. To support on-demand service deployment or self-healing services this chapter describes an extension of the globus workspace service [12]. This extension includes creating virtual appliances for grid services, service deployment from a repository and influencing the service schedules by altering execution planning services, candidate set generators or information systems.

Keywords Grid · Web services · Deployment · Virtual appliance · OGSA

1 Introduction

In this chapter deployment is defined as a composite task. It starts with the *selection* of the site where further deployment steps take place; then, it follows with the *installation* of the application's code, which is *configured* to match its new environment later on. After the initial configuration of the application the *activation* step enables its usage. The next deployment steps are closely related to system maintenance, and they are initiated later in the application's life cycle. *Adaptation* is used when the application has to be reconfigured while it is running. Meanwhile other maintenance steps are always preceded by *deactivation*, which disables the usage of the application temporarily. The two deactivation-dependent steps are the *update* and the *decommission*. The update changes the application's code, which is usually followed by a configuration step before the application can be active again. Meanwhile

G. Kecskemeti (✉)
Laboratory of Parallel and Distributed Systems, MTA-SZTAKI
e-mail: kecskemeti@sztaki.hu

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_7,

the decommission removes all the unnecessary application code and removes the application permanently from the site.

Nowadays application deployment is done by system administrators, because researchers usually cannot deal with their applications (e.g. BLAST [1], CHARMm [14], GAMESS-UK [10]) complex deployment requirements. As a result in a system where the deployment tasks have to be done manually the researchers have to run their applications on those sites where they were installed by an administrator. Even if these researchers could have deployed these applications, site policies in production grid environments usually do not allow users to deploy new applications. Therefore, they should ask the site administrators to install and configure the applications they need.

If users have the source codes and they are able to compile these, they have binaries of the applications, which can be staged to the executor site eliminating the deployment. However, this solution has several drawbacks. First *the drawback for the user* is that the application has to be compiled for every site. Also the application usually cannot make network connections with the outside world – e.g. to a database. Finally, the submission has to prepare the executor environment for the application every time. The second drawback is the *increased network traffic* for the frequent full execution environment submissions. And further drawbacks appear on the *system administrator's side* where every new application is a possible danger to the system, because administrators have limited control over this code. Therefore, a single submission could affect other executions.

As grid moves towards service orientation, trusted applications are exposed as services on the sites, and the users can only execute these services. As a result grid sites disable the "regular" execution of applications; on the other hand, they provide service interfaces for their users. Users can access the deployed applications on the different sites through these interfaces. Installing a new application in a service-oriented grid consists of two main steps. First deploying the application, installing its dependencies, configuring it, preparing network connections, etc. Then, providing a service-oriented interface to access the application and deploy the interface on a grid service container.

This service interface could be either a generic or specific one. The generic interface is provided by a grid middleware (e.g. OGSA-BES implementations and JSDL repositories) or other generic application wrappers (e.g. AHE [5], GEMLCA [6], gFac [11]). The specific interface is specific to the application and written by either the application developers or the user community.

Manual deployment of the application usually requires expertise both about the underlying system and the application. In several cases it cannot be done without the assistance from the system administrators. There is a need for a solution that can perform deployments without any specific expertise. This solution should also act between the limits set up by the system administrators. These kind of solutions are called automatic service deployment systems and they can both deploy the application and activate the service interface for further uses. There are two major approaches for automatic deployment: service-oriented and

nonservice-oriented solutions. The service-oriented solutions operate at service container level. Therefore they are not capable to handle the background application's management tasks. The nonservice-oriented solutions manage the service container as a common software component on which the currently deployed software depends.

Automatic service deployment could be based on virtualisation. The new hardware architectures give more and more support for virtualisation. Therefore, the software also has to tackle this issue by automating deployment tasks of virtual appliances of grid services. This chapter defines virtual appliances (VA) as a single application and its service interfaces with all of their dependencies (including the supporting OS) in a virtual machine image. Grid researchers have already developed some solutions in this area, for example, the XenoServer plattform [15], the Workspace Service (WS) [12] for Globus Toolkit 4 [7].

The chapter is organised in the following way. Section 2 discusses related works, Section 3 introduces some of the most relevant issues which could be solved by the deployment architecture proposed in Section 4; then, Section 5 describes an advanced virtual appliance creation service, and Section 6 provides solutions to alter an execution schedule to include deployment. Finally, Section 7 concludes this work.

2 Related Works

WS [12] (workspace service) as a globus incubator project supports wide range of scenarios. These include virtual workspaces, virtual clusters and service deployment ranging from installing a large service stack like ATLAS to deploying a single WSRF service. For service deployment purposes the virtual machine (VM) image of the service should be available. The WS is designed to support several virtual machines – XEN [3], VMWare, VServer – to accomplish its task.

The XenoServer open platform [15] is an open distributed architecture based on the XEN virtualisation technique. It is aiming for global public computing. The platform provides services for server lookup, registry, distributed storage and a widely available virtualisation server.

VMPlants [13] project proposes an automated virtual machine configuration and creation service heavily dependent on software dependency graphs. This project stays within cluster boundaries.

These solutions are focusing on the deployment process itself and do not leverage their benefits on higher levels like automation. As an opposite the solution presented in this chapter is focusing on possible extensions on the current deployment systems. The proposed architecture integrates the openness of the XenoServer platform, the widespread virtual machine support of the Workspace Service and the DAG-based deployment solution presented by the VMPlants. This chapter also introduces higher-level services supporting the service life cycle on grid architectural level.

3 Issues of a Virtualisation-Based Deployment System

On-demand deployment and self-healing services are among the most important improvements automatic service deployment could add to the deployment process. With *on-demand* deployment the applications are transferred, installed and configured on the target site prior to their execution. The deployment could be initiated by a broker or an advance reservation system. As these systems aggregate the invocation requests, they have an overview on the level of demand. After the initiation step, the application's code is transferred either from a repository or from the location the requester specifies. Repositories can be the source of trust for the system administrators who usually do not trust arbitrary code. Therefore on-demand deployment uses either centralised or distributed repositories. Centralised repositories store all the required and dependent software packages in the same location with their configuration. In distributed repositories, packages are spread around the grid. Distributed repositories could store pre-packaged software components and they have references to their dependencies. Alternatively software components could be distributed around the grid using replication solutions.

The newly deployed software components might interfere with the already deployed ones. For example, the software installed might need outgoing network connections, therefore on every local hardware resource (e.g. disks, network, processing power) the service should have strict limitations according to the service level agreements. If malfunction occurs in a grid service and the source of malfunction can be identified the service can reconfigure or reinstall the affected components. Service faults can be recognised through advance fault prediction or active service calls. Advance fault prediction uses component monitors to determine their misbehaviour. Active service calls can return regular output messages or fault messages. If the service response is a fault message the service should identify the cause of the fault, which could be internal or external. In case of internal causes the component causing the fault should be repaired. If the service responds with a regular response message the response has to be validated against the semantics of the service and violations should be reported back to the service initiating the *self-healing* process.

The service deployment with virtualisation can support both the on-demand deployment and the self-healing services in a secure way even at the site level. Using virtualisation techniques, a software repository for on-demand deployment should hold the virtual appliances of the services. Requirements against the hosting environment should also be stored together with the virtual appliances because the virtual appliances are virtualisation technique (e.g. Xen, VMVare, VirtualPC) dependent. Virtual machines could also provide restrictive domains for the deployed software in them. Therefore the limitations – e.g. outgoing network bandwidth, IP address – declared in the service level agreements of the site can be enforced via the virtualisation software. In order to support self-healing, virtual appliances should be distributed in several packages – a base package, and delta packages. The base package is a minimal and roboust package on which the delta packages are built. It should contain the necessary components to configure, install further components

of the application before their execution and it should be capable to reinstall these components when malfunction arises. The delta packages should represent those software components which the self-healing service is able to repair. The delta packages should be distributed with their individual configuration details in order to support on-demand reconfiguration.

4 Automatic Service Deployment

This chapter proposes the extension of the deployment system of the virtual workspace service (WS) with the following two new services, which solve the issues identified by Section 3.

Automated virtual appliance creation service (AVS). This service supports deployment by creating virtual appliances of grid services. The virtual appliances should be stored in an appliance repository, for example in the application contents service [9] (ACS). The ACS provides a simple interface for managing application archives (AA), which holds both the application packages and their configuration. The WS should access the contents of the service by using the endpoint references provided by the ACS for each AA. The virtual appliances (or service images) should be minimised because the repository may hold a large number of virtual appliances and because smaller sized appliances could be delivered faster to the target site. As a result, a virtual appliance shrinker should optimise the appliance's archive, even if it is using different virtual machine image formats of the different virtualisation techniques.

Scheduler assistant service (SAS). The SAS helps to define the sites where the service can be installed. It should also support selection among these sites; the SAS should be used together with the OGSA [8] execution planning service (EPS), more precisely, its candidate set generators (CSG). CSG can define scheduling candidates for sites, which are capable of deploying a service in a reasonable time. EPS could resolve schedules with deployment in the following ways. First with *schedule-driven deployment* the EPS has to insert extra deployment jobs in the schedule, which install and configure the service on the specified site. The second way to resolve schedules is the *container-managed deployment* technique, where the EPS does not need to deal with the deployment tasks, because on an unknown service call the service container should deploy the service from the repository.

The new services are shown in operation using XEN in Fig. 7.1, which can be split into two phases: virtual appliance creation and service deployment phase.

Phase I – Virtual appliance creation. In step (1) the client asks the AVS to store the Domain M's VM image of grid site A with a grid service in it in an AA instance. In this step the client is an entity wishing to publish a grid service as a virtual appliance – e.g. grid service developer, or grid scheduler who has made schedules for a grid service already deployed on a site but not available in any repositories. Then in step (2) AVS generates a basic virtual appliance from the XEN [3] Domain M.

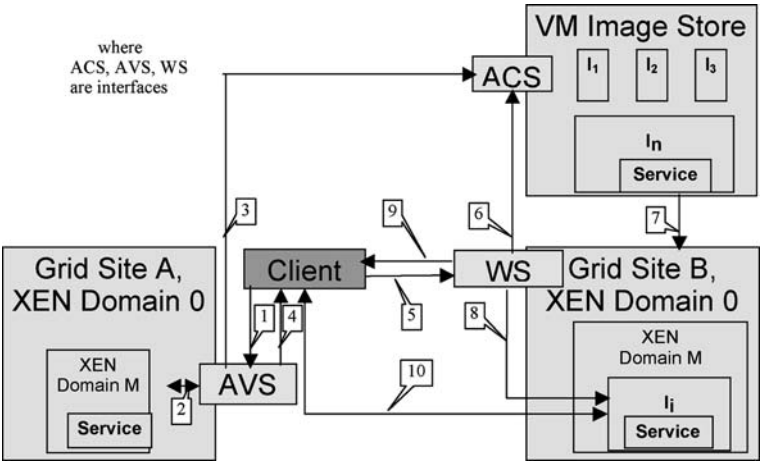


Fig. 7.1 Automatic service deployment

If its size can be decreased, the service shrinks (or optimises) the virtual appliance. This is followed by step (3) where the AVS stores the base appliance and the delta packages in an application archive (AA) instance, and adds the configuration and dependency information to support later deployment requests. Finally in step (4) the AVS acknowledges the virtual appliance (or image) creation by sending its ACS end point reference (EPR) to the client.

Phase II – Service deployment from a repository. This phase might not directly follow the virtual appliance creation phase, but it can be repeated as many times as required and it is always preceded by the single execution of phase I. Step (5) is the first step of this phase, where the client asks the virtual workspace service to deploy the VM image using the ACS EPR returned by the previous phase. In this step the client is an entity having an ACS EPR, e.g. a regular user initiating a deployment, a grid broker trying to load balance between sites. Then step (6) follows with the WSFactory request to the ACS in order to provide the application archive (AA) instance and its configuration details (including the configuration of the virtual workspace). As a result the VM image is transferred to the grid site B in step (7). Then, in step (8) the WS creates a virtual workspace according to the configuration using the VM image embedded in the AA instance. The endpoint of the created workspace is returned in step (9). Finally, in step (10) the client forwards any further service requests to the re-deployed grid service on Site B. If necessary it manages the VM instance through the previously received EPR (VM termination and configuration changes).

The automated virtual appliance creation service and the scheduler assistant service are described in Sections 4 and 5, respectively.

5 Automatic Virtual Appliance Creation Service (AVS)

The automatic virtual appliance creation service creates and stores virtual appliances in an ACS repository [9] to make the appliances available for WS. The service is built on the ACS–WS interface, which enables deployment in grids. The interface should enable WS to deploy virtual appliances retrieved as AA instances from an ACS repository. The AA instances store the virtual appliances and their states. The state of the virtual appliance is composed of WS resource properties and VM configuration parameters. This service has the following functionality:

Creating virtual appliances. The AVS service implements AVS–WS interface. The first implementation of this service uses the XEN virtual machine [3]. The service provides four operations. First, it generates an AA instance. Second, the service instructs the XEN VM to make a disk and memory image of the domain and store these images in the AA instance. Third, it collects a VM setup, such as XEN domain-specific setup, and if it exists, the network shaping setup for the Domain 0 VM, and converts these parameters into WS resource properties. Finally, it uploads the state of the VM image to the AA.

Optimising virtual appliances. To minimise the size of virtual appliances, they should be shrinked. Efficient image shrinking can be achieved by active fault injection, which is a flexible dependency detection algorithm. After the image shrinking process each WSRF (web services resource framework) service should be validated against the developer provided tests. The subsystem dependencies detected during the shrinking process should be stored for future use, for example, in a CDL [4] (Configuration Description Language) document or in a model-driven deployment descriptor.

Repackaging the appliance. The service builds deployment DAGs [13] to define deployments. These DAGs are built on top of configuration and installation nodes. Each DAG starts from a base virtual appliance – which is replicated over the grid sites. The base virtual appliance is a nonfunctional virtual appliance capable of holding further packages required for the service. The optimal selection of the base appliance is crucial, because the system has to make compromises between transfer time (deliver a complete VA from a remote location) and on site deployment time (construct the required VA from more accessible pieces on site).

Identifying base appliances. The AVS regularly analyses the AAs in the repository and checks the similarities of their deployment DAGs [13]. The similarities mean common roots in the DAGs. Figure 7.2 presents multi package (Px) deployment strategies for two services (Sx). The dashed packages are the similar deployment tasks in the two deployments. If the similar items exceed a system-level threshold, then the common nodes can be stored in a base virtual appliance (BVa). This base appliance is going to be spread over the grid by replication. Finally, the application archives used for building the base virtual appliance have to be revised. The archives should build on this newly created base appliance. The deployment of an archive built on a base appliance is composed of two phases: first the base appliance

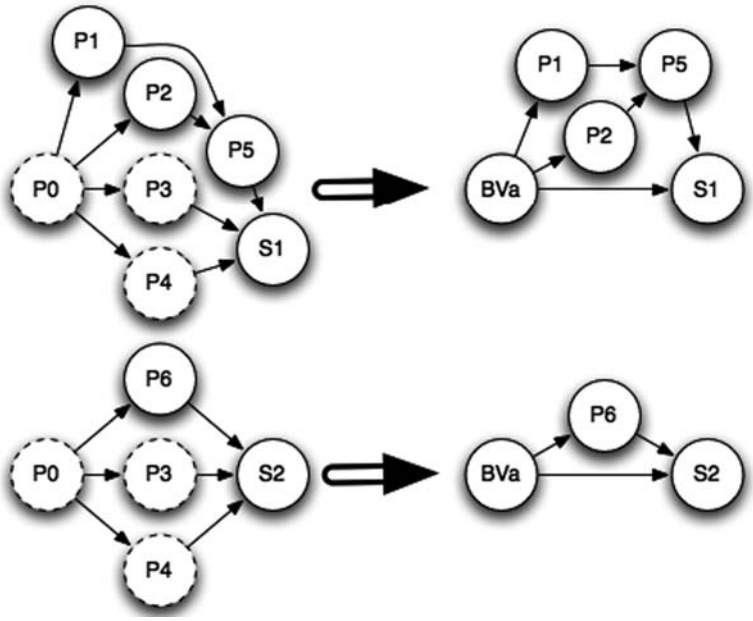


Fig. 7.2 Service deployment DAGs

is deployed, and then this base appliance is started in a self-healing state. During the self-healing process the BVa installs and configures the missing packages for healthy operation.

6 Scheduler Assistant Service (SAS)

This service is built on an ACS repository [9], which is usually prepared by the AVS. Its task is to define a schedule for executing service requests taking into consideration sites where service deployments are feasible. If the already deployed services are busy (or in a degraded state), then it checks whether any other grid sites can deliver the service with a new deployment.

OGSA-EPS [8] has two main connections with the outside world: the candidate set generators and the information services (IS). In order to influence the schedules the EPS makes, the assistant service could be installed on any of the following components or on their combination:

Candidate set generators. The scheduler assistant generates extra candidate sites for execution. These sites are the ones where the requested services have not been deployed.

Execution planning services. The scheduler assistant queries the CSG to retrieve the list of sites, which can handle the service request and which can deploy the service in a given time period. If no site can deliver the requested service, then the

EPS makes a decision upon the results of the second query and adds two separate entries to the schedule – the deployment task, and the service call.

Information services. The scheduler assistant generates virtual entries in the information services. Since both the CSG and the EPS heavily rely on the IS the assistant can include information which might alter their decision. This information states service presence on sites where the service is not even deployed. The QoS information stored in the virtual entries are degraded to represent the latency the deployment would cause before serving the first request. This solution has serious problems compared with the previous two ones. The IS has to be filled with the full service site matrix, which could increase query times and load on the IS nodes. Non-realistic information is introduced in the information systems; this might affect some systems.

Installation of the assistant service or services next to an OGSA-EPS enabled site depends on the grid policies. The assistant will be able to cooperate with an existing EPS as a CSG or an IS source, or it can offer a new, enhanced EPS on deployment enabled Grids.

The *CSG assistant* is a distributed, ontology-based adaptive classifier to define a set of resources on which a job can be executed. The CSG can build its classification rules using the specific attributes of the local IS. Each CSG may have a feedback about the performance of the schedule made upon its candidates in order to further improve its classification rules using a semi-supervised learning. The CSGs build a P2P network and the EPS's candidate nomination request might spread over the neighbouring CSGs for refinement – the request is sent when the quality of the current candidates is below a certain limit. When a new grid with a CSG is deployed it inherits the rules of the neighbouring classifiers at startup. A SAS extended CSG has three interfaces to interoperate with other CSGs and the EPS. The P2P network formed by CSGs has two interfaces. The first P2P interface manages the ontology of different ISs by sharing the classifier rules and the common ontology patterns distributed as an OWL schema. The second P2P interface supports distributing the decision-making process, and the resulting candidate set can be sent directly to the EPS without traversing through peers. The third interface lays between the EPS and the CSGs to support the supervised learning technique of the CSGs – the EPS reports the success rate of the received candidate set to the originator.

The *EPS assistant* has different implementations depending on how the other parts of the SAS are deployed. If both the CSG assistant and the EPS assistant are deployed, then the EPS can make smarter decisions. After receiving the candidate site-set the EPS estimates the deployment and usage costs of the given service per candidate. To estimate the deployment costs the EPS queries the WS with an ACS endpoint reference, which identifies a particular virtual appliance. The usage costs also include, for example, the cost of the inter-service communications – e.g. if a service is deployed closer to its dependencies then the costs decrease. Therefore, the EPS estimates the change in the communication cost between the affected endpoints (e.g. it checks for latencies and available bandwidth). The SAS has a plug-in-based architecture in order to support different agents discovering different aspects of the deployment. If the EPS assistant is deployed without the CSG assistant then the EPS

could generate deployment jobs in overloaded situations; these deployment jobs are simple service calls to the virtual workspace service with the proper ACS EPR.

The *IS assistant* provides information sources on sites which can accept service calls after deployment. The SAS calculates the necessary metrics specified by the GLUE schema [2] – like EstimatedResponseTime, WorstResponseTime – for each site in the grid according to the local service availability and publishes them in the ServiceData entry.

The SAS-enabled grid services build on the fact that the ACS EPR is available for the services. Deployments use this EPR to initiate the installation process on the selected site. Therefore, the most crucial point in the system is the place of the repository reference. The SAS can collect the reference from various locations. First it can collect from the standard service repository, the *UDDI*, which can hold metadata for each service; if this metadata is a valid ACS EPR the SAS uses it for deployments. Second the SAS can also collect the EPR from the *GLUE Schema*. The schema specifies service entries per site, and every service entity has a ServiceData map, which holds key value pairs for extended use of the schema's information. One of the ServiceData objects in the per service description should be the ACS EPR. Finally, the most commonly available service description in the web services world is the *WSDL*. It can be customised and further extended as described in the W3C standards. The WSDL can describe which virtual appliance was used for instantiating the service by including an ACS EPR element after the service element the WSDL. As a consequence, both the GLUE schema and the UDDI are supported without the extensions on them, because they both hold references to the WSDL the service conforms with.

7 Conclusion and Future Work

This chapter defined an infrastructure which an automated service deployment solution can build on. It has covered the life cycle of service and application deployment from the packaging to the execution. In the future the interface of the self-healing virtual appliance has to be further developed to support local and system-wide malfunction alerts and self-monitoring systems. Also the SAS should be extended towards seamless integration with other execution management subsystems in OGSA like CDDLM [4].

References

1. S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, May 1990.
2. S. Andreozzi, S. Burke, L. Field, S. Fisher, B. Konya, M. Mambelli, J.M. Schopf, M. Viljoen, and A. Wilson. *GLUE Schema Specification version 1.2*, 2005.
3. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

4. D. Bell, T. Kojo, P. Goldsack, S. Loughran, D. Milojicic, S. Schaefer, J. Tatemura, and P. Toft. *Configuration Description, Deployment, and Lifecycle Management (CDDLM) Foundation Document*, 2005.
5. P.V. Coveney, M.J. Harvey, and L. Pedesseau. Development and deployment of an application hosting environment for grid based computational science. In *Proceedings of UK e-Science All Hands Meeting 2005*, 2005.
6. T. Delaittre, T. Kiss, A. Goyeneche, G. Terstyanszky, S.Winter, and P. Kacsuk. GEMLCA: Running legacy code applications as grid services. *Journal of Grid Computing*, 3(1–2):75–90, Jun. 2005. ISSN: 1570–7873.
7. I. Foster. Globus Toolkit version 4: Software for service-oriented systems. In: *IFIP International Conference on Network and Parallel Computing*, 2005.
8. I. Fosterd, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich. *The Open Grid Services Architecture, Version 1.5*, 2006.
9. K. Fukui. *Application Contents Service Specification 1.0*, 2006.
10. M.F. Guest, I.J. Bush, H.J.J. van Dam, P. Sherwood, J.M.H. Thomas, J.H. van Lenthe, R.W.A. Havenith, and J. Kendrick. The GAMESS-UK electronic structure package: Algorithms, developments and applications. *Molecular Physics*, 103(6–8):719–747, Mar. 2005.
11. G. Kandaswamy, D. Gannon, L. Fang, Y. Huang, S. Shirasuna, and S. Marru. Building web services for scientific applications. *IBM Journal of Research and Development*, 50(2/3), Mar./May 2006.
12. K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron. Virtual workspaces in the grid. In *ANL/MCS-P1231-0205*, 2005.
13. I. Krsul, A. Ganguly, J. Zhang, J. Fortes, and R. Figueiredo. VMPlants: Providing and managing virtual machine execution environments for grid computing. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, Pittsburgh, PA, 2004.
14. A.D. MacKerel Jr., C.L. Brooks III, L. Nilsson, B. Roux, Y. Won, and M. Karplus. *CHARMM: The Energy Function and Its Parameterization with an Overview of the Program*, volume 1 of *The Encyclopedia of Computational Chemistry*, pp. 271–277. John Wiley & Sons: Chichester, 1998.
15. D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: Accountable Execution of Untrusted Programs. In *7th Workshop on Hot Topics in Operating Systems*, IEEE Computer Society Press, Rio Rico, AZ, 1999.

Chapter 8
Job Scheduling in Hierarchical Desktop Grids

Z. Farkas, A.Cs. Marosi, and P. Kacsuk

Abstract Desktop grid is a relatively new trend in grid computing. As opposed to traditional (service based) grid systems, desktop grids are based on volunteer computing: users can volunteer their computers' free CPU cycles to solve some kind of CPU-intensive problem. Creating a desktop grid project requires the installation of a single server and some enthusiast users to join the project by installing a simple client that downloads work from the server and uploads results after processing. MTA SZTAKI has created the hierarchical desktop grid concept, where not only single computers but also desktop grids can join another system increasing its performance significantly. In this chapter we describe scheduling issues that arise when considering hierarchical desktop grid systems and present some scheduling algorithms that can be used in such systems.

Keywords Desktop grid · Scheduling · Volunteer computing · Model

1 Introduction

The common architecture of desktop grids typically consists of one or more central servers and a large number of clients. The central server provides the applications and their input data. Clients (donors) join the desktop grid voluntarily, offering to download and run tasks of an application (binaries) with a set of input data. When the task has finished, the client uploads the results to the server where the application assembles the final output from the results returned by clients. Contrary to traditional grid systems where the maintainers of the grid infrastructure provide

Z. Farkas (✉)
MTA SZTAKI Computer and Automation Research Institute, 1518 Budapest, Hungary
e-mail: zfarkas@sztaki.hu

The research and development published in this chapter is partly supported by the Hungarian Government under grant NKFP2-00007/2005 and by the European Commission under contract numbers IST-2002-004265 (FP6 NoE, CoreGRID, www.coregrid.net) and LSHC-CT-2006-037559 (FP6 STREP, CancerGrid, www.cancergrid.eu).

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_8,

resources where users of the infrastructure can run their applications, desktop grids provide the applications and the users of the desktop grid provide the resources.

Thus a major advantage of desktop grids over traditional grid systems is that they are able to utilize non-dedicated machines. Besides, the requirements for providing resources to a desktop grid are very low compared to traditional grid systems using a complex middleware. This way a huge amount of resources can be gathered that were not available for traditional grid computing previously. Even though the barrier may be low for resource providers, deploying a desktop grid server is a more challenging task because a central server creates a central point of failure, while replicating servers require more efforts.

SZTAKI desktop grid is based on BOINC [3], but adds various enhancements for desktop grid computing, while the aim of SZDG remains volunteer computing. One of the enhancements is hierarchy [9], which allows a set of projects to be connected to form a directed acyclic graph and work is distributed among the edges of this graph. The hierarchy concept introduces two components.

First is the Hierarchy Client, which is based on the BOINC Core Client. It is always running beside any child project. Generally, a project acting as a parent does not have to be aware of the hierarchy, it only sees the child as one powerful client. The client reports currently the parent a pre-configured number of processors, thus allowing to download the desired number of workunits (and the belonging applications). There can be limitations set on the server side to maximize the allowed number of workunits downloaded per client, so the only requirement for the parent side is to set these limits sufficiently high.

The second component is a new security model for BOINC, since the current one is not adequate in a hierarchical setup. Any lower level project in a hierarchical desktop grid system must be able to automatically obtain the binary of an application from its parent and be able to offer the application to its clients without manual intervention, and this process must not increase the risk of injecting untrusted applications into the system. These requirements mean that a lower level project cannot simply re-sign the application it has obtained from the parent, since that would require the private key to be accessible on the machine hosting the lower level project, which in turn would significantly increase the risk of a key compromise if the machine hosting the project is compromised [12, 11].

An example hierarchical system can be seen in Fig. 8.1.

2 Related Work

In this section we focus on related work. On one hand, the hierarchical desktop grid concept is a relatively new concept, and as such, there has not really been any research that focuses on scheduling questions related to these kind of systems. On the other hand, the desktop grid concept is different from the traditional grid concept: in the latter case schedulers have to send a job with specified requirements to one of the services that most satisfies them, so we can think of them as a system that implements the push model when considering job execution. In the case of

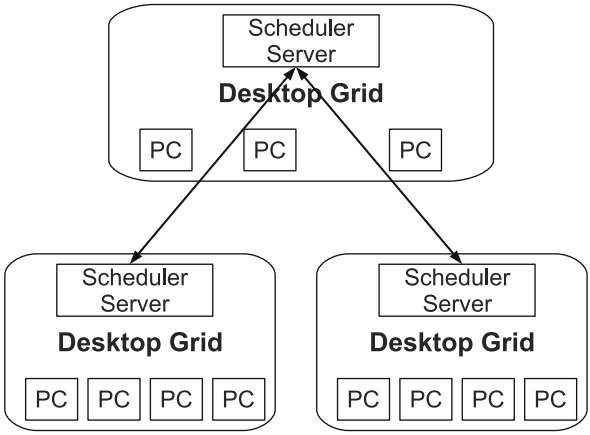


Fig. 8.1 Hierarchical system example

desktop grids, the situation changes: resources (donors) contact a central service (the desktop grid server) and fetch some work, thus implementing the pull model. As a consequence, the scheduling question changes: how much work should a donor fetch for processing? How does the one central server concept influence scheduling?

In [1] the authors described the task server component of BOINC and proved using measurements that a single machine can distribute as much as 8.8 million tasks a day, which is much more than the 100,000 jobs the EGEE infrastructure processes a day [7]. So we can say that the single central server concept does not introduce a bottleneck.

Regarding the donor side scheduling, [2] describes BOINC's local scheduling policies, [10] presents some algorithms and compares them with the help of simulation using different properties. In [5] the authors focus on scheduling techniques that improve the turnaround time of desktop grid applications; for this they use 39 days trace of computer availability of 32 machines in two classrooms and compare the results with the ideal execution time. The authors of [5] also have created a tool called DGSchedSim [6] that can be used to evaluate different scheduling algorithms on desktop grids using an existing trace.

There has been a notable amount of work regarding service grid scheduling. The authors of [8] present the grid scheduling problem: a set of heterogeneous resources, different jobs, constraints, and an objective. Fibich et al. [8] also presents a model for the scheduling problem. In [14] the authors present the TITAN scheduling architecture that uses performance prediction (performance analysis and characterization environment – PACE [13]) and focus on local scheduling. In [15] the authors show a cost-based online scheduling algorithm with a scheduling framework and analyze the performance of the presented algorithm theoretically, compared to the optimal offline algorithm. The two variants are compared using simulation, too. The work presented in [4] uses a formal framework based on Kalman filter theory to predict the CPU utilization in clusters. Using the presented predictions the authors measured a precision of 15–20%.

3 A Simple Model for Hierarchical Desktop Grids

In this section we present a simple model for hierarchical desktop grid systems. The model should cover scenarios, where

- a desktop grid fetches work from multiple desktop grids
- a donor fetches work from multiple desktop grids
- multiple desktop grids connect to one desktop grid in order to process its workunits.

A scenario that covers these cases is presented in Fig. 8.2. There is a desktop grid (D) that fetches work from multiple desktop grids (B and C), there is a donor ($donor_2$) that fetches work from multiple desktop grids (A and D), and multiple desktop grids (B and C) fetch work from a single desktop grid (A).

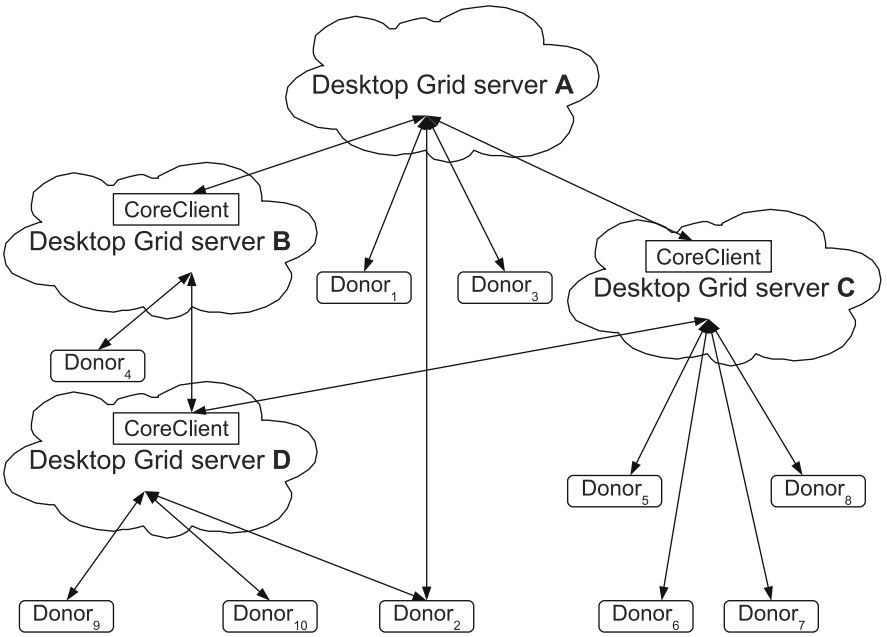


Fig. 8.2 Hierarchical system scenario

According to Fig. 8.2, a hierarchical desktop grid system can be represented as a directed graph. An important restriction of the graph is that it may not contain cycles, so infinite workunit transitions are eliminated. In the hierarchical desktop grid system we can identify the following entities:

- set of donors: $d = \{d_1, \ldots, d_n\}$,
- set of desktop grid servers: $S = \{S_1, \ldots, S_m\}$

Using the introduced notations, the set of vertexes of the graph is $V = d \cup S$. The set of arcs is $A = \{(x, y)|x, y \in V$ and y fetches work from $x\} \subset V \times V$.

With the help of these notations we can formalize the hierarchical desktop grid system shown in Fig. 8.2

$$
\begin{aligned}
d &= \{d_1, \ldots, d_{10}\} \\
S &= \{S_A, \ldots, S_D\} \\
V &= \{d_1, \ldots, d_{10}, S_A, \ldots, S_D\} \\
A &= \{(S_A, S_B), (S_A, S_C), (S_B, S_D), (S_C, S_D), \\
&\quad (S_A, d_1), (S_A, d_2), (S_A, d_3), (S_B, d_4), \\
&\quad (S_C, d_5), (S_C, d_6), (S_C, d_7), (S_C, d_8), \\
&\quad (S_D, d_2), (S_D, d_9), (S_D, d_{10})\}
\end{aligned}
$$

The model introduced so far can only be used to describe the shape of the system. It does not include properties like results to be processed by donors or workunits stored on the desktop grid servers. So, we add the following entities, too:

- workunit: $w = (pp, rl, pw, to, dm)$, where pp contains information about requested processing power needed to compute the workunit, rl represents the redundancy level, pw is a reference to the desktop grid from where the workunit has been downloaded, to is the time-out value (the maximum time a donor machine has to process the workunit), and dm is a reference to a donor processing the workunit. In order to keep things simple, we assume that $rl = 1$, so we do not use redundant computing. In the beginning pw is empty.
- set of workunits: $W = \{w_1, \ldots, w_k\}$.

Now we can finalize the model: each desktop grid may contain a set of workunits, so $S_i = (W_i)$, where W_i is the set of contained workunits. Similarly, each donor may contain a set of workunits, so $d_j = (W_j)$, where W_j is the set of workunits d_j has downloaded for processing.

4 Events Changing the Scheduling

In this section we present some events that change the status of the hierarchical desktop grid system and thus somehow influence the possible scheduling algorithms.

4.1 New Donor

This event occurs when a new donor (d_{n+1}) joins the system. Assuming it receives work from a given set of desktop grids (S'), the model changes the following way: $d = d \cup \{d_{n+1}\}$, $V = V \cup \{d_{n+1}\}$, and $A = A \cup_{x \in S'} \{(x, d_{n+1})\}$.

4.2 Exiting Donor

This event occurs when a donor (d_j) leaves the system. The consequence is that the donor is removed from the donor set ($d = d \setminus \{d_j\}$). If there are workunits that the donor has downloaded (assuming a set W'), for each workunit the workunit's *dm* value is cleared: $x \in W' : x = (x.pp, x.rl, x.pw, x.to, 0)$. Please note that the assigned time-out value is not cleared, otherwise the desktop grid containing the given workunit would assume that it is available for processing.

4.3 New Workunit

This event occurs when a new workunit is added to one of the desktop grids. The effect is that the W set flares with the new workunit, and the workunit is assigned to the relevant desktop grid.

4.4 Workunit Transition

This event occurs when a workunit (w) is downloaded from a desktop grid (s). The event contains two different cases: the workunit is transmitted to another desktop grid (s') or the workunit is transmitted to a donor (d) for processing.

In the first case a time-out value and the accepting desktop grid (as the processing donor) is assigned to the workunit (the relevant properties are *to* and *dm*). Afterward, a new w' workunit is created for s' that inherits the *pp* and *rl* properties of the w workunit, and *pw* points to s, so $w' = (w.pp, w.rl, s, 0, 0)$. In this case w is the parent and w' is the child workunit.

In the second case the workunit is assigned a time-out value and the processing donor is set to d, so $w = (w.pp, w.rl, w.pw, to, d)$, where *to* is the created time-out value.

4.5 Workunit Processed

In the beginning of this section we have assumed that the redundancy level is 1, so once a workunit (w) has been computed, the result is accepted to be a canonical result. In the case of this event, both the workunit and all its parent workunits must be removed from the system. This can be achieved by reporting finish on lower levels. So if a desktop grid that received a workunit from an upper level desktop grid receives a workunit completion event, it reports the same event toward the upper level and removes the workunit from its database. Using this algorithm all the desktop grids that were used during the original workunit's transmission are notified. Also, the procedure finishes in a finite number of steps because we have

forbidden using cycles in the system. The following algorithm can be used to do the procedure:

```
1: w = computed workunit
2: if w.pw != 0 then
3:     nw = w.pw
4:     report_computation(nw)
5: end if
6: clear(w);
```

In the above algorithm the *report_computation* function calls the algorithm recursively, but using the original workunit's parent.

4.6 New Desktop Grid

This event is very similar to the new donor event: the new desktop grid (S_{i+1}) can connect to already existing desktop grids (a set S') in order to process their workunits. First, we assume that the new desktop grid is empty, that is, it does not have any new donors or workunits. The effect of such an addition is similar to the case of new donor: $S = S \cup \{S_{i+1}\}$, $V = V \cup \{S_{i+1}\}$, and $A = A \cup_{x \in S'} \{(x, S_{i+1})\}$. The non-empty case can be simulated using the sequence of new empty desktop grid addition, new workunits addition, and new donor addition events.

4.7 Exiting Desktop Grid

The effects of such event can be multiple: in case a top-level desktop grid is exiting, the system may be decomposed into two independent systems. Image that in Fig. 8.2 the S_D does not fetch work from S_C and S_A leaves the system: two systems come into existence, a hierarchical system consisting of S_B and S_D and a single system consisting of one desktop grid, S_C. A more simple case is when a low-level desktop grid (S_D) leaves the system: the effect is that also donors connected only to this desktop grid have to leave.

We can formalize the above. If desktop grid S_j is leaving the system, it is removed from the vertexes of the graph ($V = V \setminus \{S_j\}$), any arc containing the component is also removed ($A = A \setminus \cup_{x \in V}\{(x, S_j),(S_j, x)\}$), and following this step if there are lonely donors in the graph, they are also removed ($\forall x \in d(\forall y \in S{:}(y, x) \notin A \Rightarrow d = d \setminus \{d_x\})$).

5 Scheduling Algorithms

In this section we present some scheduling algorithms that can be applied to hierarchical desktop grid systems. As we have mentioned in the introduction, hierarchical desktop grid systems can be built using the modified hierarchical Core Client

application developed by MTA SZTAKI. If we want to integrate scheduling algorithms with the Core Client application, we have two options:

Centralized method assumes that the scheduling algorithm is a global algorithm, that is, it knows everything about the hierarchical system. In this scenario the scheduling algorithm can be represented as a central service used by the different desktop grids in order to report their status, performance, or get the number of workunits to be fetched from a given higher level desktop grid.
Local method is used when the scheduler is "local" to the Core Clients, namely the algorithm is integrated into the Core Client. So, the only thing the algorithm knows is the status of the local desktop grid and the set of desktop grids the Core Client is connected to (without their status information).

We prefer to use the local option over the centralized one, because even if the first one provides a (probably) better overview of the system, it requires the installation and maintenance of a central service. Contrary to this, the local option means a simple extension of the already existing Core Client application, so it does not require any special system setup. Besides this, administrators of the different desktop grids are free to choose the scheduling policy they like.

The aim of the scheduling algorithms is to process the set of workunits present in the system the fastest way, so we would like to minimize the makespan. In order to achieve this the amount of work requested by the algorithms should reflect the desktop grid's performance, and the downloaded workunits' time-out should also be considered. It is trivial that if a desktop grid fetches less work than it can process, then this increases the total processing time of the whole system. On the other hand, if it fetches more work, some workunit results are reported after their deadline has passed, so some donors have worked needlessly.

Now we present different scheduling algorithms in four groups.

5.1 Static Scheduling Algorithms

The group of static scheduling algorithms contains one single entity, the default algorithm of the hierarchical desktop grid Core Client. This algorithm fetches as many workunits from the desktop grids as many the administrator has set (n), for example, 1,000. The fetched workunits are placed into the local desktop grid's database. Once a workunit has been processed, its result is uploaded to the desktop grid it came from and a new workunit is downloaded in its place.

There are some problems with this algorithm. First, the fixed number of requested workunits does not reflect the performance of the desktop grid by necessity; that is, the algorithm does not change its work fetch policy once a new donor connects to or leaves the desktop grid.

Imagine a scenario based on Fig. 8.2 where d_5 and d_6 first left the system, and afterward joined again in order to process workunits of S_B instead of S_C. If we

assume that S_C and S_B fetched 5 and 2 workunits, respectively, then after these events occur, donors connected to S_C will be overloaded whereas donors connected to S_B will starve. In case deadlines set for workunits received from S_A are very tight, 40% of workunits processed by donors connected to S_C will be processed uselessly.

Concluding the default algorithm, it is good in static systems (e.g., in the case of a fixed number of donor machines), but shows poor performance when the number of donor machines changes.

5.2 Algorithms Depending on Donor Number

As we have seen in the previous section, the biggest problem with the default algorithm comes forward when the number of donor machines changes. In this section, we present two algorithms that take into account the number of registered donor machines somehow during work fetching. The relevant information can be found in the `host` table of the local desktop grid's database. Some relevant information can be seen in Table 8.1.

Table 8.1 The `host` table of desktop grids

Field	Description
Number of rows	Number of donor machines
`userid`	Owner's identifier
`p_ncpus`	Number of CPUs
`p_fops`	Floating point performance
`p_iops`	Integer performance
`rpc_time`	Last connection to the server
`m_nbytes`	Memory size
`avg_turnaround`	Average result turnaround time

Donornum (DN) algorithm. A natural extension of the default algorithm is to request as many work units as many donor machines are registered in the desktop grid's database (according to Table 8.1, this is equal to the sum of `p_ncpus` fields in the `host` table). Hopefully, with this method the number of workunits fetched somehow reflects the performance of the local desktop grid. The donornum algorithm is triggered every time a donor machine connects to the server and behaves as follows:

1: nCPU = $\sum_{x \in \texttt{host}} (x.\texttt{p\_ncpus})$
2: nWU = number of workunits under processing and requested
3: **if** nWU $<$ nCPU **then**
4: request_work(nCPU-nWU)
5: **end if**

Once a new donor joins the desktop grid and connects to the server to fetch work, the algorithm collects the number of usable CPUs. Because the donor has registered

sooner then it has requested work, the donor's CPUs are already in the database, so the algorithm requests as many work units from the upper level desktop grid(s) as many CPUs the new donor has reported.

Although the algorithm is dynamic, it has several problems. First, it does not take into account the inactivity of a donor. Namely, if there are donors that have registered themselves, but after some time move on to another project, their registration remains in the database, so their CPU number is needlessly added to the *nCPU* variable defined in the algorithm. As a consequence, the requested number of workunits is more than the donors can process, so deadline violation might occur. Another problem is that the algorithm does not take into account the performance of the donor machines. Imagine a situation where a desktop grid with very slow donor machines fetches work from another desktop grid that assigns tight deadlines to the distributed workunits. Such situation also generates deadline overstepping problems.

Activedonor (AD) algorithm. As we have seen, the donornum algorithm has problem with inactive donors. There are two possible solutions to eliminate such donor machines:

- the desktop grid administrator should regularly remove donor machines from the database that have not reported any result for a long time,
- during the determination of available CPUs in the donornum algorithm, filtering should be used to bypass donors that have not reported any result for a long time.

The first solution requires human intervention, so we drop this option. A requirement toward the improved version of the donornum algorithm is that it should eliminate human involvement.

From now on the question is how to determine which donors are active and which are not. For this, we propose an algorithm that compares the set of downloaded workunits with the last activity time of the donor machines. In case a donor machine has connected to the server later than the maximum of the time-outs of the workunits, the given donor is filtered out. The algorithm that uses this filtering is called the activedonor algorithm and behaves as follows:

```
1:  nCPU = 0
2:  wuSET = set of downloaded workunits
3:  maxWUTimeOut = max {x.pw.to|x ∈ wuSET}
4:  for x ∈ host do
5:     if x.rpc_time < maxWUTimeOut then
6:        nCPU += x.p_ncpus
7:     end if
8:  end for
9:  nWU = wuSET.size()
10: if nWU < nCPU then
11:    request_work(nCPU-nWU)
12: end if
```

The above algorithm can be used to eliminate donors that have not contacted the server for a long time. The definition of the long time depends on the deadline assigned to the downloaded workunits from the higher level desktop grids. So the activedonor is an improvement over the donornum algorithm. However, there is still no guarantee that the donor machines will be able to process the downloaded workunits.

Activedonor performance (ADP) algorithm. The ADP algorithm is an improved version of the AD algorithm that tries to forecast the computation time of already downloaded workunits. If the computation time is higher than the deadline of any of the downloaded workunits, it does not fetch more workunits, but if the workunits can be processed within the deadline, it fetches more work.

There are multiple ways to do the prediction. A pessimistic way assumes that always the slowest donor gets the workunits for processing. Using this prediction we can make sure that the downloaded workunits will be processed within deadline. An optimistic way assumes that subsequent workunits are assigned to the fastest available donor. The algorithm behaves as follows:

```
1: nCPU = 0
2: aDSet = { }
3: wuSet = set of downloaded workunits
4: maxWUTimeOut = max {x.pw.to|x ∈ wuSet}
5: for x ∈ host do
6:    if x.rpc_time < maxWUTimeOut then
7:       nCPU += x.p_ncpus
8:       aDSet += x
9:    end if
10: end for
11: cTime = predict_cTime(aDSet, wuSet)
12: nWU = wuSet.size()
13: if nWU < nCPU && cTime < maxWUTimeOut then
14:    request_work(nCPU-nWU)
15: end if
```

In the algorithm `cTime` contains the predicted computation time that is determined using the `predict_cTime` function with the help of the set of downloaded workunits and active donors. The pessimistic approach uses the following implementation:

```
1: cTime = 0
2: lowestP = {min (x.p_fops + x.p_iops)|x ∈ aDSet}
3: for x ∈ wuSet do
4:    cTime += x.pp / lowestP
5: end for
6: return cTime
```

5.3 Algorithms Depending on Time-out

The algorithms in the previous group primarily used the number of donor machines registered to the desktop grid while deciding about how many workunits to fetch from upper levels. We have done some refinement to them in order to filter out inactive donors and to fetch possibly as many workunits as the active donors possibly are able to process. All these algorithms periodically have to query the desktop grid database in order to get the required information about the donor machines. In the case of a huge donor set, this operation can be expensive.

In order to eliminate this possibly expensive operation, we introduce a new algorithm group that operates through statistical information during workunit fetch count decision. All the proposed algorithms are connected to the received workunit's time-out (or deadline). Once the algorithms predict that fetching new workunits will cause to process the workunits beyond their deadlines, they stop downloading new workunits for a while.

During the model introduction we have shown that a workunit contains information like the requested processing power (*pp*), the "parent" desktop grid of the workunit (*pw*), and assigned time-out (*to*) and donor (*dm*) values. Algorithms belonging to this algorithm group operate using the *to*, the *dm*, and the *pw* attributes.

First, we introduce an algorithm that uses very basic statistical information; afterward, we show a possible refinement of the introduced algorithm.

Time-out (TO) algorithm. The time-out algorithm collects statistical information about the requested time to process workunits. For each workunit, it notices the time the workunit has been downloaded from the upper level desktop grid and the time the result of the workunit has been reported back. The time elapsed between the two moments is the total time the low-level desktop grid spent with processing the workunit from the upper level desktop grid's point of view and we call it *turnaround time (TT)*.

The idea behind the TO algorithm is to keep track of the average TT (*ATT*). Once a new workunit is about to be downloaded from an upper level desktop grid, its deadline is compared to the stored ATT value. If the deadline expires sooner than the desktop grid is assumed to finish it (that is, the deadline is smaller than the ATT), the workunit is not accepted, and the algorithm does not fetch new workunits for a while. The question is, how long the algorithm should wait to fetch new workunits? A trivial solution is to fetch new workunits when the ATT is smaller than the time of the last dropped workunit's deadline. However, if there are no workunits to process, there is no chance for the ATT to become smaller, so the algorithm has to reduce ATT periodically.

Summarizing the above, the algorithm can be implemented using the following code, which is called every time a workunit has been processed or every 10 min:

```
1: ATT = average_turnaround_time
2: pWUn = processed_wu_number
3: lfWUT = last_fetched_wu_time
```

```
4: if wu_Computed() then
5:     wuTT = computer_wu_turnaround_time
6:     ATT = ((ATT*pWUn)+wuTT)/(pWUn+1)
7:     pWUn += 1
8: end if
9: if ATT < lfWUT then
10:    request_work()
11: else
12:    ATT = ATT * 90%
13: end if
```

The above algorithm uses the function *wu_Computed()* to check if the algorithm has been called because a workunit has been processed. If yes, it updates the value of ATT. Next, it checks if the ATT is smaller than the last fetched workunit's deadline. If yes, it fetches a workunit, otherwise is reduces the value of ATT by 10%. Initially, the lfWUT variable becomes infinite, ATT and pWUn are 0.

Figure 8.3 shows how the algorithm periodically changes ATT as long as it is reduced below the last fetched workunit's deadline value (dashed line in the figure). Once the ATT decreases below the dashed limit, new workunits can be downloaded from the upper level desktop grid(s). Gray parts of ATT values show the problematic parts.

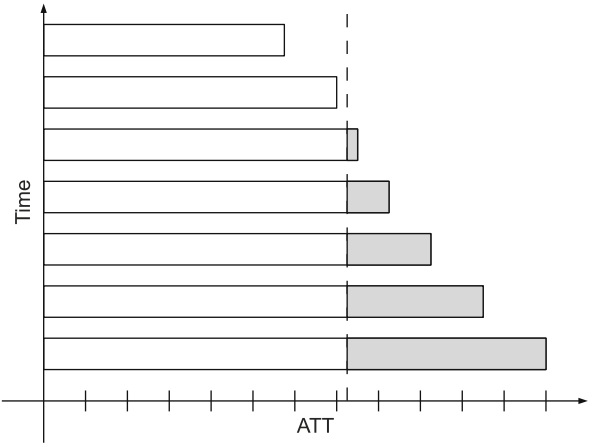


Fig. 8.3 ATT decreased by the TO algorithm

Donortime-out (DTO) algorithm. There is a problem with the ATT value defined during the description of the TO algorithm. If there are donors with very variable performance, the statistics stored in the ATT value might be very distorted. Suppose that we have two machines connected to a desktop grid that receives workunits from an upper level desktop grid. Furthermore, assume that all workunits require the same computation power are fetched with a 20-min deadline, and the first machine (A) finishes the computation in 10 min, but the other one (B) finishes it in 30 min. It is

trivial that every result belonging to workunits sent to *B* will be reported beyond the deadline. However, the TO algorithm will always fetch work also for *B* as the ATT value will be below 20 min. The first few values of *ATT* can be seen in Table 8.2. *WUA* and *WUB* contain the workunit identifiers machines A and B work on at the given time. Using the specified machine speeds, the total reported computation time at the *n*th minute (assume that *n* is a multiple of 10) is $TotalTime = n + \lfloor \frac{n}{30} \rfloor * 30$, and the number of workunits computed at the *n*th minute is $TotalWU = \frac{n}{10} + \lfloor \frac{n}{30} \rfloor$. The *ATT* values shown in Table 8.2 are the quotient of the assorted *TotalTime* and *TotalWU* values. The first few *ATT* values show that the TO algorithm fetches work for the slow computer even if it cannot report the results in time.

Table 8.2 ATT values and processed workunit IDs for machines A and B

Minutes	WUA	WUB	ATT
0	0	1	0
10	2	1	10
20	3	1	10
30	4	5	15
40	6	5	14
50	7	5	13,3
60		...	

In order to overcome this issue, a proposed improvement of the TO algorithm is the DTO algorithm that keeps track of the average turnaround times for each registered donor. So we can eliminate statistic distortions that arise from computers with very diverse performance.

The only modification of the algorithm shown in the case of the TO algorithm is to keep track of a vector of ATT values (indexed with donor identifiers) instead of a single ATT value, and in case a result has been reported, update the relevant donor's ATT value. Once a donor's ATT value grows beyond the downloaded workunit's deadline, no new work will be fetched for the given donor as long as the algorithm has not reduced the stored ATT value beyond the stored deadline.

5.4 Algorithms Considering Local Workunits

During the previous algorithm descriptions we assumed that the local desktop grid is empty, that is, it contains only workunits fetched from upper levels. The performance of these algorithms gets worse if there are local workunits, too. The desktop grid server might send the local workunits to connecting donors first, so the reporting time of fetched workunits might become longer, and the probability of reporting a result beyond deadline increase. In situations, where workunits received from upper level desktop grids have priority over local workunits, such problem does not arise. However, when this is not the case, hundreds or thousands of workunits might be processed before the fetched workunits are assigned to donor machines.

The default scheduling policy of BOINC is to send a workunit to a host only if it has enough memory and disk capacity, it is likely to complete the workunit by its deadline, and one machine can receive only one instance of a workunit[1]. As we have stated earlier, we assume that the redundancy level is 1, so this criterion can be skipped.

In order to make algorithms able to take local database state into consideration, we now show the relevant tables in the database. Workunit information is stored in the `workunit` table; relevant properties are described in Table 8.3. For every workunit, some entries are created in the `result` table. As we assumed earlier, the redundancy level is one, so for every workunit one entry is created in the `result` table. The table is described in 8.4.

Table 8.3 The `workunit` table of desktop grids

Field	Description
Number of rows	Number of workunits
`rsc_fpops_est`	Estimated required floating point operations
`rsc_fpops_bound`	Upper bound for floating point operations
`delay_bound`	Deadline
`priority`	Workunit priority

Table 8.4 The `result` table of desktop grids

Field	Description
`workunitd`	Relevant workunit identifier
`server_state`	How the server sees the result's state
`outcome`	State in the case of termination
`hostid`	Assigned donor identifier
`report_deadline`	Deadline of result reporting
`send_time`	Time the donor accepted the work
`received_time`	Processing result's reception time

Waiting queue (WQ) algorithm. This algorithm is a skeleton algorithm. That is, many algorithms can be produced using it as a skeleton. The idea is to call a function before fetching new workunits. The function's task is to predict the expected result computation time of the new workunits. In order to do so, the algorithm takes a look at the set of workunits already present in the database and the set of registered donor machines available for computations. This information can be used to predict the time when freshly fetched workunits' computation can be started.

The algorithm is activated every time a donor machine connects to the server to fetch new workunits or to report a result. First, it predicts when the last fetched workunit will be processed. If it is beyond the workunit's deadline, it does not fetch new workunits. If it is within the deadline, a new workunit is fetched that is added to the local desktop grid's database. The outline of the algorithm is as follows:

```
1: cTime = predict_computation(last_WU)
2: if cTime < last_WU.pw.to then
3:     request_work()
4: end if
```

The question is how the *predict_computation()* function should work. This function is responsible for predicting the possible computation time of the last fetched workunit, *last_WU*. We have multiple possibilities: pessimistic, optimistic, and close to BOINC approaches. A common point is that the prediction should consider workunit priorities, that is, workunits with higher priorities are computed first. With this property, it fetched workunits have priority over local workunits, the former ones are processed faster (and probably more workunits are fetched than in case fetched workunits have no priorities over local ones).

The pessimistic approach first sorts the workunits by their priorities (highest priority comes first), then by their creation time. Next, it begins to assign workunits to donor machines. It assumes that always the slowest donor gets the next workunit. The outline of the algorithm is the following:

```
1: WUs = set of workunits
2: donors = set of donors
3: sort_by_prio(WUs)
4: clear_donor_assignments(donors)
5: cTime = 0; actWU = WUs[0]
6: while actWU != selected_WU do
7:     sFD = slowest_free_donor(donors)
8:     assign(actWU, sFD)
9:     cTime += actWU.pp/sFD.performance
10:    actWU++
11: end while
12: return cTime
```

In the above algorithm, the *clear_donor_assignments* function makes every donor free, *cTime* contains the predicted computation time. The function *slowest_free_donor* selects the slowest available donor machine, and the function *assign* allocates the actual workunit (*actWU*) to the selected donor machine. Once the required workunit (*selected_WU*) is reached in the ordered vector *WUs*, the computed cTime is returned.

The optimistic version of the algorithm uses the abstract *fastest_free_donor* function to find the fastest available machine to compute the next workunit.

The close to BOINC approach should investigate the work distribution policy of current BOINC implementation and simulate its behavior for computing the requested *cTime* value of the selected workunit *selected_WU*.

It is important to note that every execution of the above described skeleton algorithm might use the desktop grid's database server heavily. In order to eliminate such problem, a memory cache can be used to store database information. The cache should be updated only if a new workunit has entered the database or another workunit's predicted termination time should be computed.

6 Conclusion and Future Work

The hierarchical desktop grid concept has been deployed in a seven-level hierarchical environment with clients attached to the lowest level. Six of the servers were running on Debian Linux 3.1/Intel, one was using Mac OS/X 10.4/PowerPC. Six clients were attached, three using Debian Linux 3.1/Intel, two Windows XP, and one Mac OS/X 10.4/PowerPC. A simple application has also been created for all platforms, with the only purpose to produce high load and run exactly for the time given in the workunit. The goal with this diverse environment was not performance analysis, but simply to test the environment for possible problems and bottlenecks. Using the prototype we were able to provide basic hierarchical functionality without modifying existing projects. Only the modified Core Client was needed for work request and distribution. After installation, a new project can be created with a single command, and then only the application executables should be set into operation on that machine. Client machines need just the standard BOINC Core Client installed. The setup and operation of the SZTAKI local desktop grid infrastructure is so simple that this is a strong reason in itself to use such an infrastructure instead of a complex grid middleware, provided it fits our applications' needs.

In this chapter we have presented a simple model for hierarchical desktop grid systems and some possible scheduling algorithms related to such systems. The presented model is enough to provide all requested information for the algorithms. A common point of these algorithms is that they are local algorithms, namely they only know about the desktop grid they are attached to. The task of these algorithms is to fetch an optimal number of workunits, so deadline violation or client starvation does not occur. We have presented four algorithm groups: static algorithms, algorithms depending on the number of clients, algorithms considering time-outs, and algorithms considering local workunits. Table 8.5 summarizes the presented algorithms.

Table 8.5 A summary of presented algorithms

Algorithm type	Algorithms	Algorithm features
Static	Default	Fetches fixed number of workunits
Donor number	DN	Fetch as many workunits as many donors exist
	AD	DN + filter out inactive donors
	ADP	AD + take donor performance into consideration
Timeout	TO	Computes ATT
	DTO	Computes ATT for each donor
Local WUs	WQ (pessimistic)	Assume workunits are assigned to slow donors
	WQ (optimistic)	Assume workunits are assigned to fast donors
	WQ (BOINC)	Try to simulate BOINC's work distribution

SZTAKI desktop grid is now operating since 3 years. A prototype of the hierarchical desktop grid system has been created, and right now it is applied to climate modeling tasks. Future work includes evaluating the above algorithms in this climate modeling environment. Moreover, different test scenarios should be created to run the algorithms, including both empty and full lower level desktop grids, different priority policies, and even different algorithms used on different desktop grids in the systems. Using the results of the simulation we can offer some of the best-performing algorithms to users of the hierarchical desktop grid system and the climate modeling problem.

References

1. D.P. Anderson, E. Korpela, and R. Walton. High-performance task distribution for volunteer computing. In *First IEEE International Conference on e-Science and Grid Technologies*, Melbourne, 5–8 Dec. 2005.
2. D.P. Anderson and J. McLeod VII. Local scheduling for volunteer computing. In: *Workshop on Large-Scale, Volatile Desktop Grids (PCGrid 2007) Held in Conjunction with the IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Long Beach, Mar. 30, 2007.
3. BOINC webpage. `http://boinc.berkeley.edu/`
4. C. Chapman, M. Musolesi, W. Emmerich, and C. Mascolo. Predictive resource scheduling in computational grids. In *Parallel and Distributed Processing Symposium IPDPS 2007*. IEEE International, 26–30 Mar. 2007.
5. P. Domingues, A. Andrzejak, and L. Silva. Scheduling for Fast Turnaround Time on Institutional Desktop grid. Technical Report, Institute on System Architecture, CoreGRID – Network of Excellence, January 2006.
6. P. Domingues, P. Marques, and L. Silva. DGSchedSim: A trace-driven simulator to evaluate scheduling algorithms for desktop grid environments. In *PDP '06: Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06)*, 2006.
7. EGEE in numbers. `http://egee-na2.web.cern.ch/egee-NA2/numbers.html`
8. P. Fibich, L. Matyska, and H. Rudová. Model of grid scheduling problem. *Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing*, pp. 17–24, 2005. ISBN 1-57735-239-4.
9. P. Kacsuk, N. Podhorszki, and T. Kiss. Scalable desktop grid systems. In *Lecture Notes in Computer Science, High Performance Computing for Computational Science – VECPAR 2006*, vol. 4395, pp. 27–38, 2007.
10. D. Kondo, D.P. Anderson, and J. McLeod VII. Performance evaluation of scheduling policies for volunteer computing. In *3rd IEEE International Conference on e-Science and Grid Computing*, Bangalore, India, December 10–13 2007.
11. A.C. Marosi, G. Gombas, and Z. Balaton. Secure application deployment in the hierarchical local desktop grid. In P. Kacsuk, T. Fahringer, and Zs. Nemeth, editors, *Distributed and Parallel Systems – Cluster and Grid Computing (Proceedings of the 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS))*, pp. 145–154. Springer, 2007.
12. A.C. Marosi, G. Gombas, Z. Balaton, P. Kacsuk, and T. Kiss. SZTAKI desktop grid: Building a scalable, secure platform for deskop grid computing. *CoreGRID Technical Report*, 2007.
13. G.R. Nudd, D.J. Kerbyson, E. Papaefstathiou, S.C. Perry, J.S. Harper, and D.V. Wilcox. Pace – A toolset for the performance prediction of parallel and distributed systems. *International Journal of High Performance Computing Applications*, 14(3), Aug. 2000.

14. D.P. Spooner, S.A. Jarvis, J. Cao, S. Saini, and G.R. Nudd. Local grid scheduling techniques using performance prediction. In *Computers and Digital Techniques IEE Proceedings*, vol. 150, pp. 87–96, Mar. 2003.
15. C. Weng, M. Li, and X. Lu. An online scheduling algorithm for assigning jobs in the computational grid. *IEICE – Transactions on Information and Systems*, E89-D(2):597–604, Feb. 2006.

Chapter 9
Toward QoS Provision for Virtualized Resources in Grids

F. Rodríguez-Haro, F. Freitag, and L. Navarro

Abstract Virtualization solutions have recently gained considerable attention for supporting the consolidation of application servers into one physical machine. In a vanilla Xen implementation the scheduler shares equally all of the available physical CPU resources among the contending VMs. However, when the application that runs in the virtual machine space changes dynamically its resource requirements, a different solution is needed. Furthermore, if the resource usage is associated with service-level agreements, a predefined equal share of the processor power is not sufficient for the VMs. In this chapter, we present an approach to manage the QoS of virtualized resources in Grids. Our solution adjusts the resources needed by each VM according to an agreed QoS. We achieve this by a local resource manager (LRM), which we implemented as a prototype and deployed on Xen-virtualized machines. By means of experiments we show that the implemented management component can meet the service-level objectives (SLOs) by dynamically adjusting virtualized resources according to demand.

Keywords QoS · Dynamic resource management · Virtualization · Service-level objective (SLO)

1 Introduction

Recently, advances in hardware such as multi-core machines and advances in software such as the implementation of virtualization solutions [1] have redefined the shape of resource provisioning in grid computing [3], high performance computing (HPC) [5], and cluster computing. Commonly, resource providers (RP) are the last

F. Rodríguez-Haro (✉)
Computer Architecture Department, Polytechnic University of Catalonia, Jordi Girona, 08034 Barcelona, Spain
e-mail: frodrigu@ac.upc.edu

This work is supported in part by the European Union under Contract SORMA EU IST-FP6-034286 and CATNETS EU IST-FP6-003769, partially supported by the Spanish MEC project P2PGrid TIN2007-68050-C03-01, and the Mexican program PROMEP.

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_9,

element in the chain of these multilayer architectures. Therefore, resource management is a critical must-have component in which all of the upper layers rely on to provide guarantees on the use of the physical machine resources.

External components, such as global schedulers, meta-schedulers, brokers, among others, negotiate and cooperate to construct high-level plans to distribute the execution of multiple user applications. For this purpose, some knowledge needs to be known about the state of the resource providers (i.e., the final site where the execution will be done). In order to construct a high-level plan, upper level components base their knowledge mainly on two approaches. First by forecasting the near future performance of resource providers, and the second by assuming and trusting that local resource managers can meet both strict and loose CPU guarantees. The work presented in this chapter is centered in the latter.

The resource provisioning and management is a known open issue in the list of challenges that need to be solved. Traditionally, without virtualization, this problem is tackled using a pool of physical nodes that are tied to a specific operating system. This scenario leads to high infrastructure and management costs, but more importantly, the infrastructure is not efficiently used due to the waste of resources in the under-utilized physical nodes.

Taking into account that different applications have different resource requirements (for instance, a web service application, a read-intensive database application, or a math-intensive application), an application might not need the total of the resources provided in the physical node and under-utilizes some of the physical resources (like CPU or network interfaces). Unlike traditional approaches, virtualization makes it possible to boot different isolated operating systems sharing the multiplexed physical resources of a node. The isolation is achieved through the concept of the virtual machine (VM), which is a virtual extension of the physical machine's resources. In this chapter we present an approach to provide QoS management in VM-based resource providers. Two contributions are made. First, a management component which is implemented as a prototype of a local resource manager, and second an algorithm which implements a policy that meets the agreed application-level Quality of Service (QoS) for VM creation and application execution. This chapter is organized as follows: In Section 2 we describe the work related to our approach. In Section 3 we explain our approach, the management component, the prototype, and the QoS interface. In Section 4 we report an experiment with the implemented component. Section 5 concludes and provides an outlook on future work.

2 Related work

Ruth et al. [8] proposes the concept of *virtual distributed environments* (VDE) as a new sharing paradigm for multi-domain-shared infrastructure. The authors also propose to support *autonomic adaptation* of virtual distributed environments, driven by both dynamic availability of infrastructure resources and dynamic application resource demand. Our work is in this direction and aims to address intra LRM fine-grain adaptation mechanisms based on application-level objectives.

Padala et al. [6] propose an adaptive control system based on classic control theory. The purpose of the controller is to alleviate the poor utilization in traditional data centers. The experimental setup includes virtualization and multi-tier applications. The results obtained show that the controller approach meets application-level QoS goals and the data center achieves high utilization. However, the controller used only static models and some interesting experiments could not be performed due to some anomalies that the authors found in the Xen *simple Earliest Deadline First* (sEDF) scheduler.

The need of QoS in Grid environments has been studied in previous works [2, 4]. Colling et al. [2] proposed a framework for supporting QoS guarantees via both reservation and discovery of best-effort services; they also discussed that applications with specific QoS requirements can benefit from mechanisms for the support of both strict and loose guarantees. Menascé and Casalicchio [4] also stated that planning the capacity to guarantee quality of service in such environments is a challenge because global service-level agreements (SLAs) depend on local SLAs. However, a new challenge is presented to the Grid research community, that is, how to manage and assure local SLAs with strict and loose guarantees in the context of VM-based resource providers.

In [7] we explored the fine-grain assignment of virtualized resources for Grid environments. However, further research is needed to offer automatic online provisioning based on dynamic parameters. With this work we aim to solve issues regarding adaptation of QoS in the context of virtualized resources.

3 QoS Management of VM-based Resource Providers

We propose a Local Resource Manager (LRM) for machines using Xen for the VM-based resource provisioning. The component for the management of VM-based resource providers has been implemented in the Python language and is evaluated in a multi-core machine with the Xen virtualization solution and credit scheduler.

3.1 Approach

The fine-grained resource management enhances the multiplexing mechanisms offered by Virtual Machine Monitors (VMMs). At the same time, it allows that LRMs can manage the pinning mechanisms of virtualization technologies according to user requirements.

Our management component runs in the first VM (known as privileged domain or domain0). It carries out the usage accounting of every single resource and groups single physical resources into subsets. These subsets which can be dynamically changed are seen by the LRM's upper level as pseudo-physical machines with minimal interfering I/O interruptions between each other. Thus, physical resources with different characteristics can be exploited by a LRM. The subsets of resources

can be assigned to VMs with different application profiles (e.g., CPU intensive, net intensive, SMPs requirement, or I/O disk intensive).

3.2 Management Component and Prototype

The LRM aims to provide Grid middleware with the infrastructure for accessing a machine's subset of virtualized resources in a transparent way. It exposes a simple way to use standard service interface for running jobs and hides the details of the managed local resources. The implementation of the proposal architecture leverages technologies that are currently in continuous development. Those technologies are virtualization based on Xen with access by XML–RPC services.

We have developed a prototype of the LRM for the QoS management of virtualized resources, which includes the components shown in Fig. 9.1. It can be seen that remote clients request VM resources using the XML–RPC service interface in which the QoS requirements are expressed. The QoS provision is carried out by the monitoring component and CPU capacity management component. The LRM encapsulates components that take advantage of low-level virtualization primitives of Xen.

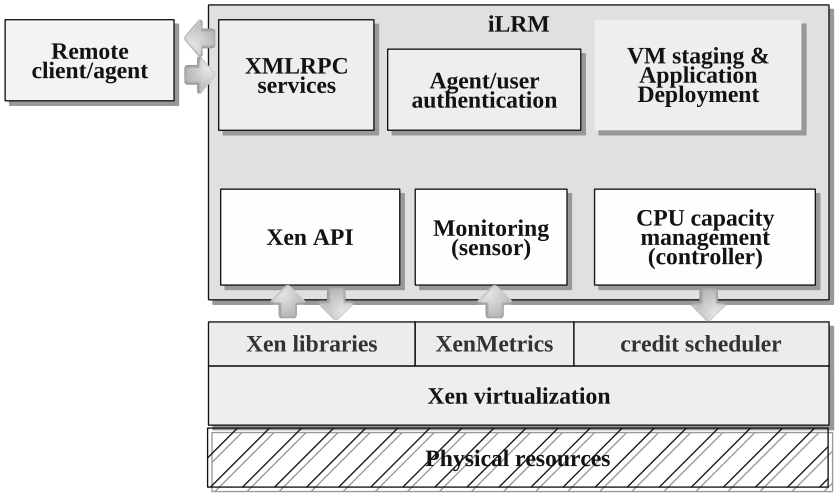


Fig. 9.1 Management component architecture

The Local Resource Manager (LRM) offers public services via XML–RPC interfaces, which can be seen in Fig. 9.2 in the format of a menu. Listing 9.1 shows the exchanged messages to retrieve the QoS features offered by the LRM. The design of the LRM includes other components that interface directly with Xen to perform tasks such as monitoring and CPU capacity management. The first is necessary to trace VMs performance in order to compute costs, and the second

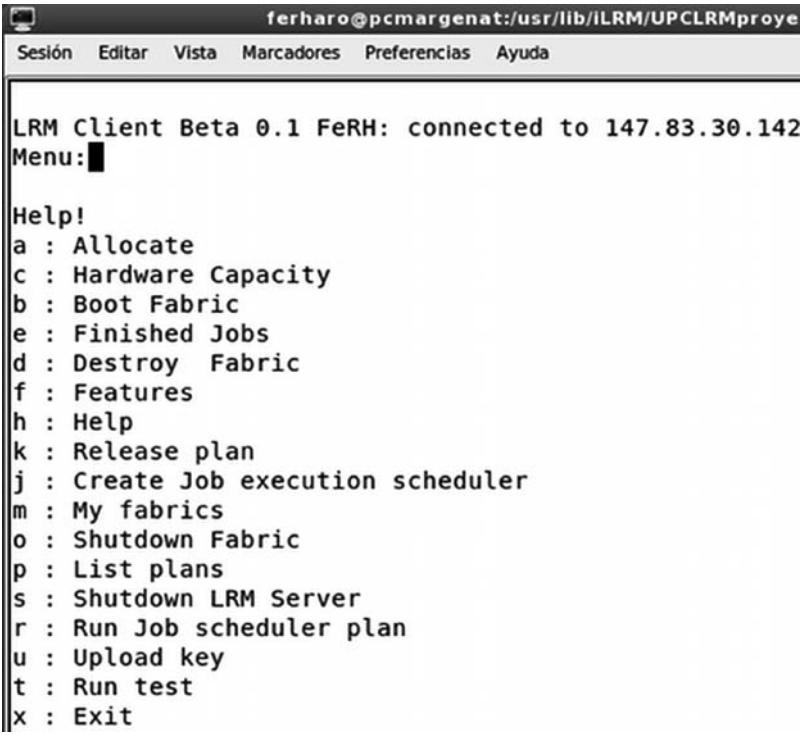


Fig. 9.2 Menu of LRM client interface

achieves offering fine-grain CPU specifications for VMs in multi-core processor machines.

Figure 9.3 shows a screenshot of the LRM client interface, as an example for partitioning the capacity of the resource provider. It can be observed that the interface shows the capacity (share) mapped from the MHz requested.

3.3 QoS Management

The VM creation request includes a job description requirement, sent by the clients, which contains the high-level input parameter values. With this information, the LRM accepts or rejects the VM creation. A process maps these values to low-level objectives in order to slice properly the resources managed by the virtualization layer. The information received from the client requests is as follows:

- An application-level requirement (*QoS*) expressed in MHz.
- A differentiated service type which can be *fixed*, *variableLow*, *variableMedium*, or *variableHigh*. This parameter aims to spread the unused resources between the contending VMs that run over their QoS requirements.
- A SLO requirement expressed in transactions per second (*tps*).

Listing 9.1 Exchanged messages for the service current_available_QoS

1) Client call

```
<?xml version='1.0'?>
<methodCall>
    <methodName>currentAvailableQoS
</methodName>
<params>
</params>
</methodCall>
```

2) LRM response

```
<?xml version='1.0'?>
<methodResponse>
    <params>
       <param>
          <value>
             <struct>
                <member>
                   <name>allocable_VMs</name>
                    <value>
                      <array>
                          <data>
                              <value>
                                 <string>pc4</string>
                                </value>
                                <value>
                                  <string>pc5</string>
                                </value>
                                <value>
                                  <string>pc6</string>
                                 </value>
                                 <value>
                                   <string>pc7</string>
                                 </value>
                                 <value>
                                   <string>pc8</string>
                                 </value>
                                 <value>
                                   <string>pc9</string>
                                 </value>
                               </data>
                          </array>
                      </value>
                </member>
                <member>
                  <name>Processors</name>
                    <value>
                       <array>
                           <data>
                              <value>
                                <double>2992.662</double>
                                </value>
                                <value>
                                  <double>2992.662</double>
                                </value>
                              </data>
                          </array>
                      </value>
                </member>
                <member>
                    <name>MaxMHz</name>
                    <value>
                      <int>1040</int>
                    </value>
                </member>
                <member>
                    <name>OStype</name>
                    <value>
                       <array>
                          <data>
                              <value>
                                 <string>linux</string>
                                </value>
                              </data>
                          </array>
                    </value>
                </member>
                <member>
                    <name>Price_GHz_Sec</name>
                    <value>
                        <double>1.0</double>
                    </value>
                </member>
                <member>
                <name>QoStype</name>
                    <value>
                        <array>
                            <data>
                               <value>
                                <string>fixed</string>
                                </value>
                                <value>
                                 <string>variableLow</string>
                                </value>
                                <value>
                                 <string>variableMedium</string>
                                </value>
                                <value>
                                 <string>variableHigh</string>
                                </value>
                              </data>
                        </array>
                    </value>
                </member>
             </struct>
          </value>
       </param>
    </params>
</methodResponse>
```

A learning procedure, which is part of the QoS Management component, traces the utilization of each VM and also traces the transactions executed by the application. This information is used to compute the current and the historic (learned) CPU demand imposed by each transaction. Finally, the controller computes an approximated *QoS* that aims to reach and meet the SLO requirement. The periodicity of this action, which can be configured to last a given number of periods, is called learning round.

4 Experiments and Results

We evaluated our deployed prototype of the LRM experimentally. The hardware of the resource provider node has a Pentium D 3.00 GHz (two cores), 2 Gb of RAM,

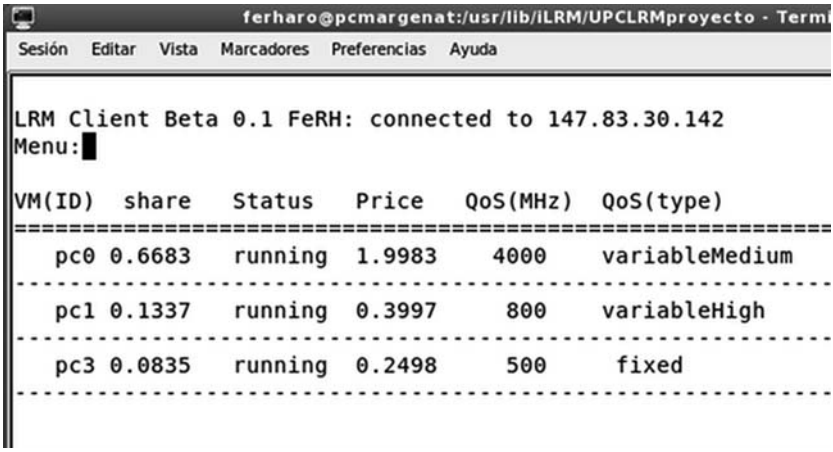


Fig. 9.3 Status information via LRM remote client interface

one hard disk (160 GB), and one network interface card (connected to a 100 Mbp switch). The operating system is Fedora Core 8 and for virtualization we use Xen 3.1.0. We deploy paravirtualized VMs with Fedora Core 7 for the guest OS. In the experiment we request the creation of two VMs with the controller component enabled.

The application-level requirements are *QoS* of 500 MHz, 250 MHz for pc0 and pc1, respectively, and SLOs of 20 and 60 tps, respectively. In order to stress each VM we run a custom application that behaves as a CPU intensive job. The fixed differentiated service makes a hard reservation for the VMs and the unassigned resources are given to the privileged domain.

Figure 9.4 shows the QoS adjustment made by the management component. The first graph shows the application's tps *TPSapp* read by the controller algorithm. The tps are passed through Xen's low-level communication mechanism called xenStore. The math application writes the measured *TPS* through xenStore, and Domain-0 collects this metrics to learn the VM's service demand of the running application. The second graph in the figure shows the current ($pc0_{curr}$, $pc1_{curr}$) and learned ($pc0_{learned}$, $pc1_{learned}$) service demand *SD*. The *SD* of each VM is computed with the tps (*TPSapp*) of the application running in the VM and the percentage of the VM's utilization, i.e., the *SD* is the amount of the CPU resource used for the execution of one transaction.

At the end of each round, which in this experiment lasts 60 s, the learned *SD* is used to compute the new *QoS* to reach the target tps. It can be seen that $pc0_{tps}$ and $pc1_{tps}$ reach and meet the target SLO requirements of 60 and 20 tps.

Figure 9.5 shows the load introduced by this LRM during online adjustment, which is quite low. The utilization of the privileged domain includes LRM tasks such as monitoring and usage accounting.

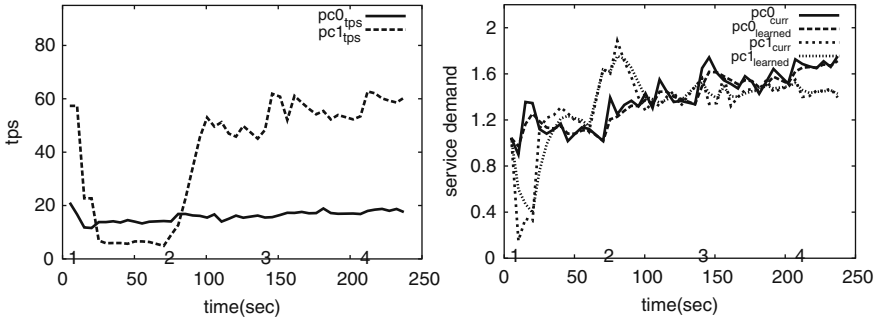


Fig. 9.4 The *first* figure shows the measured transactions per second (tps). The *second* figure shows the measured ($pc0_{curr}$, $pc1_{curr}$) and the calculated ($pc0_{learned}$, $pc1_{learned}$) service demand. The management component adjusts the CPU resources (*QoS*) to reach the SLO of 20 and 60 tps

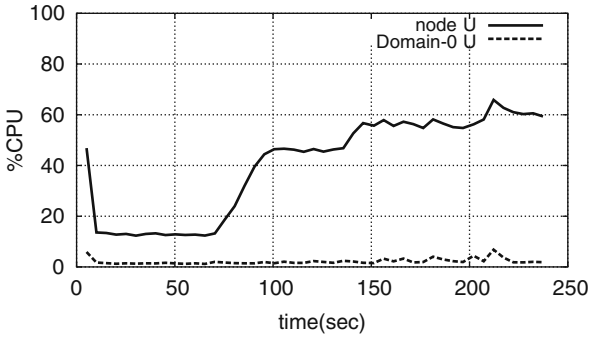


Fig. 9.5 Utilization: privileged domain vs. physical node

5 Conclusions

We presented an approach to provide QoS-based adjustment of CPU resources in multi-core VM-based resource providers for Grids. Our approach consists of a management component (LRM) which runs in the Xen's privileged domain of each virtualized resource provider. The LRM adjusts the proportion of the VMs according to the externally requested QoS. The fulfillment of the QoS is monitored and adjusted during the application execution. The LRM was developed as a prototype and deployed in a multi-core VM-based resource provider. By means of an experiment we showed that we achieved QoS goals by assuring at least the requested QoS (expressed in MHz) for each VM in the resource provider in a few learning rounds.

Currently, the differentiated service is explicitly requested by the user and remains static. However, the differentiated service value could be read dynamically from the XenStore database for adapting to changing QoS requirements. This functionality could represent scenarios where the budget that a user is willing to pay changes or where, according to metrics obtained from the running application, new resource requirements on the VM are detected.

References

1. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In: *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 164–177, ACM Press, New York, USA, 2003.
2. D. Colling, T. Ferrari, Y. Hassoun, C. Huang, C. Kotsokalis, S. McGough, Y. Patel, E. Ronchieri, and P. Tsanakas. On quality of service support for grid computing. In: *2nd International Workshop on Distributed Cooperative Laboratories and Instrumenting the GRID (INGRID 2007)*, 2007.
3. R.J. Figueiredo, P.A. Dinda, and J.A.B. Fortes. A case for grid computing on virtual machines. In: *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, p. 550, IEEE Computer Society, Washington, DC, USA, 2003.
4. D.A. Menascé and E. Casalicchio. Qos in grid computing. *IEEE Internet Computing*, 8: 485–87, 2004.
5. M.F. Mergen, V. Uhlig, O. Krieger, and J. Xenidis. Virtualization for high-performance computing. *SIGOPS Oper. Syst. Rev.*, 40:28–11, 2006.
6. P. Padala, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. Adaptive control of virtualized resources in utility computing environments. In *EuroSys2007*, 2007.
7. F. Rodríguez-Haro, F. Freitag, L. Navarro, and R. Brunner. Exploring the behaviour of fine-grain management for virtual resource provisioning. *Parallel Processing and Applied Mathematics*, LNCS 4967:961–970, 2008.
8. P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In *IEEE International Conference on Autonomic Computing, 2006. ICAC '06*, pp. 5–14, 2006.

Chapter 10
From Grid Islands to a World Wide Grid

P. Kacsuk and T. Kiss

Abstract The World Wide Web has become a phenomenon that now influences our everyday life in any possible areas and disciplines. This chapter describes a potential roadmap establishing its grid equivalent, a scientific, workflow-oriented, computational World Wide Grid (WWG). In order to achieve such a WWG, this chapter suggests three major steps. First, create uniform meta-brokers and connect existing production grids by them in order to form the WWG infrastructure where existing production grids become interoperable at the level of job submission. Second, create workflow-oriented, advanced grid portals and connect them to the meta-brokers in order to exploit workflow level grid interoperability. Finally, create workflow grid services and their registry and repository in order to make the workflows developed by different workflow communities interoperable and shareable.

Keywords Meta-broker · Grid portal · Grid workflow · World Wide Grid

1 Introduction

The goal of this chapter is to assess where we are in the road of establishing a scientific, workflow-oriented, computational World Wide Grid (WWG) that is similar to the World Wide Web (WWW) in the sense that anyone can access and use its services according to one's needs. If we look at the current trend of how and where grid develops, we can see that isolated production grids have been created based on different grid technologies that are not interoperable or only in a limited way. Although the different grids are capable to handle heterogeneous resources inside their own boundaries, the different technologies they use enforce a new level of heterogeneity onto a global grid infrastructure. One of the biggest challenges of the grid community is to solve the interoperability of these production grids in order to get closer to the idea of WWG.

P. Kacsuk (✉)
MTA SZTAKI Lab. of Parallel and Distributed Systems, H-1518 Budapest, Hungary
e-mail: kacsuk@sztaki.hu

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_10,

This chapter is intended to show that we are not far from creating a WWG if we make reasonable assumptions on how the users would like to use a WWG and also reasonable restrictions on how such a WWG can be used and for what purposes. The basic assumptions are as follows:

1. We restrict ourselves to a computational WWG.
2. The goal of the user is to dynamically collect and use as many grid resources as possible to accelerate his grid application.
3. The basic type of application users run in the WWG is a complex workflow.
4. The WWG will be used in the beginning by the scientific community.

Assumption 1 says that we restrict the first version of the WWG to a computational grid where the size of files used is not too large and hence they can efficiently be moved between computational grid resources. It does not exclude the usage of large files, but it significantly reduces the overall performance of the WWG if many users try to use such large files. Obviously, in a longer term the efficient management of large amounts of data should also be addressed in the WWG. However, since our goal is to demonstrate that a computational WWG is already feasible, we neglect this problem in this chapter. A follow-up paper will be written to outline the possible solutions for a data-oriented WWG.

Assumption 2 requires that all possible parallelism of an application should be exploited in the WWG. Users go for the WWG to access and use many resources in parallel to speed up the execution of their complex applications. The possible ways to exploit parallelism are analyzed in [18]. Assumption 3 might require some more explanation than the first two assumptions. Why workflow applications are so interesting for the WWG? The workflow concept abstracts a collection of services (tasks) that can be executed in a partially ordered way. As a result, workflows could be considered as the most general type of applications including any other types as special cases.

Assumption 4 is based on the current usage scenario of large e-Science grids. In order to establish a WWG that is used by the general public or by businesses, a significant improvement is required in the development of a grid market model. At first WWG will be used by the scientific community only; then, the grid market model could be much simpler than a real commercial one. Since our interest is to establish a WWG as soon as possible, it is better to start with a scientific WWG and later extend it to other directions.

At this point many readers could say that these assumptions are too restrictive and it is not worth defining and creating a WWG that can support only these kinds of applications and goals. We would argue that we cannot create the ultimate WWG at once. The WWW is much more complex today than it was in its initial stages. In the beginning it was used for scientific purposes and only later it was extended toward the commercial world. Once we established an infrastructure that is useful and works, then there are plenty of opportunities in the future to improve and extend that system. Even this restricted version of the WWG that is suggested in this chapter can be used to support many more applications than we can dream today. The establishment of such a WWG could tremendously widen the user community of the

grid and would significantly accelerate the take-up of grid technology worldwide. As a result, it would lead to a WWG that is usable for the whole population including commercial services as well.

In this chapter we identify and describe the main steps of creating a WWG system according to the assumptions mentioned above. The following sections describe three steps by which existing production grids (current grid islands) can be connected into a scientific computational WWG. Obviously, when the goal is to create a WWG as soon as possible, any proposed solution should be built according to the current situation. It should be accepted as a fact that production grids already exist and they are not willing to give up their freedom of developing their own roadmap and direction. Therefore, a WWG concept should be built on the autonomicity and collaboration of these existing production grids.

2 Step 1: Introduction of Meta-Brokers

Parameter sweep (PS) applications, where the same job or workflow must be executed with thousands of different parameters, require a very large number of resources. Current production grids are either small (having sites in the order of magnitude of tens with processors in the range of hundreds) or divided into similarly small VOs. As a result these grids or VOs are far too small to serve many simultaneously running PS applications. A suitable solution would be connecting these grids and VOs so that they could mutually help each other: if a grid (or VO) becomes overloaded it could direct jobs to other less loaded grids forming a WWG architecture. A crucial issue here is resource selection in order to achieve the highest possible parallelism, throughput, and load balancing among the grids that are ready to join and form a WWG. In a WWG system four models can be distinguished:

- User selected Grid User selected Resource (UGUR)
- User selected Grid Broker selected Resource (UGBR)
- Broker selected Grid User selected Resource (BGUR)
- Broker selected Grid Broker selected Resource (BGBR)

More and more production grids use brokers nowadays inside their own boundaries in order to select the most appropriate resources for execution. However, even if the user is allowed to submit jobs to several grids, it is his responsibility to select between these grids. Brokering at grid level is not supported in today's production environments. As a result, the BGBR model, that provides the largest freedom of accessing grid resources in the WWG, is hard to realize. The BGBR model means that the users can access several grids simultaneously and these grids have all got brokers. However, the user is not connected directly to a grid broker, rather to a new level of brokers called meta-broker. It is the task of the meta-broker to select the right grid according to the users' requirements as described in the Broker Property Description Language (BPDL) [23]. BPDL is similar to the JSDL [1], but it provides metadata about brokers (grids) and not about resources. Once the grid is selected

the meta-broker passes the job and its description (RSL [15], JDL [16], or JSDL depending on the actual grid) to the selected grid broker, and it will be the task of this broker to select the best resource according to the requirements specified in the job description or resource specification language file. This proposed meta-broker architecture is described in detail in [22].

Naturally, a single meta-broker would be a bottleneck in the WWG used by many thousands of scientists. Therefore, many uniform meta-brokers should be applied and these should be interconnected in order to realize load balancing among them. The architecture of the WWG based on the BGBR model and interconnected meta-brokers is shown in Fig. 10.1.

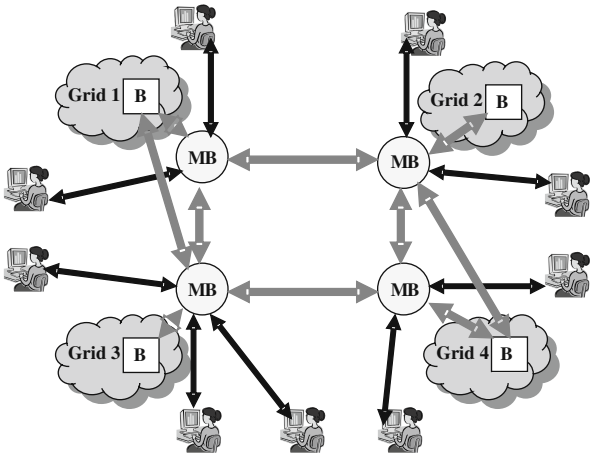


Fig. 10.1 Grids connected by a meta-broker

In such a WWG every client can be connected in a standard way to one of the uniform meta-brokers. As a result if we want to build the WWG, the only thing we have to do is to define and implement

1. The functionality of the meta-brokers
2. The intercommunication protocol of meta-brokers
3. The communication protocol of clients and meta-brokers
4. The standard intercommunication protocol of meta-brokers and grid brokers

The solutions for requirements 1 and 3 are already designed and described in [22]. Work is needed for requirements 2 and 4. Please note that the BGBR model is usable even if requirement 4 is not fulfilled by translating between the JSDL of the meta-broker and the job description format of the selected resource broker [24].

3 Step 2: Introduction of Advanced Grid Portals

After defining how to connect existing production grids into a WWG, it is time to see what we can provide at the user support level. The WWW became popular because its user interfaces (web browsers) are extremely easy-to-use, and anyone can access the WWW from any client machine without installing and maintaining complicated client software. We have to follow a similar route to make the WWG popular. Unfortunately, this criterion of success was tremendously overlooked in the first decade of grid computing. In current grid systems accessing even a single grid raises many problems:

1. The user has to learn the low-level command-line interface of the grid middleware that is far from being user-friendly.
2. The learning curve for the grid is quite steep.
3. The definition and execution of workflows and parameter studies are not standard and hence in many cases require the user's own development.

Using a multi-grid system like the WWG will just increase the number of problems:

4. Using another grid requires the learning of the low-level command-line interface of this grid, too.
5. Porting applications between grids based on different grid technology require substantial re-engineering of the grid application.
6. The user should be able to use these several grids simultaneously.

In order to solve problems 1, 2, and 4 the user needs a high-level workflow concept that could be implemented on top of any kind of grid middleware and hence it would hide the low-level details of the actual grid. Since the workflow application would run on top of any kind of grid middleware, porting the workflow application from one grid to another would not require any modification of the application code and in this way problem 5 could be solved easily. The WWG concept requires exactly this approach. In this case it does not matter which grid is selected by the meta-broker, the workflow application could run on top of the selected grid as well.

To support the development and execution of such high-level grid applications, we need workflow-oriented application hosting environments that can manage the workflows and coordinate the grid resources on behalf of the user. Such application hosting environment should be able to support the simultaneous usage of several grids and orchestrate grid resources even if they belong to different grid systems. This way problem 6 could be solved by the application hosting environment concept. This application hosting environment can be implemented in two ways: based on the thick client concept (e.g., the AHE [7]) or based on a portal and the thin client concept (e.g., the BIRN portal [26]). If grid portals are available, the user does not have to install and maintain the application hosting environment on his machine. He can use the grid from anywhere in the world through a simple web browser in the same way as the WWW can be accessed from any client machine. Due to this reason grid portals are more and more popular in scientific communities.

Problem 3 is a difficult issue. There are already a large number of workflow concepts (OMII-BPEL [37], Taverna [27], Triana [5], P-GRADE [20], Kepler [17], and GridAnt [36]) that are accepted by different communities. As all of these communities insist on their own workflow system, it would be extremely difficult to agree on a common standard. However, it is not unconditionally necessary. A solution for the interoperation of different workflow systems within the WWG is proposed in Section 4.

As a conclusion we can say that in order to solve problems 1–6, besides the introduction of meta-brokers (as described in Section 2), we also need

- portals,
- interoperable workflows and workflow concepts,
- workflow application hosting environments that provide workflow management.

These three components, integrated together, can solve the user interface problems of the scientific computational WWG. Notice that a grid portal that integrates these three aspects of the grid user interface is much more than people usually mean by a portal. In order to distinguish such integrated portals from the generally used simple job submission portals, we shall call them advanced grid portals (AGP).

Figure 10.2 summarizes the suggested WWG architecture so far. Meta-brokers are accessed via advanced grid portals, and the collaboration of AGPs and meta-brokers results in a convenient user access mechanism for the scientific computational WWG. So where are we? At this point we have a WWG infrastructure (corresponding to Fig. 10.2 where A = AGP, B = broker, and MB = meta-broker) that can execute even parameter study workflow applications in a dynamic, distributed, and parallel way exploiting the connected production grids and their resources. We have user interfaces (AGPs) that can be used to develop and execute workflows. This is now a very attractive WWG because it provides a very large number of resources

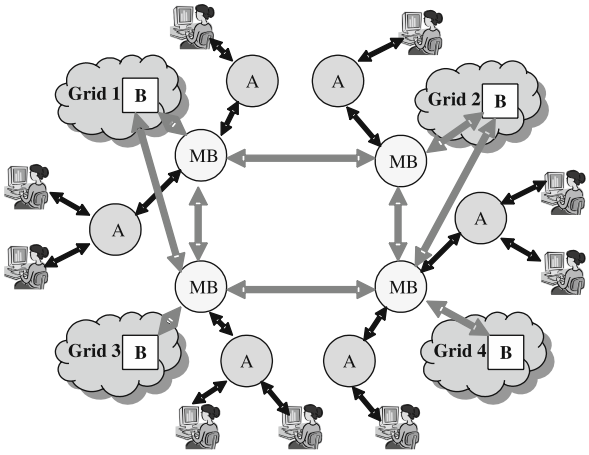


Fig. 10.2 User interface for the WWG

that can be accessed by anyone who collects the necessary grid certificates, and users can build and run their applications in a very convenient way.

The limitation of this WWG stage is that users cannot use each other's results, i.e., an application workflow developed on a certain portal by user X cannot be used by user Y on a different type of portal. Not mentioning that a workflow developed by user Y cannot use as a workflow node another workflow developed by user X. In short, workflow interoperability is still missing.

4 Step 3: Network of AGPs to Realize Workflow Interoperation

There are many different existing workflow models and different AGPs can realize any of these models [36, 3, 34, 9]. It means that different types of workflows can be developed by the different AGPs even for the same application. This leads to a very redundant development of grid applications. In order to avoid this waste of efforts the interoperation of grid workflows should be solved. We suggest the following extension of AGPs to accomplish this aim:

- Publish workflows as web services.
- Upload workflows as web services to workflow repositories.
- Find the requested workflows using web service search engines.
- Download workflows from workflow repositories.
- Insert the downloaded workflow as a node into the higher level workflow.
- Execute the whole workflow including the embedded workflows.

Let us introduce several definitions first. We call a workflow application exposed as a web service and able to run on some of the existing grid systems a workflow grid service (WFGS). A WFGS can be invoked as a usual web service and the same way as web services can invoke each other, WFGSs should be able to invoke each other. A WFGS should have a type including at least three attributes:

- Type of the workflow showing that the WFGS was defined according to workflow concept type X (Taverna, Triana, P-GRADE, etc.).
- Type of the grid where the WFGS can run (GT4, gLite, UNICORE, etc.).
- Type of the portal by which it was developed and can be managed (MyGrid [9], P-GRADE, etc.).

It also means that the workflow concepts, grids, and portals have types. The grid community should agree which these types are (the most frequently used workflow concepts, grids, and portals should be considered here). This type definition should be flexible enough to be easily extended with new emerging types. The overall goal is that the execution of jobs (or services) of a WFGS could be initiated by the user from any portal (his favorite portal) and he can also get the results back to this portal.

In order to achieve the goal written above the following solution is proposed:

- Create a WFGS repository (WFREP) where correct executable workflows are stored as web services.

- Create a WFGS registry (WFREG) where the workflows stored in the WFREP are registered with their type. Web services use UDDI and a similar registry can be used here as well.
- A WFREG record should contain a web service interface description of the workflow extended with the workflow type.

Once a user developed a workflow and would like to publish it for the whole user community, he has to register the workflow as a WFGS providing the type of the workflow as defined above. This description can later be extended with domain-specific metadata information allowing even semantics-based search of workflows. Portals should also register themselves in a portal registry (PREG). A portal registration record defines the type of the portal, the type of workflows it can execute, and the locations of available portals belonging to this class.

If a user through portal X would like to execute a WFGS, he can browse the workflow registry looking for a suitable workflow description. Once it is found, the user can initiate the execution of this workflow with the required input sets. The portal will check whether the WFGS type can manage such a workflow. If yes, the portal starts the execution of the workflow. If not, the portal checks in the portal registry if there is any portal that can manage the execution of such type of workflow. Once such a portal is found, the portal sends a request to the selected portal.

Notice that portals should be able to communicate with each other and hence a network of portals should be established. These portals can be connected to meta-brokers, grid brokers or both of these. In any case, these portals should be able to communicate with each other and access the WFREG, WFREP, and PREG services.

The described concept means that every grid can set up its own specialized portals but these portals can collaborate with other portals in order to realize workflow interoperability. The communicating portals can also help in creating a balanced usage of the portals. Overall, we have to define the following protocols:

- Communication between portals.
- Communication between portals and WFREG, WFREP, and PREG services.

The final architecture of the scientific computational WWG as described above is shown in Fig. 10.3 (where A = AGP, B = broker, MB = meta-broker). For the sake of simplicity WFREG, WFREP, and PREG services are presented as centralized services, but naturally they could be distributed services, too. The main features of this WWG architecture are as follows:

- There can be an unlimited number of grids connected in the WWG.
- To connect a grid to the WWG means to connect its broker (B) with a meta-broker (MB).
- The meta-brokers are connected by a network (e.g., P2P network).
- Portals can be connected to a grid's broker (special grid portal), a meta-broker, or both.
- Portals are connected to each other by a network (e.g., P2P network).
- Portals are connected to the WFREG, WFREP, and PREG services.
- Users can access the grid

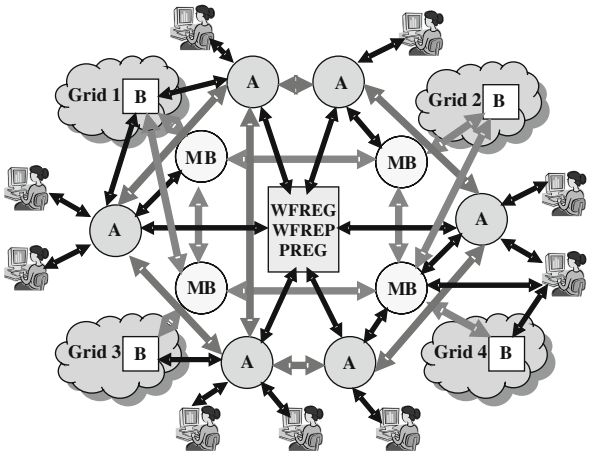


Fig. 10.3 The final architecture of the WWG

- via portals (either special or generic portals) and can execute any grid service (workflow) that is registered in WFREG and stored in WFREP
- directly via a grid broker (this access is for advanced users)
- directly via a meta-broker (this access is for advanced users).

5 Related Work

The most notable work related to grid interoperability is carried out within the framework of the GIN initiative [11] of the OGF. As written there, "The purpose of this group is to organize and manage a set of interoperation efforts among production grid projects interested in interoperating in support of applications that require resources in multiple grids." The GIN related web page of the UK NGS [35] writes, "Grid Interoperation Now (GIN) is a Community Group of the Open Grid Forum (OGF). It aims to organize and manage a set of interoperation efforts among production grid projects to support applications that require resources in multiple grids." Obviously the goal of the GIN is very close to the objectives of establishing a WWG although their ambitions do not go so far. The phase 1 tasks of the GIN VO is "to plan and implement interoperation in specific areas, initially data location and movement, authentication/authorization and identity management, job description and execution, and information services."

The GIN has created the GIN VO with resources from the major production grids: TeraGrid [33], UK NGS [35], EGEE [30], OSG [32], and NorduGrid [31]. All these grids allocated some grid sites to do experiments on their interoperability. In the framework of the GIN VO activity a GIN Resource testing portal [12] has been set up based on the P-GRADE/GEMLCA portal technology [8]. This portal enables the job submission (even workflow submission) to all these grid sites and

hence can be used to constantly monitor their availability and usability in the GIN VO. Although the goals of the GIN and the WWG described in this chapter have many similarities the concept of the GIN is quite different from the concept of the WWG.

A major European effort in providing grid interoperability between gLite [30], UNICORE [10], and Globus [25] is the OMII-Europe project [29] that tries to establish interoperability at the level of five services (JSDL, OGSA-DAI, VOMS, RUS, and GridSphere).

The World Wide Grid testbed [39] initiated by Monash University has the same name as we used in this chapter but their WWG has a quite different meaning than our WWG. Their WWG is not about connecting the existing production grids in order to establish an interoperable WWG, rather it is a volunteer grid testbed specifically intended to test the grid economy concept developed in the grid economy project [13]. Their recent paper on InterGrid [2] shows many similarities with our concept of connecting existing grids into a WWG. They introduce InterGrid Gateways to connect the different grids while we propose the usage of meta-brokers. They do not emphasize the importance of advanced grid portals and workflow interoperability but they put more emphasis on grid economy.

Another important approach is the OurGrid architecture that tries to restore the idea of symmetric volunteer grids as it was in the early days of grid computing. The main idea of OurGrid is to make the grid fast, simple, scalable, and secure. These are the requirements that can attract very large number of laboratories to connect their machines and clusters into a large grid installation [6]. Although they do not intend to build a WWG based on the OurGrid concept, in fact, its success could lead to a WWG system. Unfortunately, so far not too many laboratories joined OurGrid showing that large well-paid national and international grids like EGEE, UK NGS, and TeraGrid. have more chance to attract resources even if they require more maintenance work.

Solving grid interoperability based on resource management is not new as shown in a survey report of the CoreGrid project [24]. However, the meta-broker concept for providing grid interoperability is quite new and was first proposed by Kertesz and Kacsuk [21] at the CoreGrid workshop organized in conjunction with Euro-Par 2006. Since that time another CoreGrid partner, the Technical University of Catalonia (UPC) has started to work on this concept. The redefinition of the Broker Property Description Language (BPDL) [24] is an on-going joint work between SZTAKI and UPC. The detailed architecture plan of the meta-broker is described in [22] and further refined in [24].

The alternative approach of solving grid interoperability based on resource management tries to make existing grid schedulers interoperable. This concept is investigated by the GSA-RG of OGF [14] and by the Latin American grid initiative (LA grid) [4]. The drawback of this concept compared with the meta-broker approach is that it requires the modification of the existing grid brokers. However, the production grids are very conservative in accepting any new extension to their working services and hence the chance that the necessary extensions will be introduced and applied in the existing production grids is quite low. Another drawback of this concept is

that it requires a common agreement of all the major grid systems on the necessary intercommunication protocol standard. Although OGF works on this standardization issue, it is still in the research stage so it will take a long time to produce the final standard accepted by all the major grids. In contrast, the meta-broker concept does not require any standardization procedure in the beginning, provided that the meta-broker has the necessary translator and invoker components for all the major grid brokers as written in Section 3.

Supporting grid interoperability by advance portal systems (or application hosting environments) is also a known research direction. Two notable representatives of this direction are the P-GRADE portal extended with GEMLCA that can connect GT2, GT4, LCG-2, and gLite grids at workflow level [19] and the portal developed in the HPC-Europa SPA project [28]. Integrating these portals with the meta-broker concept is a work in progress [24].

Recently workflow interoperability became a major issue that was explicitly addressed during the Workshop on Scientific and Scholarly Workflows [38].

6 Conclusions

This chapter described three major steps to establish a scientific, workflow-oriented, computational World Wide Grid:

1. Create meta-brokers and connect existing production grids by them in order to form the WWG infrastructure where existing production grids become interoperable at the level of workflows.
2. Create workflow-oriented advanced grid portals and connect them together and to the meta-brokers.
3. Create workflow grid services and their registry and repository in order to make them interoperable.

The order of these steps is important and represents the priority among the steps. Step 1 is an indispensable action in order to establish the WWG infrastructure. Notice that it does not require the interoperability of different grids at every level of the grid middleware. This is exactly the advantage of this concept. It provides interoperation only from the user's perspective at the application level. The only concern of the user is that his application should be distributed all over the WWG resources in order to exploit as many resources as necessary. The meta-broker concept requires only one protocol definition and standardization, namely the communication protocol between the meta-broker and the brokers of the production grids. Once this protocol is agreed upon, the only thing the production grids should do in order to join the WWG is to realize this protocol as an extension to their broker.

In the meantime, until this protocol is not agreed, the developer of the meta-broker can extend the meta-broker architecture with the necessary translator and invoker units that fit to the requirements of the different production grids and enable the connection of the different production grids to the WWG without any

modification of the production grid middleware (including its broker). The development of the meta-broker is an on-going work in SZTAKI in the framework of the EU CoreGrid project. The translator and invoker modules for the EGEE broker and the GTBroker are already under development [22].

Step 2 extends the usability of the WWG established in Step 1 with the usage of workflow and parameter sweep workflow applications that can be managed by advanced grid portals (AGP). These AGPs contain workflow managers and by collaborating with the meta-broker, the nodes (jobs and service calls) of the workflow application can be distributed among the different grids of the WWG. In order to connect AGPs to meta-brokers the communication protocol between the meta-broker and the AGPs should be defined, standardized, and implemented. Once it is done, the AGP's user community can access the WWG and exploit all the resources of the WWG (provided that they have certificate to those resources).

Step 3 further extends the usability of the WWG by enabling the shared and interoperable usage of workflows developed by different communities. It means that any community using its own workflow concept can easily use a complete workflow developed by another workflow community in the form of a workflow grid service (WFGS), and place the call of such a WFGS as a node in its own workflow. Unlike Steps 1 and 2 that are underdevelopment and close to be finished, work on Step 3 has not started yet. Within the framework of the EU CoreGrid project we plan to integrate the Triana and P-GRADE workflow systems as written in this chapter.

Acknowledgment This research work has been carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

References

1. A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. *Job Submission Description Language (JSDL) Specification, Version 1.0*. `http://www.gridforum.org/documents/GFD.56.pdf`
2. M.D. Assuncao, R. Buyya, and S. Venugopal. InterGrid: A case for internetworking islands of grids. In *Concurrency and Computation: Practice and Experience (CCPE)*, Wiley Press, New York, USA, Jul. 16 2007. Online ISSN: 1532-0634, Print ISSN: 1532-0626.
3. R. Barbera, A. Falzone, and A. Rodolico. The genius grid portal. In *Computing in High Energy and Nuclear Physics*, La Jolla, California, 24–28 Mar. 2003.
4. N. Bobroff, L. Fong, S. Kalayci, Y. Liu, J.C. Martinez, I. Rodero, S.M. Sadjadi, and D. Villegas. Enabling interoperability among meta-schedulers. In *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pp. 306–315, May 2008.
5. D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang. Programming scientific and distributed workflow with triana services. *Grid Workflow 2004, Concurrency and Computation: Practice and Experience*, 18(10):1021–1037, Aug. 2006. ISSN: 1532-0626.
6. W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray. Labs of the World, Unite!!! *Journal of Grid Computing*, 4(3):225–246, 2006.
7. P.V. Coveney, M.J. Harvey, and L. Pedesseau. Development and deployment of an application hosting environment for grid based computational science. In *Conference of the Proceedings of the UK E-Science All Hands Meeting*, pp. 874–881, Nottingham, UK, 19–22 Sep. 2005. ISBN 1-904425-534.

8. T. Delaitre, T. Kiss, A. Goyeneche, G. Terstyanszky, S. Winter, and P. Kacsuk. GEMLCA: Running legacy code applications as grid services. *Journal of Grid Computing*, 3(1–2):75–90, 2005.
9. S.R. Egglestone, M.N. Alpdemir, C. Greenhalgh, A. Mukherjee, and I. Roberts. *A Portal Interface to myGrid Workflow Technology*. http://www.mrl.nott.ac.uk/~sre/papers/NETTAB2005/nettabextabst.doc
10. D.W. Erwin and D.F. Snelling. UNICORE: A grid computing environment. *Lecture Notes in Computer Science*, 2150:825–834, 2001.
11. GIN Project. https://forge.gridforum.org/projects/gin
12. GIN Resource testing portal. https://gin-portal.cpc.wmin.ac.uk:8080/gridsphere/gridsphere
13. Grid Economy project. http://www.gridbus.org/ecogrid/
14. Grid Forum. https://forge.gridforum.org/sf/projects/gsa-rg
15. GT 2.4: The Globus Resource Specification Language RSL v1.0. http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html
16. JDL Job description language. http://www.grid.org.tr/servisler/dokumanlar/DataGrid-JDL-HowTo.pdf
17. M. Jones, E.A. Lee, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, Aug. 2006. ISSN 1532-0626.
18. P. Kacsuk, A. Kertesz, and T. Kiss. Can we connect existing production grids into a world wide grid? Submitted to the VECPAR '08. In *8th International Meeting on High Performance Computing for Computational Science*, Toulouse, France, 24–27 June, 2008.
19. P. Kacsuk, T. Kiss, and G. Sipos. Solving the grid interoperability problem by P-GRADE portal at workflow level. In *Proceedings of the GELA'2006 Workshop in Conjunction with HPDC'06*, Paris, 2006.
20. P. Kacsuk and G. Sipos. Multi-grid, Multi-user workflows in the P–GRADE grid portal. *Journal of Grid Computing*, 3(3–4):221–238, 2005.
21. A. Kertesz and P. Kacsuk. Grid meta-broker architecture: Towards an interoperable grid resource brokering service. In *CoreGRID Workshop on Grid Middleware in conjunction with Euro–Par 2006*, pp. 112–116, Dresden, Germany, Aug. 28–29, 2006. Springer-Verlag Berlin Heidelberg.
22. A. Kertesz and P. Kacsuk. Meta-broker for future generation grids: A new approach for a high-level interoperable resource management. In *CoreGRID Workshop on Grid Middleware in Conjunction with ISC'07 Conference*, Dresden, Germany, Jun. 25–26, 2007.
23. A. Kertesz, I. Rodero, and F. Guim. Data model for describing grid resource broker capabilities. In *CoreGRID Workshop on Grid Middleware in Conjunction with ISC'07 Conference*, Dresden, Germany, Jun. 25–26, 2007.
24. A. Kertesz, I. Rodero, and F. Guim. Meta-Brokering requirements and research directions in state-of-the-art grid resource management. Technical Report TR-0116, Institute on Resource Management and Scheduling, CoreGRID – Network of Excellence, Nov. 2007.
25. I. Foster and C. Kesselman. The globus project: A status report. In *Proceedings of the Heterogeneous Computing Workshop*, pp. 4–18. IEEE Computer Society Press, 1998.
26. J. Novotny, R. Manansala, and T. Nguyen. BIRN portal overview. In *Portals & Portlets 2006*, Edinburgh, UK, 17–18 Jul. 2006. http://www.nesc.ac.uk/action/esi/download.cfm?index=3246
27. T. Oinn, M. Addis, J. Ferris, D. Marvin, T. Carver, M.R. Pocock, and A. Wipat. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics Journal*, 20(17):3045–3054, Jun. 2004.
28. A. Oleksiak, A. Tullo, P. Graham, T. Kuczyński, J. Nabrzyski, D. Szejnfeld, and T. Sloan. HPC–Europa single point of access as a framework for building science gateways. *Concurrency and Computation: Practice and Experience*, 19(6):851–866, 2006.
29. OMII–Europe Project. http://www.omii-europe.com/

30. The gLite website. http://glite.web.cern.ch/glite/
31. The NorduGrid website. http://www.nordugrid.org/
32. The OSG website. http://www.opensciencegrid.org/
33. The TeraGrid website. http://www.teragrid.org/
34. H.-L. Truong, B. Baliś, M. Bubak, J. Dziwisz, Th. Fahringer, and A. Hoheisel. K–WfGrid distributed monitoring and performance analysis services for workflows in the grid. In *2nd IEEE International Conference on e–Science and Grid Computing*, IEEE, Amsterdam, 2006.
35. UK NGS web page on GIN. http://wiki.ngs.ac.uk/index.php?title=GIN
36. G. von Laszewski, K. Amin, M. Hategan, N.J. Zaluzec, S. Hampton, and A. Rossi. *GridAnt: A Client–Controllable Grid Workflow System*. http://www-unix.mcs.anl.gov/~laszewsk/papers/vonLaszewski--gridant-hics.pdf
37. B. Wassermann, W. Emmerich, and B. Butchart Sedna. A BPEL-based environment for visual scientific workflow modelling. *Workflows for e–Science*, pp. 428–449, 2007.
38. Workshop on Scientific and Scholarly Workflows. https://spaces.internet2.edu/display/SciSchWorkflow/Agenda+and+Presentations
39. World Wide Grid testbed. http://www.gridbus.org/ecogrid/wwg/

Chapter 11
The Anatomy of Grid Resource Management

A. Kertész and T. Prokosch

Abstract Grid resource management is a highly studied research field since grids were born. Though well-designed, evaluated, and widely used resource brokers and meta-schedulers have been developed, new capabilities are still required; the most wanting one is interoperability. Existing solutions cannot cross the borders of current middleware systems that are lacking the support of these requirements. In this chapter we first investigate the current resource management solutions from the lowest to the highest level of abstraction, then we examine and compare the building blocks of these systems. The presented anatomy helps the users grasp the basics of their operation and the researchers to identify common components and open issues. We highlight remote instrumentation as one of these issues, and meta-brokering, which enables higher level resource management by establishing communication among existing grid brokers or utilizing them in a centralized manner. The latter is a forerunner of an ongoing grid revolution by implementing a service-oriented knowledge utility (SOKU) service, which is to serve future generation grids.

Keywords Grid interoperability · Grid resource management · Grid broker · Grid scheduler · Grid meta-broker · Remote instrumentation

1 Introduction

More than a decade has passed and grid computing has become a detached research field targeted by many worldwide projects. Nowadays research directions are focusing on user needs; more efficient utilization and interoperability play the key roles. Grid resource management has always been in the spotlight; as grid technologies matured, new challenges arose. Though well-designed, evaluated, and widely used resource brokers and meta-schedulers have been developed, new capabilities are still required – the most wanting is interoperability support. Since most of the existing solutions depend on other grid middleware capabilities and services, they

A. Kertész (✉)
MTA SZTAKI Computer and Automation Research Institute, H-1518 Budapest, Hungary
e-mail: attila.kertesz@sztaki.hu

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_11,

can hardly cross the border of these middleware systems: They need revolutionary changes affecting the whole system. Many resource management tools have been developed named with different acronyms and built on different middleware components. This variety has shaped grid resource management to a blurry, grey cloud, in which neither the users nor the researchers can see how these tools are related and what their relevant properties are. Until these definitions and connections have not been clarified, further development and interoperability cannot be facilitated. In this chapter we present the anatomy of current grid resource management by revealing the connections of the main components and giving general definitions of similar solutions.

We start our investigation at the lowest, fabric level, where most of the resources are, and finish the list with grid portals at the highest, application level. Local resource managers are responsible to perform resource provisioning at the lowest level in grid resource management; a survey of the widely used ones can be read in [2]. Grid brokers or grid resource management systems operate one level higher in grid middleware systems; taxonomies in [11, 13] cover the well-known solutions. Workflow managers [19] can be found at higher levels, but still belong to the middleware. At the highest level (application level), grid portals provide easy to use interfaces to grid resources.

In the following section we go through these categories and the related tools by depicting their components. This classification should give the grid architect an idea what services exist and how they are related to each other, so that they can understand why interoperability is one of the key issues which need to be addressed in the near future.

In the third section we state the emerged requirements and new components in this field. For one, we introduce remote instrumentation [4] that is going to affect grid resource management in the near future. Another relevant component that gets ever more attention is the meta-broker. Meta-brokering is a novel approach to establish interoperability by the utilization of the existing grid brokers. Since the advance of grids seems to follow the way envisaged and assigned by the Next Generation Grids Expert Group [14], this new component should be designed as a SOKU (service-oriented knowledge utility [14]) that enables more flexibility, adaptability, and additionally has advanced interfaces. With the help of a meta-brokering service the World Wide Grid (WWG), a consolidation of grids, can be established in a similar manner as the Internet was born. Two works have already envisaged a future world wide grid:

- One is InterGrid [1], which promotes interlinking of grid systems through peering agreements to enable inter-grid resource sharing. This approach is a more economical view, where business application support and sustainability are primal goals. In this new architecture so-called IntraGrid resource managers (IRM) would play the role of resource brokers. The InterGrid gateway (IGG) would be responsible of making agreements with other grids through their IRMs. Its main task is to manage peering agreements and discover and allocate resources from different domains.

- The second approach is shown in [9], which states that three major steps are required to create the WWG architecture: Existing production grids should be connected by uniform meta-brokers, workflow-oriented portals should interface these meta-brokers, and a repository of workflow grid services should be created. As a final solution the inter-connected meta-brokers should solve the interoperability among different grids through uniform interfaces.

2 The GRM Anatomy

Taking a closer look on current grid resource managers and research directions we are facing confusing contradictory definitions or references. When searching and comparing related works we meet with meta-schedulers, local and global schedulers, brokers, resource allocation managers, and some more. In this section we gather and classify the expressions used in the area of Grid resource management. We refer to these definitions further on in this chapter by naming specific components, building blocks of these solutions. In Fig. 11.1, we can see the architecture of a grid system focusing on resource utilization. All the abbreviations used in this figure denote collections of similar components or services. In the following we go through these groups by gathering the generally used acronyms and expressions for components having similar roles.

R – resource: In general it means a physical machine, where user programs are executed. We distinguish between two, possibly three, types regarding utilization:

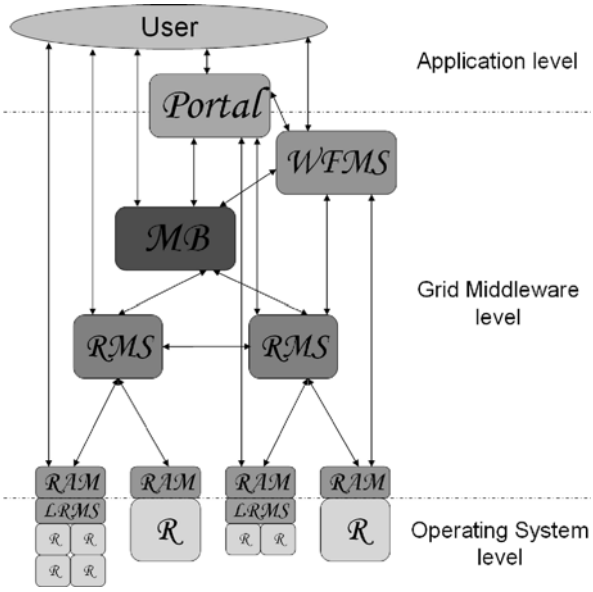


Fig. 11.1 Grid resource managers and their connections

computing element (CE), where the user program (also task or job) execution takes place, storage element (SE), where user data is stored, and possibly instrument element (IE) as defined in the domain of remote instrumentation. Remote instruments (RI or IE as instrument element [4]) are various tools to access, process, produce, or consume data usually by physicists, chemists, biologists, or other researchers. Finally, web services (WS) or other grid services (GS) could also be regarded as resources, since they similarly produce output for a given user input.

LRMS – local resource management system, scheduler, local scheduler, local resource manager, and sub-scheduler: These tools are usually cluster managers that were taken from high performance and distributed computing, and now generally used in grid systems. The widespread ones are PBS, SGE, and LSF [2].

RAM – resource access manager: This is usually a component of a grid middleware that accesses the resource directly and handles the arriving jobs. It provides an interface for the middleware to the resource. An example is GRAM in the Globus Toolkit [3].

RMS – grid resource management system, meta-scheduler, global scheduler, resource broker, grid scheduler, and orchestrator: It usually consists of one or more components of a grid middleware. It can be an internal middleware service or an external tool that uses middleware components or services. Its role is similar to the role of LRMS; this is the reason why they can be confounded sometimes. While the LRMS schedules user tasks in a cluster among free processors, the RMS schedules jobs at the level of grid middleware by selecting a resource that best matches the requirements of these jobs. (The selected resource can be a cluster with an LRMS.) Therefore an LRMS is generally called a scheduler, and the RMS is a meta-scheduler. The other listed expressions for RMS basically mean the same tool. Some of them only slightly differ, some of them support different middleware solutions, job types, agreements, or quality of services. A taxonomy in [11] reveals these properties and the differences among the lately used brokers. Figure 11.2 is used to reveal the differences among the used expressions. In general a meta-scheduler is focusing more on job scheduling (executing an algorithm), a resource broker incorporates more services such as job handling and job state tracking, while a resource management system takes care of the whole job management process with job accounting and other related services. A meta-scheduler and a broker can communicate with and use other services of the middleware to be able to manage the whole job's life-cycle. In this sense scheduling is an atomic process, brokering incorporates scheduling, and resource management includes brokering.

MB – meta-broker, inter-domain broker, and inter-grid gateway: Meta-brokering is a novel approach that introduces another level above current grid resource managers in order to facilitate inter-grid load balancing and more efficient brokering. The idea was first published in [9], in which the grid meta-broker sits on top of the resource brokers and uses meta-data to decide where to send a user's job. Section 3 deals with this new component and with the related research efforts.

WFMS – workflow management system, workflow manager, workflow enactor, and workflow scheduler: The role of this component is to manage workflows. Its

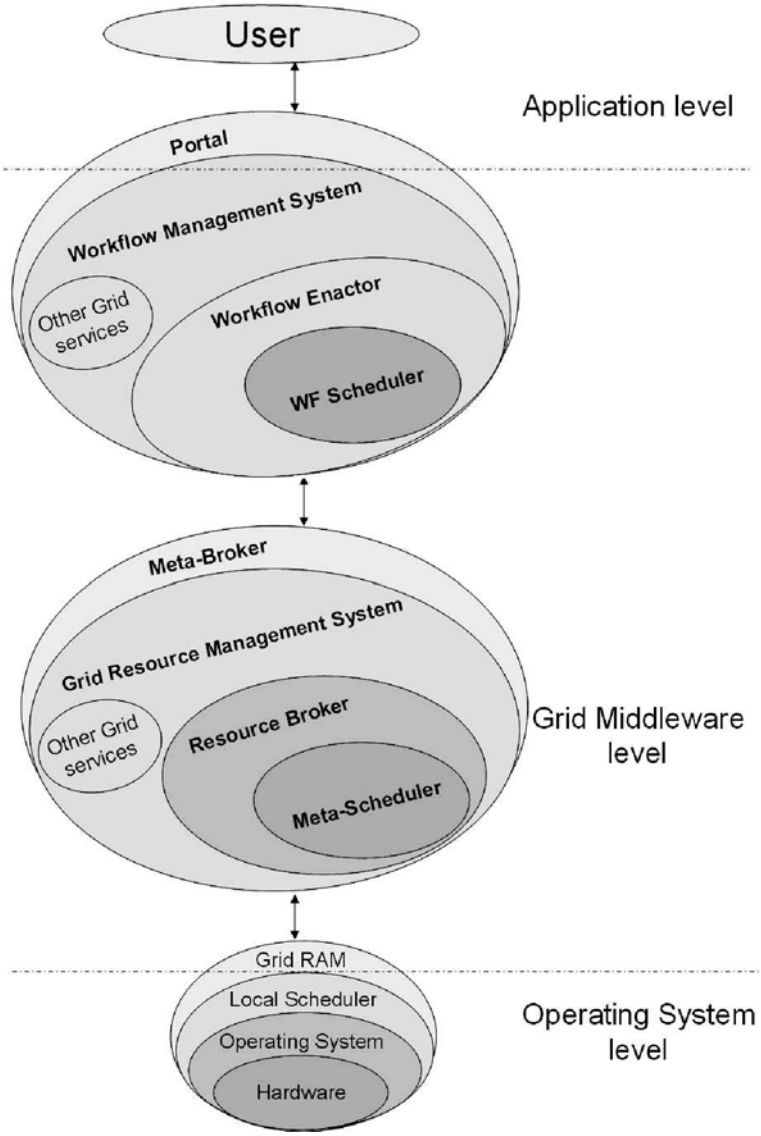


Fig. 11.2 The anatomy of grid resource management

core is the workflow scheduler, which is responsible to submit the jobs in the workflow in a way that the execution of the whole workflow will be the cheapest regarding some cost function (time, budget, etc.). These workflow enactors usually connected to one or more RMSs that take care of the actual job scheduling. The workflow management system can be connected to more middleware services. This more complex tool usually incorporates resource-related information into its

scheduling (not just workflow properties); in this case it is better to access the resources directly and manage the whole grid resource management.

Portal – grid portal, problem-solving environment: This tool provides a high-level user friendly environment for accessing grid services. It usually uses a graphical user interface to define jobs, create workflows, submit them, and track their states. It is usually connected to or incorporates some WFMS or RMS to manage resources. More information on grid portals can be found in [8].

We have now seen how resource management is carried out by various tools from the fabric level up to the user level. This summary reveals the connections and possible interfacing parts among these components. As production grids need to become sustainable, researchers need to pay more attention to enhance interoperability among the different solutions. The following section deals with interoperability efforts in current research directions in the field of grid resource management and highlights open issues for future extensions.

3 Interoperability Issues and Requirements for New Solutions

The proper solution to interoperability is to design the system in the first place so that all components know how to talk to each other. Protocols need to be established and followed. However, when protocols are diverse by themselves, the classical approach to this problem is to create a superior instance, a "mediator" in a sense, which has the only task to provide an interface between the individual components. This approach is taken in the grid world: Here, this software component is known as a "meta-broker".

3.1 Interoperability Problems at All Levels: The Solution Is Meta-Brokering

Interoperability problems appeared not only in resource management but also in other parts of grid middleware. Furthermore other components have a great impact on this field, since most of them rely on resource managers and connect to them. The lack of standards is a problem that cannot be solved right away. Figure 11.1 shows the relevant actors in grid resource management. Below the RMS level no changes can be introduced, since this lower layer is part of the core of every middleware system, therefore all of them would need to be redesigned to support a common standard. Above the RMS level each component could actually help enhancing interoperability.

Starting from the user interface, the diversity in job description languages means the first problem: though the JSDL [5] is a standardized language, only few systems have already adopted it. Portals are the most obvious place for solving the whole interoperability problem. While one could make an effort by supporting all the job description languages, interfacing all resources, RAMs, RMSs, WFMSs, and related

middleware services, this effort would be destined to fail. They need the help of some new component that takes up some of these duties. One solution could be a general WFMS that supports all the workflow description formats [6], interfaces the RMSs or RAMs, and can be integrated to every portal. This seems to be an achievable solution, since only some services need to be supported, but still many descriptions have to be learned, and adapting such a WFMS would need high efforts from the portal developers. The only solution left would be the MB, the new meta-brokering component. It is high level enough to hide and manage the RMSs and low level enough to be integrated into portals and WFMSs. In the following we summarize the latest research done in this area.

The goal of a meta-brokering layer is to establish a connection between individual grids (domains or "Virtual Organizations", in short VO) by managing their RMSs. Most of the research groups that identified this requirement follow the vision of GSA-RG of OGF [15], which plans to enable grid scheduler interaction by defining a standard inter-communication protocol and interface. Once this standard will be published, all RMSs need to support it. However, standardization takes time, and so Gridway [18], Koala [7], and the LA Grid [17] researchers took a step forward, and defined their own interfaces and integrated them into their RMS instances. Each instance is responsible for serving a domain, and when it becomes overloaded or a request cannot be fulfilled, it turns to a neighboring instance for execution. After all we still need to wait for an OGF standard, because in the current situation the different RMS instances still cannot communicate with each other. Our idea was to design a new middleware component called "Grid Meta-Broker" [10] that hides the different job descriptions and RMS interfaces and offers a general JSDL-based submission interface for portals and users. Since this meta-brokering service is middleware independent, it solves the interoperability problem and it can be easily interfaced by portals. Its operation (meta-brokering) is based on the different properties of RMSs. After we examined the available RMSs, we have created a language called BPDL [12] to store these properties. The meta-broker stores a BPDL of each utilized RMS and takes into account these scheduling capabilities during matching a user request to an RMS. This service has been designed and built on the latest Web standards, therefore it follows the foreseen path for grid development by the Next Generation Grids Expert Group. Its prototype has an agent-based architecture and it is possible to improve its design during the forthcoming evaluation in order to create a service-oriented knowledge utility (SOKU) service. Once this development is done and portals use these meta-brokers for accessing grids in an interoperable way, the world wide grid will be a real achievable goal by inter-connecting these meta-brokers. RMSs have different capabilities, therefore special user requests can be better served by some brokers. There are less supported capabilities such as advance reservation or co-allocation, and even new requirements can arise as grid technology matures. As more and more RMSs appear in production grids, the need for the meta-brokering layer will be more and more crucial. One of the newly arosen capabilities is the support of remote instrumentation. Though this field is still in its infancy, the next section introduces the requirements and possible solutions for support.

3.2 New Candidates for Resource Management: Remote Instrumentation

The RINGrid European Project [16] has been initiated to discover remote instruments, investigate the requirements of remote instrumentation, and design specifications of future grid utilization. One of the first tasks in this project was to investigate the state of the art in this field, which shows that some groundbreaking work has already been done, for example, the creation of the former mentioned Instrument Element (IE) as developed by the GridCC project [4]. Though the automatic management of these resources is unlikely to be a reality in a short term, we examine its requirements and place it in the anatomy described in Section 2.

Basically, having instruments as grid elements is in no way different than having computing and storage elements regarding resource management resources, therefore all the monitoring, accounting, and even scheduling and brokering services can stay in place. However, computing and storage resources can be handled in a trial-and-error fashion (making requests and handling failures), while this is not the case with instruments. Instruments are much more sensitive to poor operation and prone to hardware failure if they are stressed over their limits. As such, we need to be able to verify their state under all circumstances and refrain from operating them beyond their recommended limits. Some of this is addressed by the Instrument Element abstraction.

Some of the instrument's needs (or more generally: the researcher's needs) cannot be addressed at that level. Some enhancements need to be made within the RAM and RMS and even the WFMS level:

- The main requirement of an instrument-aware RMS is the knowledge of the capabilities of the instruments. Users need to specify special parameters, which set some properties of an instrument, to access resources. A description language needs to be created to cover these capabilities that can be used by the RMS to match a user request to a remote instrument. Once such a language is defined resource brokering can be extended with remote instrumentation.
- Instruments sometimes need to be accessed exclusively. For example, a telescope may only be able to watch a certain area of the sky at the same time. As a result, the instrument needs to be reserved in advance. We have previously already touched the topic of advance reservation and co-allocation. To be able to use instruments as first-class grid elements like all other grid resources, advance reservation and co-allocation capabilities in the RMSs need to be greatly enhanced.
- Conducting experiments in the grid also pushes the current workflow managers to their limits. While currently workflows are created to ease the researcher's life, they are crucial when talking about experiments. The reason is that some experiments are time critical, especially when the instrument creates a massive data stream of data of several Gbs – most of the time this data needs to be stored on storage elements in its raw format, but it may also happen that it needs to be distributed to thousands of computing elements right away. In parallel, the

network must not get congested so that the instrument's operator is still able to steer the instrument and set parameters for correcting possible misbehavior. Data management (which data resides where?) is a crucial point in this scenario.

4 Conclusions

Grid resource management has been a highly targeted research field since grids were born. Though well-designed, evaluated, and widely used resource brokers have been developed, new capabilities are still required. The variety of available resource management tools has shaped grid resource management into a blurry cloud, in which both users and researchers can hardly find what they are looking for. In this chapter we gathered all the aspects of resource management and presented an anatomy by revealing the connections of the main players to facilitate further development and interoperability. This summary helps the users grasping the basics of their operation and the researchers identifying common components and open issues. We have investigated remote instrumentation as one of these issues, and meta-brokering, which enables higher level resource management by establishing communication among existing grid brokers or utilizing them in a centralized manner. The discussed grid meta-broker is implemented as a SOKU service, which can be a forerunner of an ongoing grid revolution toward service-oriented future generation grids.

References

1. M.D. Assuncao, R. Buyya, and S. Venugopal. InterGrid: A Case for Internetworking Islands of Grids. *Concurrency and Computation: Practice and Experience (CCPE)*, 20(8):997–1024, Jul. 2007.
2. Y. Etsion and D. Tsafrir. A Short Survey of Commercial Cluster Batch Schedulers. Technical Report 2005–13, School of Computer Science and Engineering, the Hebrew University, Jerusalem, Israel, May 2005.
3. I. Foster and C. Kesselman. The Globus project: A status report. In *Proceedings of the Heterogeneous Computing Workshop*, pp. 4–18, IEEE Computer Society Press, 1998.
4. E. Frizziero, M. Gulmini, F. Lelli, G. Maron, A. Oh, S. Orlando, A. Petrucci, S. Squizzato, and S. Traldi. Instrument element: A new grid component that enables the control of remote instrumentation. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (Ccgrid'06)*, vol. 00, p. 52, IEEE Computer Society, Washington, DC, May 16–19, 2006.
5. Global Grid Forum. *Job Submission Description Language*. http://www.ggf.org/documents/GFD.56.pdf
6. Grid Workflow Forum. http://www.gridworkflow.org/
7. A. Iosup, D.H.J. Epema, T. Tannenbaum, M. Farrellee, and M. Livny. Inter-operating grids through delegated matchmaking. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC07)*, Reno, Nevada, Nov. 2007.
8. P. Kacsuk and G. Sipos. Multi-grid, multi-user workflows in the P-GRADE Grid Portal. *Journal of Grid Computing*, 3:1–18, Feb. 2006.
9. A. Kertesz and P. Kacsuk. Grid meta-broker architecture: Towards an interoperable grid resource brokering service. In *CoreGRID Workshop on Grid Middleware in Conjunction with*

Euro-Par 2006, pp. 112–116, Dresden, Germany, Aug. 28–29, 2006. Springer-Verlag Berlin, Heidelberg, 2007.
10. A. Kertesz and P. Kacsuk. Meta-broker for future generation grids: A new approach for a high-level interoperable resource management. In *CoreGRID Workshop on Grid Middleware in Conjunction with ISC'07 conference*, Dresden, Germany, June 25–26, 2007.
11. A. Kertesz and P. Kacsuk. A taxonomy of grid resource brokers. *Sixth Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS 2006) in Conjunction with the Austrian Grid Symposium 2006*, pp. 201–210, Austria, Sep. 21–23, 2006.
12. A. Kertesz, I. Rodero, and F. Guim. Data model for describing grid resource broker capabilities. In *CoreGRID Workshop on Grid Middleware in Conjunction with ISC'07 Conference*, Dresden, Germany, Jun. 25–26, 2007.
13. K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience (SPE)*, 32(2): 135–164, Feb. 2002. ISSN: 0038-0644.
14. Next Generation Grids Report. *Future for European Grids: GRIDs and Service Oriented Knowledge Utilities Vision and Research Directions 2010 and Beyond*, Dec. 2006. NGG3.
15. OGF Grid Scheduling Architectures Work-Group. https://forge.grid-forum.org/sf/-projects/gsa-rg
16. RINGrid Project Webpage. http://www.ringrid.eu/.
17. I. Rodero, J. Corbalan, F. Guim, L.L. Fong, Y.G. Liu, and S. Masoud Sadjadi. Looking for an evolution of grid scheduling: Meta-brokering. In *Proceedings of the Second CoreGRID Workshop on Middleware at ISC2007*, Dresden, Germany, Jun. 2007.
18. T. Vazquez, E. Huedo, R.S. Montero, and I.M. Llorente. Evaluation of a utility computing model based on the federation of grid infrastructures. *Euro-Par 2007*, pp. 372–381, Rennes, France, Aug. 28, 2007.
19. J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec., ACM Press*, 34:44–49, 2005.

Chapter 12
SZTAKI Desktop Grid: Adapting Clusters for Desktop Grids

A.Cs. Marosi, Z. Balaton, P. Kacsuk, and D. Drótos

Abstract Although clusters usually represent a single large resource for grid computing, most of the time they are utilized as individual resources for desktop grids. This chapter presents a lightweight, fault-tolerant approach for connecting firewalled already deployed clusters for desktop grid computing. It allows to easily offer cluster resources for desktop grids without requiring a large administrative overhead of connecting them separately. Our approach uses Condor for cluster management and an extended version of BOINC to connect those resources across the Internet or wide area networks. We describe security considerations and present our security model for the system and sharing resources. We present running projects where our solution is being used to utilize already deployed clusters requiring low effort from administrators.

Keywords SZTAKI Desktop Grid · BOINC · Condor · Cluster

1 Introduction

Originally, the aim of grid research was to realize the vision that anyone could donate resources for the grid and anyone could claim resources dynamically according to their needs, e.g., in order to solve computationally intensive tasks. This twofold aim has been, however, not fully achieved yet. Currently, we can observe two different trends in the development of grid systems, according to these aims.

Researchers and developers following the first trend are creating a service-oriented grid, which can be accessed by lots of users. To offer a resource for the grid, installing a predefined set of software (middleware) is required. This middleware is, however, so complex that it needs a lot of effort to install and maintain it. Therefore individuals usually do not offer their resources this way, but all resources are typically maintained by institutions, where professional system

A.Cs. Marosi (✉)
MTA SZTAKI, Computer and Automation Research Institute, Hungarian Academy of Sciences, H-1528 Budapest, Hungary
e-mail: atisu@sztaki.hu

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_12,

administrators take care of the complex environment and ensure the high availability of the grid. Offered resources are usually clusters of PCs rather than single resources.

A complementary trend can also be observed for realizing the other part of the original aim. According to this direction anyone can bring resources into the grid, offering them for the common goal of that grid. The most well-known example of such systems, or better to say, the original Internet-based distributed computing facility example is SETI@home [3]. In grids following the concepts of SETI@home, a large number of PCs (typically owned by individuals) are connected to one or more central servers to form a large computing infrastructure with the aim of solving problems with high computing needs. Such systems are commonly referred to by terms such as Internet-based distributed computing, public-resource computing, or desktop grids; we will use the term desktop grid (DG).

In this chapter we describe our approach to connect clusters of resources to desktop grids by using a lightweight bridge. The chapter is organized as follows. The next section discusses related work. Section 3 introduces SZTAKI Desktop Grid and details our bridge, Section 4 presents some projects using the bridge, and the last section concludes the chapter.

2 Related Work

2.1 BOINC and Condor

BOINC [2] (Berkeley Open Infrastructure for Network Computing), originated from the SETI@home project, is an effort to create an open infrastructure to serve as a base for all large-scale scientific projects that are attractive for public interest and having computational needs so that they can use millions of personal computers for processing their data. BOINC, in contrast to the original SETI@home distributed computing facility, can run several different distributed applications and yet, enables PC owners to join easily by installing a single software package (the BOINC core client) and then decide what projects they want to support with the empty cycles of their computers, without the need to delete, reinstall, and maintain software packages to change between projects.

Condor [19] is a complex cycle-scavenger platform, which is originating from the Condor research project at the University of Wisconsin-Madison. Condor connects several computers together to form an administrative domain, which is called pool in Condor terminology. There is one (or more) special machine in the pool which is called *submitter* (front end of the pool) and can be used to start jobs (called submissions). Condor can select the most idle machine from the pool to run the job so the load is balanced optimally. The behavior can be fine tuned; the administrator can set how Condor tries to figure out which is the ideal computer for the job according to the job's requirements. The job must be defined by a so-called submit file, which defines the executable, input and output files and constraints about the environment

it needs. Condor also allows to interconnect different pools to allow to exchange tasks and resources using a technique called *flocking*. Since Condor uses specific ports this could lead to problems connecting firewalled pools, although there are workarounds [14, 16].

While aiming for similar goals, Condor's approach is radically different from the DG concept. Condor employs a push model by which jobs can be submitted into a local resource pool or a global grid (friendly Condor pools or a pool overlaid on grids using different middleware). The DG concept on the other hand applies the pull model whereby free resources can call for task units. The scalability of Condor is limited by the centralized management implied by the push model (largest experiments are at the level of 10,000 jobs in EGEE but it requires a very complicated grid middleware infrastructure that is difficult to install and maintain at the desktop level).

Condor has partial support for running BOINC jobs. When it detects an idle computing slot on an execution machine and the pre-set conditions allow it enters the *backfill* state. This allows spawning of a command line BOINC core client, which then joins a predefined DG. This way each node represents an individual computing resource for the project requiring outbound Internet connection, which is not always allowed for nonfront-end nodes. Input data is downloaded after the core client is started, which may result in a significant overhead for the execution time if the size of the input data is large enough.

The goal of this approach is to utilize the computing cycles on the otherwise idle Condor worker node [20], and not to use the cluster as a single, dedicated and powerful DG client.

2.2 XtremWeb and Condor

XtremWeb [7] is a research project from University of Paris-Sud, France. Similar to BOINC it aims to serve as a substrate for global computing experiments. It supports the centralized setup of servers and PCs as workers. In addition, it can be used to build a peer-to-peer system with centralized control, where any worker node can become a client that submits jobs. XtremWeb manages tasks as follows. The *Coordinator* is responsible for a set of tasks and their scheduling among a set of computing resources (*Workers*), which are either provided by private or institutional users; thus, they are not administered by the Coordinator and might be volatile. XtremWeb workers are deployable as Condor tasks ("pilot jobs") [12] fetching their tasks from a predefined coordinator. This allows a lightweight solution to connect Condor resources to XtremWeb, it is done on a per resource (per task) basis.

For each Condor execution machine to be added an XtremWeb Worker needs to be run on the node as a task and allowed to connect to the remote coordinator, thus the bridged node requires outbound Internet access.

2.3 BOINC and the Grid

The LHC@home [11] project was started as master's thesis project [15] with the official topic to combine BOINC and grid resources. A bridge was deployed to map work units from BOINC to LCG-2 resources. Each task consisted of a single work unit and a modified client, which were sent to a worker node. The sole purpose of the client was to provide an execution environment for the task, the client did not perform any network activity, and the resource was not associated in any way with the DG. They have also investigated the possibility of a reverse direction bridge, mapping LCG-2 jobs to DGs.

Another approach, the Superlink@EGEE [17] queue, is utilizing EGEE resources by sending BOINC core clients to worker nodes which then connect to the DG server at Superlink Online. This is similar to the XtremWeb and Condor bridge: worker nodes require outbound Internet access and represent a single computing resource (each using a separate "bridge").

3 SZTAKI Desktop Grid and Condor

Today, most of the DG projects, including SZTAKI Desktop Grid (SZDG) [10], utilize BOINC because it is a well-established free and open source platform that has already proven its feasibility and scalability and it provides a stable base for experiments and extensions. BOINC provides the basic facilities for a DG in which a central server provides the applications and their input data, where clients join voluntarily, offering to download and run an application with a set of input data. When the application has finished, the client uploads the results to the server. BOINC manages the application executables (doing the actual work) taking into account multiple platforms as well as keeping a record of and scheduling the processing of work units, optionally with redundancy (to detect erroneous computation, either due to software or hardware failures or clients being controlled by a malicious entity).

The advantages provided by the DG concept are not only useful on a world-wide scale but also can be used for smaller scale computations, combining the power of idle computers at an institutional level or even at a departmental level. The basic building block of SZTAKI Desktop Grid is such a local desktop grid (LDG) connecting PCs at the given organizational level. SZTAKI LDG [5] is built on BOINC technology but is oriented for businesses and institutes. In this context it is often not acceptable to send out application code and data to untrusted third parties (sometimes this is even forbidden by law, for example, in the case of medical applications), thus these DGs are normally not open for the public, mostly isolated from the outside by firewalls and managed centrally. SZTAKI LDG focuses on making the installation and central administration of the LDG infrastructure easier by providing tools to help the creation and administration of projects and the management of applications. LDG also aims to address the security concerns and special

needs arising in a corporate environment by providing a default configuration that is tailored for corporate use and configuration options to allow faster turnaround times for computations instead of the long-term projects BOINC is intended for.

Clients are usually single computers, but in some cases a user would be able to provide more than one computer's computing power for a DG. Traditionally the user must deploy a client on every node she has and manage them individually. Another way to provide multi-computer resources for a DG could be connecting computers together and building up a virtual multiprocessor environment and presenting this environment as a single resource for the DG. Computers can be clustered using a batch management system. These systems usually use a particular node (submitter) to start jobs and the system automatically selects a worker from the cluster to run it. We can think of two scenarios when connecting clusters to a DG. In the *first scenario*, although we refer to the cluster as one large resource, if it were to be connected to a DG as a client, each node would have to be added separately. The ideal solution is a bridge which needs to be installed only on the submitter of the cluster, whose task is to get enough work units from the project server to fill up the whole cluster with jobs (with respect to local policies). This approach is intended for already deployed clusters that need to be added as a resource to an (existing) DG, most likely in industrial or institutional environments, thus without introducing a large administrative overhead. The *second scenario* is to connect distinct administrative domains of Condor clusters (pools) using a bridge to safely share resources by forming a large DG. Applications are always deployed at the DG server, using its security measures and tasks are dispatched by via the bridge to the cluster nodes.

3.1 Security

The question arises, how can I trust the tasks sent to my pool not performing any malicious activities? Applications are deployed at the DG, so it should be its responsibility to ensure application deployment and verification, user authentication, and result protection, but the bridge is the entry point for the pool, thus it needs to trust the dispatcher of tasks, too. Allowing foreign code to run on a computer always has a risk of either accidental or intended misbehavior. BOINC mitigates this risk by only allowing to run code that has been digitally signed by the project the client (in this case the bridge) is connected to. Clients trust the operators of the BOINC project not to offer malicious code, and digitally signing the application provides technical means to ensure this trust relation. Of course it is not enough to only sign the application binary, the input data must be signed as well (think of the case when the application is some kind of interpreter and the input data can instruct it to do just about anything). Therefore BOINC uses two separate key pairs: one is used to sign the work units (which in this context means the set of input files and a link to the application binary) and the other is used to sign the application code. The private key used for work unit signing is usually present on the project's central server, while the private key used for application signing is usually kept at a separate location.

The different handling of the private keys stems from their usage pattern: the work unit signing key is used very often while the code signing key is normally seldom needed and therefore it can be protected better. This implies that trust is on a per DG basis, since any application deployed there is signed using the same key. In the case of the second scenario deploying applications to be run either becomes a security risk or an overhead. Overhead when there is limited access to the application signing key; risk when there is wide access to it.

One of the enhancements of BOINC by SZTAKI Desktop Grid is a new security model which replaces the current one of BOINC with a certificate enabled system [13]. This removes among many the limitation of the per project-based trust. Trust is now based per application developer (AD) with optional signatures by the DG project itself. AD is an entity or group of entities who is trusted by the DG to deploy applications. Clients may accept applications either from only the ADs the DG is trusting or from the ADs the administrator of the client is trusting. It is done by deploying the certificates of the trusted entities at the client, ensuring that applications are verified both by the DG server and by the client.

Communication and data transfer between the client and the SZTAKI LDG server are performed via HTTPS; this ensures protection of the confidentiality of data. Communication is always initiated by the client, and this and the HTTPS protocol allow traversal of firewalls. A server certificate ensures that the client is in fact talking to the server it thinks it is talking to and prohibits unauthorized access.

We are convinced that grid resources should be considered safe resources, thus results returned to the DG should be considered valid, but the standard BOINC protection against fraud, task replication to different pools and result comparison can be applied here too. Another aspect is to isolate the execution environment (host) from the executed (malicious) application. Since tasks are executed on a worker node of a cluster (without the DG client execution environment), it is the responsibility of the node to provide the required level of isolation for the task, although there is an ongoing work for SZDG to provide sandboxed application execution on the DG clients using virtualization techniques.

3.2 Application Deployment

In the case of the first scenario tasks of the application already deployed at the DG are sent to its clients. DGs usually require applications to be modified in order to be able to run on their clients, or in this case the already modified applications are needed to be able to run, without their normal execution environment, as Condor tasks. “BOINCified” applications, ported to the BOINC platform, can detect whether they are run without the BOINC execution environment. They can fallback to a *standalone* mode, which allows running them as ordinary applications. BOINC supports application-level (periodic) checkpointing, the application asks whether it is time to checkpoint, and periodically a specific checkpointing function implemented by the application is invoked. In the case of standalone mode the check

for time for checkpointing will always return yes, causing to checkpoint more often. This allows them to resume from checkpoint without any modification or third party tool.

In the case of the second scenario applications are deployed and distributed via the DG into the Condor pools. Since there are no other clients besides the connected Condor pools the applications do not need to be modified and may use Condor services like checkpointing, although this requires recompiling them before deployment.

3.3 Flow

Our bridge is a modified BOINC core client which enables to submit work units to Condor clusters. It is done as follows (see Fig. 12.1). The bridge needs to be deployed at a *Submit Machine* of the pool, which is usually the front-end node. Normally the client performs benchmarking on startup to have a guess about the available resources such as number and performance of CPUs, available memory and disk space. The bridge instead is using values defined by the administrator for the resources allowing to dedicate only part of the pool to the DG (it will always submit one task per available CPU to the pool).

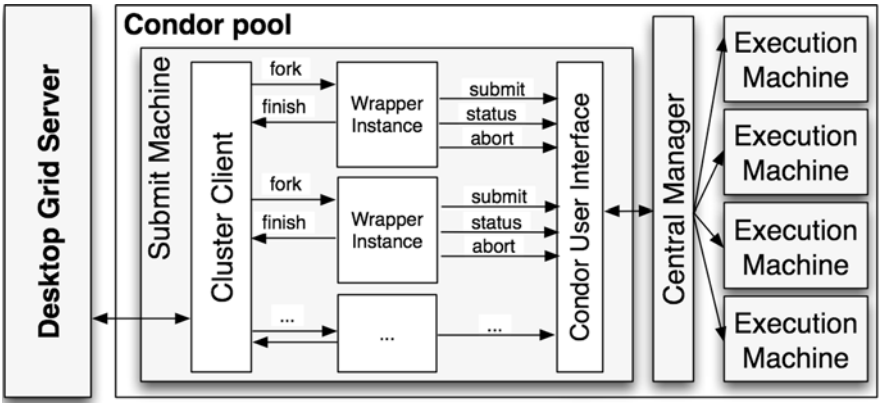


Fig. 12.1 SZTAKI Desktop Grid – Condor bridge

Once started the bridge will connect to a chosen DG server to download application executable (if it was not done yet) and work units. For each work unit a *wrapper* is executed creating a wrapper instance; its task is to *prepare the work unit*, *submit* it to the pool, *monitor* its status, and *retrieve* the output. The instance is running until the result from the pool is retrieved. The client has no direct information about the status of the tasks being executed at the pool; instead, the wrapper instance is representing the task for the client.

Work unit preparation consists of collecting the input files to a sandbox and creating a task description file (*submit file*) for Condor. The wrapper allows to use a template for the submit file to be created, thus allowing to fine tune the execution requirements, like which machines to exclude from the execution and which time of day it is allowed to run. The template is used for all tasks submitted providing a way to control execution on the nodes without having to modify the existing configuration and to prioritize tasks.

Submission is done using the Condor interface on the submit machine. This means the bridge is using a well-defined interface and not relying on the internal structure of the pool. Basically the pool is a black box for the bridge and the *execution machines* do not need to have in/outbound Internet connection or any connection with it. If the bridge is instructed (by the administrator) to *abort* a task, it will remove it from the pool via the interface and report it to the DG as failed.

Monitoring is done by polling the information provided by the cluster management about the job. Condor writes information about the changes of the job status into a log file. The wrapper uses the Condor API to parse this log file and checks for new events; this way, it keeps track of the status of each task.

When a job finishes (whether success or failure) the wrapper picks up the exit code from the log record, retrieves the output, and terminates. The client only notices after this that the task has finished, contacts the DG server, uploads the output files, reports the status of the task (either finished or failed, depending on the exit status of the wrapper), and requests new work.

3.4 Fault Tolerance

Since several components interact, it is crucial that each component and the combined system as well are fault tolerant. The purpose is to enable the system to restart properly after failure, but we do not consider hardware failures here, since those have to be solved at a different level.

The DG server stores application binaries and work unit input files on the filesystem, while the metadata resides in a database. The database keeps track of the status of any work unit. Each work unit in progress has a deadline set for completion; when it passes, the status of the work unit is reset to the initial state. This prohibits that any work unit can be lost by clients, meaning that in the worst case a work unit is resent to a client (or another client) when the deadline passes and no result was yet reported.

When a DG client requests work, it gets a description of the work and the locations of the application binaries and input files required to execute the task. This way if a file is missing it can be retrieved later anytime from the data server of the DG. The client periodically writes its state to disk, the state files are rotated ensuring that the client can resume anytime from a recent saved state. The bridge stores the status of the task in a separate file for each task. This guarantees that a running task is never re-submitted or lost from a client and a not submitted task will be submitted upon restart from a failure.

Tasks running in the pool are managed by Condor, sometimes they need to be suspended or migrated to another node. In the case of the second scenario applications can use the checkpointing services of Condor (when running in the *standard* universe), thus they are checkpointed before migrating to another node, or they can resume from a checkpoint when restarting from a failure. In the case of the first scenario when "BOINCified" applications are run on the cluster, checkpointing is done at application level. The checkpoint file is just another output file of the task (but it is read for input when the task restarts), thus it is migrated along with the task to the node where the computation can resume. In the case of failure the task can resume from the last checkpoint file.

3.5 Future Work

Our approach provides a lightweight solution, but there are ways to improve it. Currently tasks are submitted per work unit basis; when there are many short jobs this approach is ineffective since the overhead of time until a task is scheduled to start could be comparable with its execution time. A better solution is to batch short jobs together and submit them as a larger unit. Determining how large a task should be requires information both from the pool (available CPUs, performance of CPUs) and from the DG (estimated runtime of a single work unit). Input data and binaries for the tasks are fetched from a single data server, thus creating a possible bottleneck; there is ongoing research to adopt peer-to-peer techniques for data distribution for DGs [8], by adopting these methods data could be fetched from several sources. Currently each wrapper instance is now autonomous, meaning it is responsible for a single task and queries the Condor infrastructure for a single task, too. A better approach is to use a sole agent who is performing actions (submit, monitor, abort, and retrieve) for all tasks. We are currently using BOINC and Condor, but the bridge could be generalized to connect desktop grids with arbitrary service grids (SGs) and should be bidirectional.

A new FP7 infrastructures project has started in January 2008, entitled Enabling Desktop Grids for e-Science (EDGeS) [4], that is building technological bridges to facilitate service and desktop grid interoperability. EDGeS is attempting to close the gap between DG and SG computing. The future work described here is a tiny subset of the goals of EDGeS and will be done in the frame of this project.

4 Applications

Several projects are actively using SZTAKI Desktop Grid. One of the fundamental tasks in modern drug research is the exclusion of chemically unstable, biologically inactive or toxic compounds from the research process in the early stages, thereby reducing the cost and the time period of the drug development. The main purpose of the ADMEToxGrid [1] and CancerGrid [9] projects is to develop an enterprise grid system that is suitable for predicting these chemical parameters of millions of

compounds in a short time and in a secure manner, while also exploiting the free capacity of the office computers located at the different sites of the company. In this project SZTAKI Desktop Grid serves as the base of the grid framework and was extended with additional features such as the cluster support, which is currently being deployed to connect (cluster) resources from project partners.

SZTAKI also runs a public BOINC project also known by the name SZTAKI Desktop Grid (SZDG) [18]. The original intention of SZDG was to serve demonstration purposes among scientists mainly in Hungary and also worldwide to prove that it is not necessary to have expensive hardware and truly monumental aims to catch the attention of the voluntary public. Soon after starting SZDG in early summer of 2005, the Computer Algebra Department of the Eötvös Loránd University applied for the project with their already developed and running single-threaded program: project BinSYS [6], which was modified to run on SZDG. The goal of BinSYS was to determine all of the binary number systems up to the dimension of 11. The difficulty in this is that the number of possible number-system bases explodes with the rising of the dimension. The input of the program is a huge, but finite part of the number space and the output is a bunch of matrices, or more precisely their characteristic polynomials fulfilling certain criteria. Further narrowing the criteria on these selected matrices, the resulting ones make up the generalized binary number systems of the given dimension. Knowing the complete list of these number systems the next step is to further analyze from the view of information theory. Sketching the integer vector of the vector space in the usual way and in the generalized number system, their form can greatly vary in length. Also the binary form of vectors close to each other can vary on a broad scale. With this in mind the research will continue further with the use of number systems in data compression and cryptography. The assumption was that the program will be able to handle the dimensions up to 11 on a few computers and by the time the cooperation has been started it has already finished up to dimension 9. The prediction has assumed that the processing of dimension 10 will last for about a year, yet it has been successfully finished by the end of the year 2005 with the help of few thousand computers of volunteers joining the project. After this success the application has been further developed making it able to handle dimensions higher than 11 and also to break the barriers of the binary world and process number systems with a base higher than 2. The bridge is used here to connect dedicated resources from SZTAKI and University of Miskolc to SZDG (*first scenario*).

5 Conclusion

There are several existing solutions for connecting DGs and clusters; what they have in common is that they connect the resources of a cluster to a DG individually, thus generating a deployment and management overhead. This way the nodes of the cluster need (outbound) Internet access, which is sometimes inadequate. Also the connection to the DG is temporary or low priority, meaning that dedicating these resources to DGs becomes difficult.

Our approach allows to connect clusters as a single large resource to DGs with little effort and respect for the local policies. Only a single bridging client is required at the front end of the cluster. It is allowed to dedicate only part of the cluster and for only selected periods of time for the DG. When needed an improved security infrastructure can be used, which allows secure communication and data transfer and allows to verify and accept only selected tasks to be run on the cluster. No in/outbound Internet access is required for the worker nodes of the cluster. We presented our plans for future work and two currently running projects which utilize clusters as DG resources using our solution. The bridge package is available for download [18].

Acknowledgments The research and development published in this chapter is supported by the European Commission under contract numbers LSHC-CT-2006-037559 (FP6 STREP, CancerGrid), IST-2002-004265 (FP6 NoE, CoreGRID), and RI-211727 (FPF7 IST Capacities, EDGeS).

References

1. ADMEToxGrid. `http://www.admetoxgrid.hu/`
2. D.P. Anderson. BOINC: A system for public-resource computing and storage. In R. Buyya, editor, *Fifth IEEE/ACM International Workshop on Grid Computing*, pp. 4–10. IEEE Computer Society, 2004.
3. D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, November 2002.
4. Z. Balaton, Z. Farkas, G. Gombás, P. Kacsuk, R. Lovas, A.C. Marosi, A. Emmen, G. Terstyánszky, T. Kiss, I. Kelley, I. Taylor, O. Lodygensky, M. Cardenas-Montes, G. Fedak, and F. Araujo. EDGeS: The common boundary between service and desktop grids. Accepted for publication at the CoreGRID Integration Workshop 2008, Hersonissos-Crete, Greece, 2008.
5. Z. Balaton, G. Gombás, P. Kacsuk, Á. Kornafeld, A.C. Marosi, G. Vida, N. Podhorszki, and T. Kiss. SZTAKI desktop grid: A modular and scalable way of building large computing grids. In *Workshop on Large-Scale and Volatile Desktop Grids, PCGrid 2007*, 2007.
6. Project BinSYS. `http://compalg.inf.elte.hu/projects/binsys/`
7. F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Neri, and O. Lodygensky. Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems*, 21(3): 417–437, 2005.
8. F. Costa, L. Silva, I. Kelley, and G. Fedak. Optimizing the data distribution layer of BOINC with BitTorrent. Accepted for publication at the Second Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid), Miami, USA, 2008.
9. Z. Farkas, R. Lovas, and P. Kacsuk. CancerGrid: Enterprise desktop grid solution with workflow support for anti-cancer drug design. *Proceedings of the Cracow Grid Workshop 2007*, 2008.
10. P. Kacsuk, N. Podhorszki, and T. Kiss. Scalable desktop grid system. *High Performance Computing for Computational Science – VECPAR 2006*, pp. 27–38, 2007.
11. LHC@home. `http://lhcathome.cern.ch/lhcathome/`
12. O. Lodygensky, G. Fedak, F. Cappello, V. Neri, M. Livny, and D. Thain. XtremWeb & Condor : sharing resources between Internet connected Condor pool.*Cluster Computing and*

the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on, pp. 382–389, 2003.
13. A.C. Marosi, G. Gombas, Z. Balaton, P. Kacsuk, and T. Kiss. SZTAKI desktop grid: Building a scalable, secure platform for desktop grid computing. CoreGRID Workshop on Grid Programming Model Grid and P2P Systems Architecture Grid Systems, Tools and Environments, Heraklion – Crete, Greece, June 2007.
14. C. O'Donnell. Condor and firewalls. University of Wisconsin-Madison. `http://www.cs.wisc.edu/condor/`, 2002.
15. J. G. Pedersen and C. U. Sottrup. Developing Distributed Computing Solutions Combining Grid Computing and Public Computing. M.sc., University of Copenhagen, 2005.
16. S. Son and M. Livny. Recovering internet symmetry in distributed computing. In Proceedings of the 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, May 2003.
17. Superlink@EGEE. `http://cbl-link02.cs.technion.ac.il/cplan/`
18. SZTAKI Desktop Grid. `http://www.desktopgrid.hu/`
19. D. Thain, T. Tannenbaum, and M. Livny. Condor and the grid. In F. Berman, A. Hey, and G. Fox, editors, *Grid Computing – Making the Global Infrastructure a Reality*, Chapter 11. John-Wiley & Sons, Ltd., 2003.
20. D. Wright. BOINC and Condor – Scavenging the Scavenger. 2nd Pan-Galactic BOINC Workshop, 2006.

Chapter 13
SoRTGrid: A Grid Framework Compliant with Soft Real-Time Requirements

A. Merlo, A. Clematis, A. Corana, D. D'Agostino, V. Gianuzzi, and A. Quarati

Abstract Current grid resource management tools provide sufficient support for jobs executed in batch or "best-effort" modes, but are not adequate for jobs with demanding quality of service requirements, like jobs with time constraints. For such class of applications, missing the required deadline can have unpredictable consequences, from loss of quality to inconsistent results. In this chapter we present SoRTGrid, a grid framework that provides a set of services and features to discover and manage resources for soft real-time jobs. SoRTGrid is based on a direct comparison of the application requirements with the available resource offers and relies on a distributed Bid repository coupled with an overlay grid architecture.

Keywords Grid computing · Soft real time · Quality of service

1 Introduction

Grid computing [15] is becoming more and more a powerful paradigm to manage complex jobs, using different virtual resources (e.g., CPU, memory, data, instrumentation, and applications) from all over the world. Grid computing grants the execution of a large number of jobs whose requirements can overcome the availability of a single organization. However, the grid paradigm provides sufficient support for jobs executed in batch or "best-effort" mode, while it neither offers enough control nor provides adequate level of guarantees on applications with demanding quality of service (QoS) requirements, like jobs with time constraints. Unfortunately, QoS on the grid is a harder issue with respect to other infrastructures used in distributed systems, such as the network infrastructure: in fact, the large number of resources that belong to different owners and domains, the *virtualization* provided by the layered grid architecture [14], and the great *dynamism* (e.g., the whole set of

A. Merlo (✉)
DISI – Università degli Studi di Genova, Via Dodecaneso 35, 16146 Genova, Italy
e-mail: merlo@disi.unige.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_13,

resources changes continuously) complicate the approach to the problem, making hard to obtain a single, general solution.

Since general-purpose grid middlewares do not provide sufficient functionalities to manage jobs with deadlines, they must be improved with overlay architectures and proper mechanisms to become suitable also for this class of applications.

For this reason, we present here SoRTGrid (soft real-time grid), a grid framework that offers services for an efficient resource discovery and selection, aimed to the execution of *deterministic* jobs with *time constraints*, from strict soft real-time jobs to "best-effort" ones with loose time boundaries.

A deterministic job is characterized by a well-defined behavior that allows to appraise an upper bound in its execution or response time on a given set of resources. In this sense, different metrics are used to evaluate this bound, like number of instructions, MiPS or BogoMips. In our context, it does not matter how to measure it, but it suffices to assume that, in some way, an evaluation of the maximum execution time can be calculated and used in SoRTGrid.

A job with time constraints is such that the missing of a deadline can have different consequences, from a little loss of quality to a total invalidation of the obtained results. Dealing with soft real-time jobs we suppose that a small fraction of them are allowed to miss the deadline; the aim of SoRTGrid is to keep the miss rate as low as possible.

In general, there are many applications that require the acquisition and processing of huge quantities of data from distributed sources and fast response to particular events. For instance, distributed data acquisition through sensor networks and remote control of instruments and devices.

In this chapter a general introduction to SoRTGrid is given: more specifically, we present the motivations of our work, the architecture, the main entities involved, and the resource discovery process.

The chapter is organized as follows: in Section 2 we briefly sum up some of the most prominent works in grid QoS, while Section 3 motivates SoRTGrid and describes its main features. Section 4 presents the architecture of SoRTGrid, and Section 5 focuses on the resource discovery process. Finally, Section 6 discusses some possible problems on the negotiation and production of bids.

2 Related works

QoS in grid computing is a different issue than in various distributed environments like the Internet [25] or ad hoc networks [4] since, in such cases, less variables have to be considered. Likewise, also the soft real-time problem on grid is trickier to be defined and tackled than in an embedded or an integrated system [20].

Grid computing has an inter-domain, virtualized nature, and evolved in its implementation standards. At present, grid computing is modeled as a service-oriented architecture [10], compliant with the OGSA standard [13]. Previous general QoS grid frameworks like GARA (Globus Architecture for Reservation and Allocation)

[16] are not suitable for current needs of QoS, since they are neither OGSA compliant nor allow simultaneous allocations. This second aspect can be managed through service level agreement (SLA) [21] and negotiation algorithms [2]. In this sense, many projects have been recently developed on the current grid computing standards.

Al-Ali et al. have proposed G-QoSM (grid QoS management) [1], a framework that grants SLA negotiation and QoS at computational, storage, and network level. MG-QoS [23] constitutes another example of SLA-based and service-oriented QoS framework, but applied in a well-defined grid environment.

Side by side with QoS frameworks, proper systems to manage and negotiate SLAs have been designed, like GRUBER [9] and its distributed version [8] that we have taken into account in our SLA negotiation phase.

The economic grid presented in [22] and in other related papers is an interesting proposal that considers the time reservation for a cluster, performed through SLAs where users can define one out of two optimization parameters: either the response time or the budget to spend. The negotiation is performed through a manager agent that queries the shared directory (possibly distributed over a peer-to-peer network) to obtain information required to contact the owner of the resource that matches the user-specified SLA parameters.

In [19] a syntax for the specification of resources (Rspec) and a set of rules for their usage is provided.

We have also examined grid metaschedulers like Condor-G [18] or Nimrod/G [3] as a way to *matchmake* virtual resources and grid users; unfortunately, such systems have accurate mechanisms neither to manage deadlines nor to appraise an "upper bound" for the time required to find a suitable set of resources for the requesting job.

SoRTGrid has been designed taking into account the previous approaches, namely the aforementioned economic grid, with some substantial changes, as described in the following, with the goal of deploying a grid framework that can efficiently support different classes of QoS for time constrained jobs.

3 SoRTGrid

A framework aimed to be compliant with soft real-time requirements must have the following features: (1) a resource discovery time as fast as possible; (2) mechanisms to assure the availability of the required resources for the whole requested time interval; and (3) a mechanism of pre-reservation of resources to avoid inconsistencies during the negotiation phase. SoRTGrid tries to meet such requirements with a direct interaction between users and resource owners. SoRTGrid constitutes a sort of facilitator between users and resource owners, a black box that provides different services allowing the efficient management of jobs with deadlines. In our context, a user is a grid user that needs to execute a job with time constraints, while a resource owner is someone who can make resources that he controls directly available.

Both parts need some abilities to interact efficiently with SoRTGrid. The grid user must be able to evaluate the length of his deterministic jobs so that the choice among the different resources could correctly take to a complete execution of the job within the requested time limit. The resource owner must control the resources he shares directly, without giving the control to any grid metascheduler. In this way, the non-deterministic latency due to the metascheduler system and possible inconsistencies in the state of resources are avoided.

Users and owners are represented in SoRTGrid by proper autonomic agents (*user and owner agents*, respectively). Physical actors can interact with such entities to provide decisional policies or constraints, but are directly involved neither in the offering, discovery, selection, and negotiation of resources nor in execution activities. Discovery and selection of resources are carried out through the interaction of user and owner agents with SoRTGrid, while the subsequent negotiation and job execution are performed by direct interactions between such agents.

SoRTGrid makes available to owner and user agents a distributed shared data structure (repository), similar in some aspects to the coordination model paradigm [24].

The owner agent freely chooses which resources to offer and under which conditions; it produces a number of resource proposals (in the following SoRT-Bids) and writes them into the repository. The user agent provides a list of job requirements, the number of required SoRT-Bids, and a maximum time (*maxtime*) for the discovery activity.

SoRTGrid searches the published SoRT-Bids in the distributed repository and returns to the user agent a set of SoRT-Bids matching the requirements. The discovery activity ends within maxtime. The user agent chooses a subset of the retrieved SoRT-Bids, contacts the respective owner agents to negotiate them within the prereservation time, and finally execute its job on negotiated resources.

Moreover, SoRTGrid sends back to the owner agent feedbacks on the demand of its SoRT-Bids so that it can adapt its policies for the proposal of further SoRT-Bids to the user agent requests.

The architecture of SoRTGrid provides the following benefits:

- *Autonomy*. The two parts involved (users and owners) are independent in their actions and pursue different objectives. Users need to find resources that could grant them that a job submitted to the grid could be correctly executed within the required deadline; owners require to have an efficient use of their resources under the terms and conditions they choose.
- *Adaptability*. The behavior of SoRTGrid depends on a set of parameters (e.g., the grid size, the kind of resource requests, and proposals, respectively, from the grid user and resource owner side, allowed search time, and number of neighbors). The appropriate evaluation of such parameters allows SoRTGrid to adapt to different grid contexts.
- *Consistency*. Since SoRTGrid does not rely on metaschedulers but is based on a direct interaction between owner and user agents, it provides up-to-date information on resources that are effectively available. Moreover, it implements

functionalities assuring that the provided information remains valid for a time interval sufficient to perform negotiation (timed pre-reservation) and that the same resources are not offered simultaneously to two different users (no over-booking).

- *Automation*. The use of agents eliminates direct interaction with human users from both resource and grid user side. In this way a high level of automation of the parts involved is obtained and interaction time is reduced, according with the time-critical goal of SoRTGrid.

4 Architecture of SoRTGrid

SoRTGrid is an overlay grid network composed by a set of connected facilitator agents (Fig. 13.1) whose aim is to grant the required services both to the user and to the owner agents. Every facilitator agent is composed by two grid services [6] (BidMan, DiPe) (Fig. 13.2) and interacts with the grid middleware (e.g., Globus Toolkit 4 [11]) to obtain basic grid functionalities.

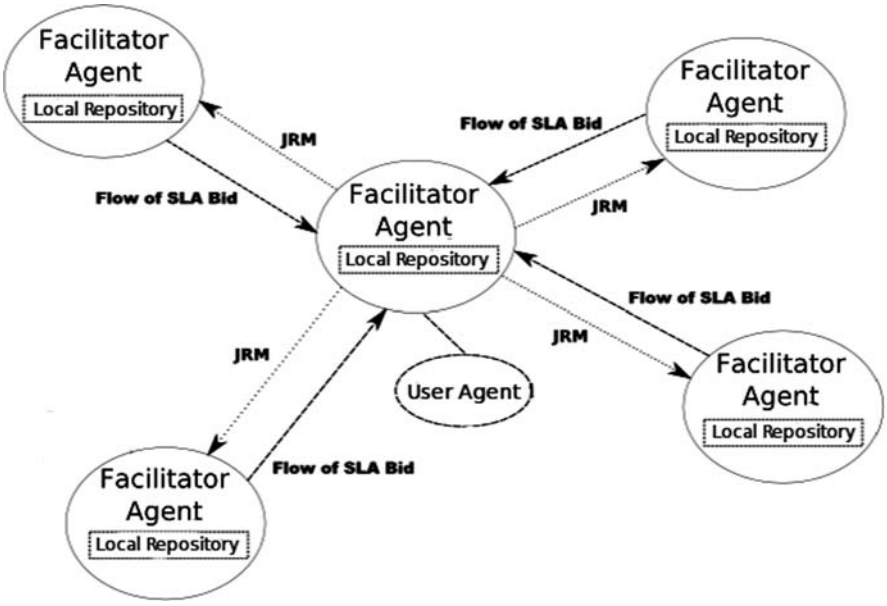


Fig. 13.1 SoRTGrid as a set of interacting facilitator agents

Virtual resources of the grid are divided among different facilitator agents and every resource refers to a single facilitator agent only.

Every facilitator agent can work with every taxonomy of virtual resources. As a resource scenario example, we adopt here a classification we used in previous works [5], where resources are divided into single machines, homogenous clusters, and heterogeneous clusters (Fig. 13.2).

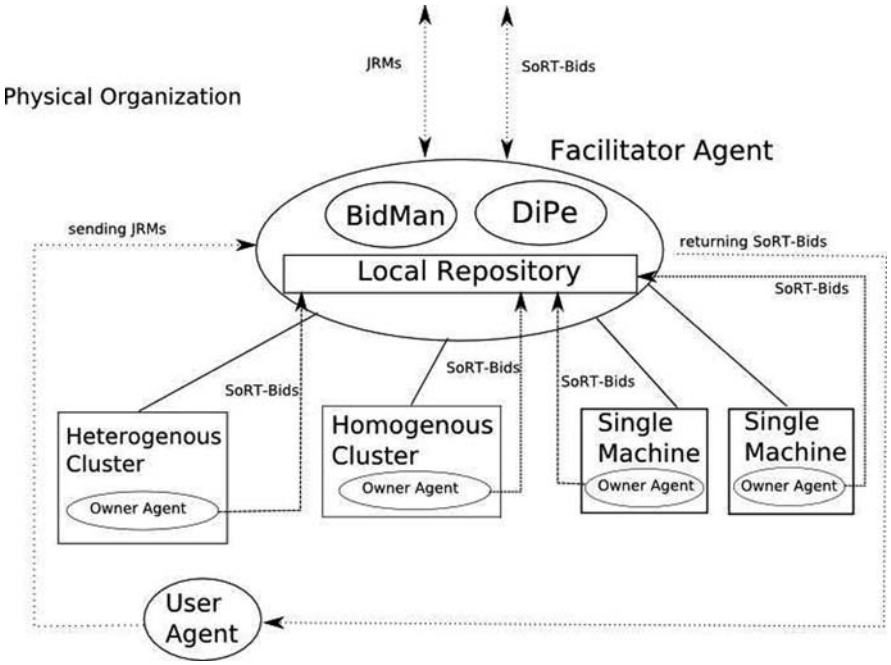


Fig. 13.2 The facilitator agent composed by BidMan and DiPe grid services. It manages resource queries from user agents and the access to the local repository

We consider the grid as constituted by the union of several sub-portions, each of which comprises only resources within an administrative domain (physical organization (PO)).

In the following, we assume, for the sake of simplicity, that there is a facilitator agent for every PO in SoRTGrid and that the whole set of resources managed by an owner agent is related to a unique facilitator agent.

4.1 SoRT-Bids and Owner Agents

The aim of SoRTGrid is to provide sufficient support for jobs with time constraints. A key point is that a selected resource would be effectively available when the job needs to execute. For this reason, approaches relying on a shared middleware scheduler that controls virtual resources are unsuitable since there is not direct control on resources at lower level that remain controlled by the local scheduler belonging to the physical organization. Indeed, since the owner manages the access to resources directly, the idleness of a certain virtual resource at the grid level could not match with an effective availability at the physical level.

SoRTGrid resolves this possible inconsistency involving the resource owner and giving him the possibility to directly offer his resources to grid users through a

proper extension and customization of the mechanism of SLA Bids [22] and the SoRT-Bids. In the SLA-Bid approach, every owner decides in which way and how long to offer its resources and publishes a number of SoRT-Bids. A SoRT-Bid contains quantitative and qualitative information about one or more resources (number and frequency of processors, RAM and disk space, etc.). The set of resources is offered until an expiration time and for a time interval ΔT during which the execution of the job is guaranteed. The terms in which ΔT is offered define the *quality of service level* provided by the SoRT-Bid.

We choose to define and support different classes of QoS for different kinds of jobs. Our aim is to provide a maximum guarantee level for jobs with critical deadlines and a lower level for those executions where missing a deadline does not imply serious consequences.

At present, SoRTGrid supports three different classes of QoS:

1. *Soft real-time class*. The higher level of QoS; it guarantees the total availability of the resources for ΔT and no interruption of the job execution. This class is particularly suitable for interactive jobs.
2. *High level class*. The ΔT execution time is granted but the job can be interrupted. It is suitable for generic jobs with deadlines but not for interactive ones since the response time cannot be granted.
3. *Standard class*. The ΔT execution time is granted only if jobs with higher QoS level do not run on the same resources. It is suitable for generic jobs without time constraints or such that the violation of a deadline does not take to serious consequences.

In our approach, the operations at the owner's side regarding the definition and the publication of SoRT-Bids are performed by an appropriate process acting in the owner's stead called owner agent. This can be an intelligent agent [17] or a proper application that implements logical rules or heuristics to create SoRT-Bids.

The owner agent has a dual interface, to the resource owner and to SoRTGrid (Fig. 13.2). On the one hand, the owner agent accepts policies and parameters to adapt its decision model to the will of the resource owner. On the other, it interacts with the BidMan service of the facilitator agent the resource belongs to; in this way it can publish and remove SoRT-Bids.

The use of the owner agent entity allows the automation of the SoRT-Bids production process, so that it could be quick, correct and independent of grid user's behavior.

4.2 Facilitator Agent: BidMan Service

SoRT-Bids produced by owner agents must have visibility inside the physical organization and outside to the whole grid. For this, the facilitator agent has to receive and manage the set of SoRT-Bids referring to the resources it controls.

From our previous experience in deploying GEDA (grid explorer for distributed applications) [7], we developed BidMan, a grid service for the management of SoRT-Bids, which constitutes one of the two grid services composing the facilitator agent.

BidMan creates and interacts with a shared repository (in the following, local repository) in which SoRT-Bids are contained. It allows an owner agent to upload new SoRT-Bids and to cancel previously uploaded ones. Moreover, it performs maintenance operations on the resident SoRT-Bids (i.e., removal of expired, canceled, or successfully negotiated bids). BidMan is the only grid service that has the rights to write on the local repository and that interacts with owners agents.

At this level, we do not provide any detail on the implementation of the local repository; for the focus of the article, it suffices to assume that it is addressable and accessible by the whole set of facilitator agents composing SoRTGrid.

Functionalities of BidMan

BidMan has been designed as a *singleton* grid service. This means that there is a single instance of BidMan that receives the offers from all the owner agents of the physical organization (Fig. 13.2). Every owner agent performs its task through a set of functionalities offered by BidMan service:

- *SoRT-Bids upload*. It allows an owner agent to upload SoRT-Bids to the local repository. It returns a *unique ID* assigned by BidMan to every correctly uploaded SoRT-Bid. The ID is valid as long as the bid expires.
- *SoRT-Bids removal*. It allows to remove a not yet expired SoRT-Bid from the local repository. The removal is possible only if the bid is not being negotiated or reserved due to a previous discovery query. Otherwise, an error has to be reported to the owner agent.
- *Local repository localization*. It returns the location of the local repository to the DiPe grid service for reading operations.
- *Local repository state*. It returns up-to-date statistics on the local repository state like number of resident SoRT-Bids, pre-selected SoRT-Bids, free SoRT-Bids, and so on.

Besides the public functionalities, BidMan performs maintenance activities on the local repository like removing expired SoRT-Bids and freeing their *unique ID*.

4.3 User Agents and Job Requirements Manifests

Every grid user is represented in SoRTGrid by a well-defined process called user agent.

The user agent belongs to the facilitator agent to which belongs the respective grid user. It receives and manages user's job requests, performing three functionalities in user's stead:

- *Job requirements appraisal*. The user agent receives jobs to execute and deadlines from the grid user. From such information, it must be able to evaluate job

execution time and on which resources; then it creates a proper requirements document (job requirements manifest in the following JRM).
- *Resource discovery and selection*. It uses the JRM to interact with facilitator agents and retrieves a set of suitable resources.
- *Job negotiation and execution*. It selects some resources from those returned by the previous phase and negotiates directly with the respective owner agents. After a successful negotiation, it executes the job on such resources under the terms and conditions negotiated.

Because of the activity performed by the user agent, the grid user is not directly involved in requirements definition, resource selection, and negotiation, and the whole activity is automated, perfectly according to grid philosophy [12].

Let us now describe the JRM document; it is composed of various fields:

- *Resource requirements*. Quantitative and qualitative information about requested resources.
- *QoS Class*. The class of required QoS and related parameters.
- *Expiration Threshold*. Resource discovery will consider only SoRT-Bids with an expiration subsequent to this value. In fact, the user is only interested in SoRT-Bids available at the time T_{subm} when the job will effectively need to be submitted, so the threshold must be greater than or equal to T_{subm}.
- *Appraisal Threshold*. During the discovery phase, SoRT-Bids are evaluated through a *utility function* that provides an *appraisal value* on the suitability of the SoRT-Bid for that JRM. The larger is the appraisal value, the more suitable is the SoRT-Bid. The appraisal threshold indicates the minimum appraisal such that the Bid could be considered *valid* for the JRM.
- *Utility function parameters*. Values that provide *weights* for different variables in the utility function used to appraise a SoRT-Bid. Depending on job nature and user's requests, some characteristics of the SoRT-Bid can be considered more important than others.

JRMs are sent by the user agent to the facilitator agent to perform a discovery of suitable SoRT-Bids. Differently from common grid resource discovery, in this case time is the main variable to take into account.

5 Resource Discovery in SoRTGrid

The aim of resource discovery in SoRTGrid is to retrieve, within a given discovery time, a prefixed number (N_{Bids}) of SoRT-Bids with an appraisal value equal to or greater than the appraisal threshold in the JRM.

The greater the N_{Bids} value, the longer will result the time required for the discovery and more SoRT-Bids will be potentially returned; on the contrary, a lower value grants a quicker discovery and fewer results. Generally, the choice of the N_{Bids} value depends on the deadline of jobs that the user agent needs to execute but can also depend on particular policies of the user agent itself. Because of the criticality

of time in the discovery process, the aim of the discovery cannot be the retrieval of the best N_{Bids} SoRT-Bids in the whole SoRTGrid; on the contrary, the final target is to find the first N_{Bids} suitable SoRT-Bids (i.e., fulfilling the JRM requirements), interacting with as few as possible outer facilitator agents.

For the user agent, the discovery activity must be as quick and efficient as possible, while, on the SoRTGrid hand, it has to produce a reduced number of communications between facilitator agents.

Moreover, resource discovery must be modeled to allow the retrieval of the desired number of SoRT-Bids within the required discovery time limit. We propose here a discovery algorithm aimed to take into account all the previous considerations.

The algorithm comprises two phases, namely *local discovery* and *remote discovery*. The second phase is performed only if the first one does not reach a satisfying result. Resource discovery is performed by a single grid service (DiPe) that, together with BidMan, constitutes the facilitator agent (Fig. 13.3).

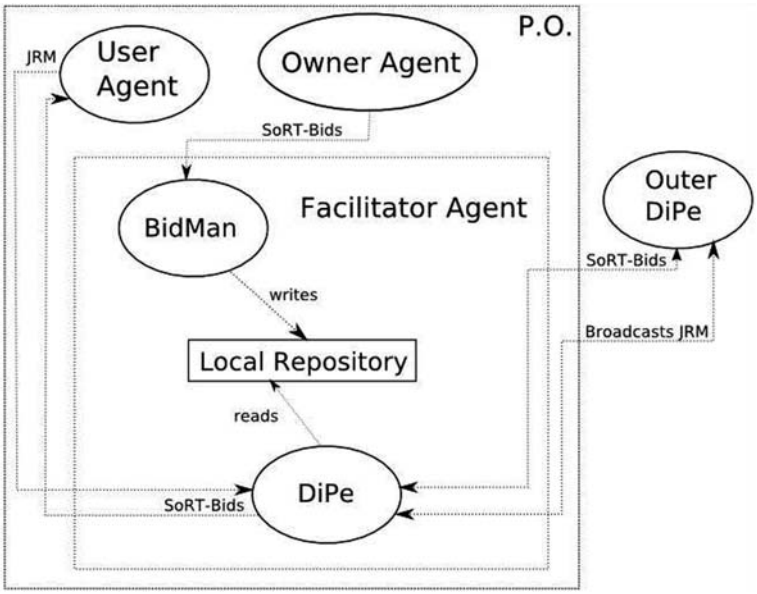


Fig. 13.3 Interactions in SoRTGrid

5.1 Facilitator Agent: DiPe Grid Service

DiPe – that stands for discovery performer – is a *factory* grid service that after a user agent request starts the local discovery and, if necessary, the remote discovery. The factory nature of DiPe implies that a new instance of DiPe is created at every

request and is dedicated to that query resolution. A DiPe instance is created due to a call from a user agent or from another DiPe (Fig. 13.3).

A user agent request is composed by three parts: the JRM, the number of required SoRT-Bids that must satisfy the JRM (N_{Bids}), and the time limit within which the discovery activity must be completed (*maxtime*).

Following a user agent request, the new instance of DiPe performs first of all a local discovery on the local repository of SoRT-Bids. If the local discovery provides the required number of satisfying SoRT-Bids, the results are returned to the user agent and the DiPe instance terminates. Otherwise, a remote discovery is performed by the DiPe instance through an interaction with other DiPe services belonging to neighbor facilitator agents; a new instance is created for every DiPe service involved in the remote discovery.

Local and remote discovery will be presented in detail in the following. Here we present the functionalities of DiPe in order to define the functions and procedures that will be used in such discovery algorithms; the first functionality interfaces with the user agent, whereas the others provide interaction with outer DiPe instances.

- *SoRT-Bids discovery*. It allows a user agent to start a bid discovery submitting a user agent request. DiPe performs a discovery activity that ends when N_{Bids} *suitable* SoRT-Bids are retrieved or *maxtime* is reached. In both cases, a group of SoRT-Bids is returned to the user agent (the required number in the first case, a smaller number in the second one). After the delivery of the SoRT-Bids, the instance is destroyed.
- *Remote SoRT-Bids request*. It is used by an outer instance of DiPe to require the search of SoRT-Bids in the local repository. It requires the URI of the requesting DiPe and the JRM. Selected SoRT-Bids are returned to the requesting DiPe through the next functionality.
- *Remote SoRT-Bids delivery*. It is used by an outer DiPe instance to deliver the SoRT-Bids it found.

The system automatically divides the maximum search time (*maxtime*) into two portions, namely t_{loc} (the maximum time allowed for local discovery) and t_{rem} (the maximum time allowed for remote discovery). The way in which search time is divided can impact in SoRTGrid behavior; as a particular case we can also start local and remote search at the same time.

5.2 Local SoRT-Bid Discovery

Local discovery starts after a call from a user agent to the DiPe functionality *SoRT-Bids discovery*, creating a new local instance of DiPe that manages the whole discovery process.

The local instance interacts with the local repository, reads, and appraises every SoRT-Bid, defining incrementally a subset of SoRT-Bids whose appraisal value is

greater than or equal to the appraisal threshold in the JRM. This process terminates when all the SoRT-Bids have been checked or t_{loc} has passed.

Two roads can then be followed, depending on the number of suitable SoRT-Bids found in the local search:

- $N_{\text{loc}} \geq N_{\text{Bids}}$. The required number of SoRT-Bids is reached. N_{Bids} SoRT-Bids are returned to the user agent, the local DiPe instance and the discovery activity end. The returned SoRT-Bids are those with lower appraisal value. This choice depends on a grid optimization issue: every selected SoRT-Bid satisfies the JRM, so it is not a good choice to return the user agent resource proposals that exceed a lot the real requirements, since it will take to the negotiation of resources that will not be totally used.
- $N_{\text{loc}} < N_{\text{Bids}}$. The local discovery has not found enough suitable SoRT-Bids, so a remote discovery will be performed in the remaining time t_{rem}.

5.3 Remote SoRT-Bid Discovery

If the local discovery is not sufficient to satisfy the SoRT-Bids request of the user agent, then a discovery on other facilitator agents has to be performed. The remote discovery is started by the *local DiPe instance* (i.e., the one that interacts with the user agent) and involves a set of *remote DiPe instances* that answer the remote request. The algorithm works as follows:

- *Initialization*. $N_{\text{rem}} = N_{\text{Bids}} - N_{\text{loc}}$ is the amount of suitable SoRT-Bids the remote discovery has to retrieve. $t_{\text{rem}} = \text{maxtime} - t_{\text{loc}}$ is the remaining time for the remote discovery.
- *Local DiPe instance*. The local DiPe instance starts the remote discovery by using the *Remote SoRT-Bids request* functionality of *neighbor* Facilitator Agents. Afterward, it waits until N_{rem} Bids are delivered or t_{rem} passed, and then terminates.
- *Remote DiPe instance*. First of all, the DiPe grid service receiving the call checks if it has already answered a previous request coming from the requesting DiPe instance for the same JRM. If so, it discards the request. Otherwise, it creates a new DiPe instance that performs a local discovery and tries to send back the set of suitable SoRT-Bids it has found through the *remote SoRT-Bid delivery* functionality of the requesting DiPe instance. If the requesting instance still exists, the SoRT-Bids are correctly delivered and the DiPe instance *broadcasts* the request to its neighbors by calling their *remote SoRT-Bids request* functionality and sending the JRM it previously received. In this way a new remote discovery step is started. On the contrary, if the requesting DiPe instance is no longer available, the broadcast is not performed.
- *Termination*. The local DiPe instance that originated the remote discovery terminates when it receives the required number of SoRT-Bids (i.e., $\geq N_{\text{Bids}}$) or when t_{rem} expires. In the first case it returns back the first N_{Bids} with the lower appraisal

value to the user agent, whereas in the second case the whole set of SoRT-Bids retrieved till expiration is returned.

The main advantage of the remote discovery algorithm is that it follows a completely distributed approach in which every outer facilitator agent manages independently a local discovery and a SoRT-Bids delivery to the unique DiPe instance that originated the discovery activity. Such instance receives *concurrent flows of SoRT-Bids* from different portions (POs) of the SoRTGrid until the desired amount is retrieved.

Since the request is forwarded simultaneously to a number of neighbors, groups of local discoveries are performed in parallel, reducing the total discovery time. Moreover, the broadcast is forwarded to neighbors only if there is still need of SoRT-Bids (i.e., the request in the DiPe instance is still existing) and the same DiPe instance does not execute a second discovery for the same request; both these characteristics allow to avoid useless searches, improving the answer time.

Denoting with N_{FA} the number of facilitator agents, the number of steps in the remote discovery phase is bounded by $\log_{\mathrm{neigh}} N_{\mathrm{FA}}$, where neigh is the average number of neighbors of the facilitator agents.

5.4 Neighboring

As shown in the previous sections, SoRTGrid consists of an undirected graph (see Fig. 13.4) where nodes are facilitator agents and edges are neighboring relationships (i.e., if A and B are directly connected it means that A is neighbor of B and vice versa).

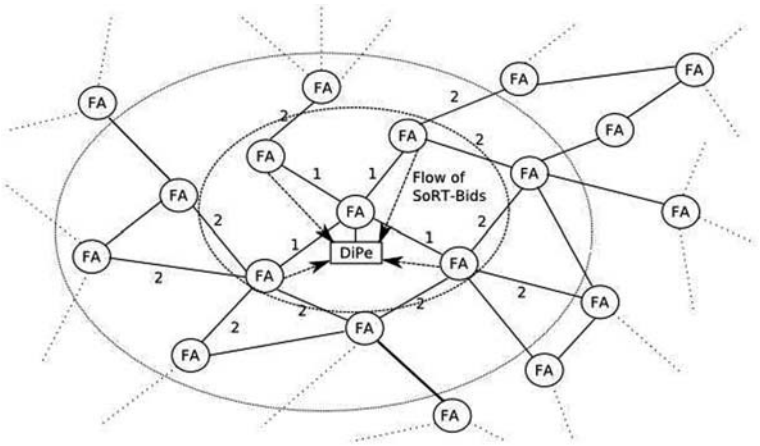


Fig. 13.4 A two-step remote discovery. The inner ellipse indicates the first broadcast and the outer one stands for the second step. A third broadcast is not forwarded because the requesting DiPe instance terminates before the end of the second step

Initially, SoRTGrid is composed by a single facilitator agent; afterward other facilitator agents can join SoRTGrid. Whenever a new facilitator agent enters or leaves SoRTGrid, a proper graph algorithm manages the topology, granting that the graph is completely connected.

In this chapter we do not take into account any particular graph algorithm for the definition of the set of neighbors; any Delaunay graph could be suitable, provided the previous constraints are satisfied for every modification of the topology.

In such a scenario, another requirement for the graph definition and maintenance is that information on existing facilitator agents in SoRTGrid can be retrieved. In this sense, the location of facilitator agents is provided by lower level middleware information services (e.g., monitoring and discovery system of globus toolkit 4 [11]).

For now, we do not assume any rule or filter to choose a node instead of another as neighbor (i.e., the graph algorithm chooses the first solution that satisfies the constraints), although some policies, in this sense, can be adopted by the new entering facilitator agent to indicate possible criteria (e.g., physical distance and relationship between offered resources) on the neighbors' choice.

6 Notes on SoRT-Bids Negotiation and Production

It is important to avoid situations of inconsistency in the state and the management of SoRT-Bids during their lifecycle from the definition to the negotiation.

In the following we will point out some critical situations, proposing guidelines for their management in SoRTGrid.

6.1 Timed Pre-reservation of Resources

Resource retrieving in SoRTGrid differs from a typical grid discovery because the negotiation and acquisition of resources is not mediated by any grid metascheduler, but is performed through a direct interaction between the user agent and the owner of the selected resource using the mechanism of *SLA-negotiation*.

A possible inconsistency is that a certain SoRT-Bid selected by the user agent could be not effectively available when the user agent tries to perform a SLA negotiation.

Such situation may occur in two cases:

- *SoRT-Bid yet negotiated*. In the time interval between the return of the SoRT-Bid to the requesting user agent and the start of SoRT-Bid negotiation, the SoRT-Bid has been negotiated by another user agent.
- *SoRT-Bid removed by the owner agent*. Before the SLA negotiation, the owner removes from its local repository the SoRT-Bid previously selected by the user agent.

The way to avoid these inconsistencies is to adopt a *timed pre-reservation* on the SoRT-Bids returned by a discovery. For a certain amount of time (T_{PR}), the SoRT-Bids are reserved for the requesting user agent: this means that during this pre-reservation interval they can neither be removed by the owner nor considered valid or returned for every other user agent query.

In this way, the timed pre-reservation assures the user agent of the effective negotiability of the SoRT-Bids obtained by a discovery.

6.2 SoRT-Bids Overbooking

An owner agent can produce any number of SoRT-Bids for the same resource: this creates another inconsistency problem due to the fact that, at every instance, the total SoRT-Bid proposal can overcome the real capacity of the resource.

For instance, let us suppose that two *soft real-time class* SoRT-Bids offer the total availability of a CPU for two periods of time that overlap each other. Since it is impossible to grant the simultaneous availability of both SoRT-Bids, such situations must be avoided by the owner agent; moreover, BidMan performs a control activity on owner agent behavior checking the Sort-Bids contained in the local repository.

For this, a set of rules and control policies has to be defined to grant both independence to the owner agent for SoRT-Bids production and a satisfying level of guarantee on SoRT-Bid negotiation for the user agent.

In general, the choice of the overbooking policy depends on the specific context and on user agent requests. The aim of our further studies will be to define for which scenarios different policies prove to be suitable, granting the maximum gain to both user agents and owner agents, and discouraging every abuse.

7 Conclusions

SoRTGrid is a grid framework designed to provide support for jobs with demanding QoS requirements, like jobs with time constraints. The efficiency of SoRTGrid strictly depends on the values of some critical parameters, like N_{Bids} (number of requested SoRT-Bids) and T_{PR} (pre-reservation time). Moreover, it depends on the policies behind a remote discovery, on the utility function that appraises the SoRT-Bids, and on the management of SoRT-Bids overbooking.

At present, we have completed the architectural design and a first implementation of SoRTGrid.

The next step will be a deep evaluation of the previous parameters: unfortunately they cannot be static and fixed, but depend on different factors, from the grid size to the number of user agents and their requests.

For these reasons, we plan to carry out various simulations to understand the impact of parameter values and other possible choices on the SoRTGrid behavior, so that SoRTGrid can be able to adapt itself to different grid scenarios. Finally, we

will investigate different possible organizations of the local repository for various SoRT-Bid characteristics in order to obtain a high search efficiency.

References

1. R. Al-Ali, K. Amin, G. von Laszewski, O. Rana, and D. Walker. An OGSA-based quality of service framework. In *Proceedings of the 2nd International Workshop on Grid and Cooperative Computing, Shangai*, pp. 529–540, 2003.
2. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. *Web Services Agreement Specification(WS-Agreement), Open Grid Forum*, 2007.
3. R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. In *Proceedings of the Fourth International Conference on High-Performance Computing in the Asia-Pacific Region*, Jun. 3–7, 2000.
4. S. Chen and K. Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17: 8, 1999.
5. A. Clematis, A. Corana, A. Merlo, V. Gianuzzi, and D. D'Agostino. Resource selection and application execution in a Grid: A migration experience from GT2 to GT4. In R. Meersman and Z. Tari, editors, *Lecture Notes in Computer Science 'On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE'*, vol. 4276, pp. 1132–1142, Springer, Berlin, 2006.
6. C. Comito, D. Talia, and P. Trunfio. Grid services: Principles, implementations and use. *International Journal of Web and Grid Services*, 1:48–68, 2005.
7. A. Corana, D. D'Agostino, A. Clematis, V. Gianuzzi, and A. Merlo. Grid services for 3D data analysis in virtual laboratories. In *Proceedings of the 2nd International Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID 2007)*, 2007.
8. C. Dumitrescu, I. Raicu, and I. Foster. DI-GRUBER: A distributed approach to grid resource brokering. In *Proceedings of the ACM/IEEE SC 2005*, 2005.
9. C.L. Dumitrescu and I. Foster. GRUBER: A grid resource usage SLA broker. In *Euro-Par 2005 Parallel Processing*, 2005.
10. T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.
11. I. Foster. Globus Toolkit Version 4: Software for Service Oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, Jul. 2006.
12. I. Foster. What is the Grid? A three point checklist. *Grid Today*, 1(6), Jul. 22, 2002.
13. I. Foster and A. Grimshaw. The Open Grid Service Architecture, Version 1.5, 2006.
14. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15: 3, 2001.
15. I. Foster and K. Kesselmann. *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier, 2003.
16. I. Foster, A. Roy, and V. Sander. A quality of service architecture that combines resource reservation and application adaptation. In *Proceedings of the 8th International Workshop on Quality of Service*, 2000.
17. Foundation for Intelligent Physical Agents Specification, 1997. `http://www.fipa.org`
18. J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 2002.
19. GENI – A Global Environment for Network Innovations.
20. C. Lin, T. Kaldewey, A. Povzner, and S.A. Brandt. Diverse soft real-time processing in an integrated system. In *Proceedings of the RTSS'06, Real-Time Systems Symposium*, 2006.
21. J. MacLaren, R. Sakellariou, K.T. Krishnakumar, J. Garibaldi, and D. Ouelhadj. Towards service level agreement based scheduling on the grid. In *14th International Conference on Automated Planning and Scheduling*, Jun. 3–7 2004.

22. R. Ranjan, A. Harwood, and R. Buyya. SLA-based coordinated superscheduling scheme for computational grids. In *Proceedings of the 8th IEEE International Conference on Cluster Computing (Cluster 2006)*, IEEE Computer Society Press, Barcelona, Spain, Sep. 27–30, 2006.
23. Z. Shi, T. Yu, and L. Liu. MG-QoS: QoS-based resource discovery in manufacturing grid. In *Grid and Cooperative Computing*, Apr. 2004.
24. G.C. Wells. New and improved: Linda in Java. *Science of Computer Programming*, 59(1–2): 82–96, 2006.
25. W. Zhao, D. Olshefski, and H Schulzrinne. Internet quality of service: An overview. Technical Report, Columbia University, New York, USA, 2000.

Chapter 14
A Data Grid Architecture for Real-Time Electron Microscopy Applications

F. Mighela and C. Perra

Abstract An architecture for remote access to scanning electron microscope (SEM) and transmission electron microscope (TEM) is presented. Abstraction between users and true instruments is provided by data grids architecture. Instruments are virtualized and remotely accessed by adaptive graphical user interfaces (GUIs) that enable functionalities on the basis of user permissions and expertise level. The proposed architecture enables for remote real-time control of main electron microscopes functionalities in a collaborative environment to be used for medical services, research services, and training services.

Keywords Remote instrumentation · Collaborative environment · Virtual laboratory · Grid architecture

1 Introduction

Remote control of scientific instruments is a field of research gaining every day increasing interest. Scientists have always shared knowledge and information, but now the necessity to share and interact in the use of physical resources is pressing them more and more.

The electron microscopes are instruments widely diffused in several research field. In the past, some microscopists have started to think about remote microscopy [11, 21, 22, 1, 3, 19] helped by the structure of the up-to-date instruments. Together with microscope technologies, also network architectures have been improved in the last years. As a consequence, the actual challenge in remote control of instruments is the development of a collaborative environment enabling the management of several scientific instruments. In [4, 2, 12, 13] some of these projects are presented. A solution implemented to overcome geographical distances between collaborating laboratories and to facilitate data sharing is presented in [4]. A demonstration of

F. Mighela (✉)
Telemicroscopy Laboratory, Sardegna Ricerche, Pula (CA), Italy
e-mail: francesca.mighela@diee.unica.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_14,

remote control of an aberration-corrected electron microscope (ACEM) realized on a dedicated large and latency-free connection between Europe and USA is presented in [2]. A grid architecture under development for a project of remote control of a high-voltage electron microscope (HVEM) is presented in [12, 13].

The project presented in this chapter makes use of a grid architecture to realize a virtual laboratory where the shared resources are a set of scanning electron microscope (SEM) and transmission electron microscope (TEM).

Abstraction between users and true instruments is provided by data grids architecture. Instruments are virtualized and remotely accessed by adaptive graphical user interfaces (GUIs) that enable functionalities on the basis of user permissions and expertise level. The proposed architecture enables for remote real-time control of main electron microscopes functionalities in a collaborative environment to be used for medical services, research services, and training services.

The chapter is organized as follows: Section 2 describes our experience about the realization of a remote SEM. In this case we have designed and implemented a server/client application for single-user remote control. Section 3 presents the virtual laboratory project reporting the ideas and the reasons for choosing a grid architecture as a basis for a remote control electron microscope application. Section 4 suggests future development for this research project. Section 5 summarizes and concludes the chapter.

2 SEM Remote Control Project

Preliminary researches on the remote control application proposed by us were published in [16, 15]. One of the objective of the ongoing research project is to define the theoretical basis for the development of a standard application.

In [16, 15] the objective was to define a remote control for a digital SEM. At first, the use of remote desktop as remote access tool was experienced. Nevertheless, user interaction was proven to be very limited and with scarce interactivity. For this reason a dedicated server/client application was designed and developed in order to augment the user experience in remote access to the SEM. The main blocks of this application are a server and a client.

The server application is installed on a PC physically close to the instrument, which works as a router between the microscope intranet and the Internet. It provides an abstraction of the user interface used by the local microscope operator to allow the connection of the remote user and a tool to manage the video of the microscope. This video management tool gets the video from the local control, encodes the compressed bitstream, and sends it through the Internet to the remote operator.

The client application resides on the remote operator's PC and is connected to the server through a TCP/IP connection. The client requests for connection to the server. If the server accepts this connection, the client gains access to the microscope through the server. The client application interacts with the remote microscope through a user interface developed ad-hoc, that provides also video player for

displaying the images of the specimen under analysis. A real-time control allows the remote operator to use the instrument with the same interaction of a local operator.

An aspect that requires more research and development is the video streaming functionality. In fact, as better explained in [4, 2], the video is delivered to the client with an average of 3 s of delay on a public connection. This delay is not admissible for using an instrument that needs real-time interaction: in fact the specimen images are the instrument feedback to the user operations than it is necessary to have a real-time feedback also for the remote operator.

This project has been a university project aiming at the research and development of a SEM remote control architecture [20]. On the basis of these preliminary results, the SEM remote control project has evolved into another project focused on remote control of several instruments. This second one is supported by the European Community and the Sardinian Government P.O.R. 2000. The target is to define a remote control architecture for controlling four different electron microscopes, located in three different laboratories: one in Cagliari (Italy), two in Pula (Italy), and the other in Sassari (Italy) [14].

While with the first project we demonstrated that it is possible to develop remote control applications on public networks, this second project is mainly focused on remote real-time control of distributed electron microscopes functionalities in a collaborative environment to be used for medical services, research services, and training services.

3 Remote SEM/TEM and Grid Architecture

3.1 Remote SEM/TEM Requirements

Starting from the results of the projects described above, a new idea of remote control microscopes has been considered: create a virtual laboratory for sharing all the instruments physically present in the involved laboratories. In fact, in this new scenarios there are four instruments, three SEMs and one TEM, located in three different laboratories. The objective is then to remote control each microscope and moreover create a network where an authorized remote client anywhere in the world can be able to manage one of them. At the beginning we defined the features of the architecture:

- security access;
- security connection;
- real-time interaction;
- interoperability;
- concurrent access by multiple users to the same resource;
- public network availability;
- real-time streaming video.

These features can be easily implemented on a grid architecture. Grid architectures have been proposed as collaborative and distributed computing infrastructures [6], with the following characteristics:

- sharing resources and knowledge between different organizations;
- interoperability;
- services;
- controlled access;
- user identification;
- internet protocols;
- World Wide Web technologies.

Foster, Kesselman, and Tuecke discussed the importance and relevance of each point in [7]. Some of the project features have been already implemented at the end of the first remote microscope project. Some others are peculiar of this current project. These additional features can be easily implemented if a grid architecture is chosen as the core architecture of the virtual laboratory.

The first requirements concern security access and connection; this is because of instruments' delicacy, and the grid architecture allows to define security access control and also gives the possibility to define different user profiles with different permissions. This point acquires more relevance considering the necessity to give concurrent access of a same resource to more than one user. This can be useful in more than one case; some examples are

- when there is a particular specimen and it could be useful to contact some experts to have a consultancy;
- when there are more than one student under training;
- when the person who needs the microscope use is not able to use it than an expert has to carry out the analysis for him.

The other aspect is the interoperability condition, determined by the different types of instrument, which means also different manufacturers. In fact, the considered instruments are different, also even if there are three SEMs, but a lot of standard microscope operations are the same. The user should focus only on the microscope analysis process without being really concerned into the comprehension of proprietary user interfaces and control commands. Another reason of interoperability property is the possibility to improve the microscope services with specific tools to elaborate images chosen by the user, regardless of the instrument used to acquire those.

The last feature is the integration of World Wide Web technologies into grid architecture. This point is fundamental to reach the purpose of a virtual laboratory on the Internet.

In this section we described the kind of architecture chosen to realize the virtual laboratory project. The next section will discuss about the configuration architecture needed to realize a SEM/TEM grid architecture.

3.2 A Grid for a Virtual SEM/TEM Laboratory

A grid architecture might be configured following different solution schemes. In general, each solution is a layered structure, but the number of layers or the layers' functions can be different. A typical solution is represented in Fig. 14.1, where there are four layers, each with a given function [10].

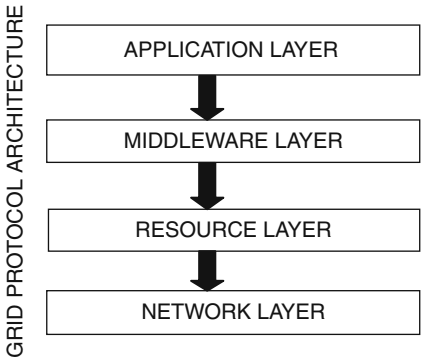


Fig. 14.1 An example of layered grid architecture

The network layer is the layer that connects all grid resources and contains the interfaces with the local control.

The resource layer groups grid resources, such as computers, storage systems, instruments, sensors, electronic data catalogue, and database.

The middleware layer can be considered the grid brain; this is the layer where there are all the tools that allow the interaction between the grid resources available.

The highest layer is the application layer, that is the only one accessible by the user. This layer implements the user interfaces of the available applications.

This is one of all possible existing layered grid solutions. The architecture choice will be followed by the architecture configuration that will be divided into two different parts: the first one concerning the definition of layers' functionalities, while the second one regarding the protocols' definition for each layer.

If we assume to use the configuration shown in Fig. 14.1, we have a four-layered grid architecture; then, we can image to assign to each layer the functions that will be available in the virtual laboratory. The network layer should contain the communication protocols and the security connection system. The main aspect for this layer is the protocols definition. The resources layer groups the shared resources: the four electron microscopes and some storage systems. At this layer also the video streaming system will be implemented, which represents one of the main challenges considering the real-time requirements of the virtual laboratory. The middleware layer is the layer that manages the resources' availability and the customers' requests. Also in this case the protocols used to create the effective collaborative environment are determinant in order to satisfy the project requirements. The last layer is the application layer, where there are all the tools that can be managed by

the users. The tools are the interfaces for the remote control of the instruments, the video streaming application, the storage of data and images acquired during the microscope analyses that can be shared with other microscopists, and some tools for image processing. Figure 14.2 shows a more detailed description of each layer, considering the assigned role.

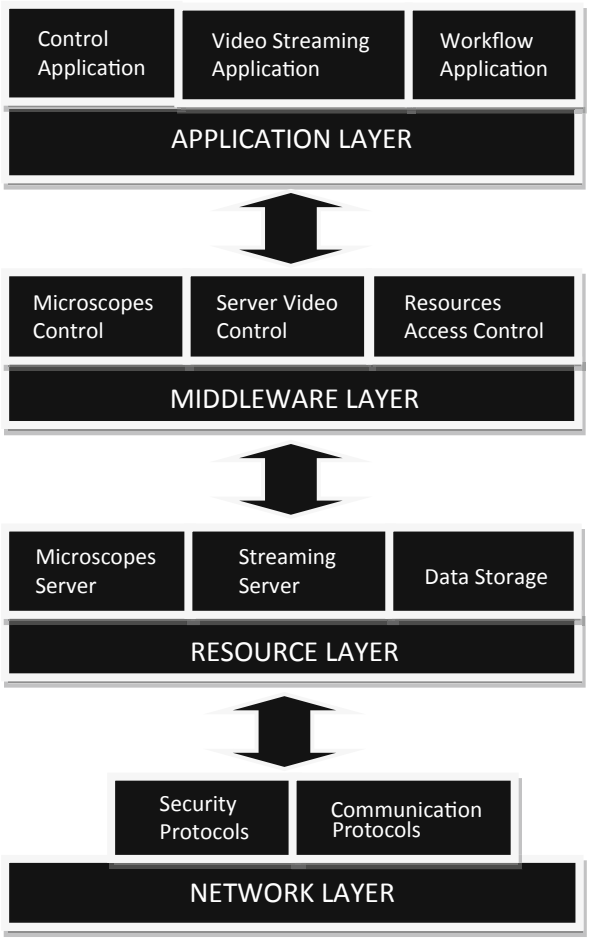


Fig. 14.2 A description grid architecture on the basis of the functionalities assigned to each layer

As explained before, the software instruments used to implement these roles are determinant concerning the good use of the virtual laboratory.

In order to satisfy this aspect, we are considering several data grid frameworks and middleware tools, solutions that have been developed and tested in several projects [5].

The most common used system is, actually, the Globus Toolkit 4 (GT4). The Globus Toolkit is an open source software toolkit used for building grids, developed by the Globus Alliance [8, 9].

Different from other systems, the GT4 is a complete toolkit that gives instruments to develop every part (or layer) of a grid architecture. The advantage of choosing it is that all protocols used on each layer are already defined. These could be either standard protocols or particular versions elaborated starting from standard protocols; this guarantees the interoperability with different platforms and then with sharing resources. The integration of microscopes' applications is possible thanks to the API given by the instruments manufacturers.

Thanks to the API and to the layered grid structure, it has been possible to design a unique interface for all instruments. Figure 14.3 shows the designed GUI. As explained above, the instruments are three SEMs and one TEM. The SEMs are an environmental SEM, a low vacuum SEM, and a FEG SEM. All these instruments have some common controls, but others are related to the particular configuration. The GUI designed contains all common controls in the framework that will be available for every instrument (server connection, normal operations, beam setting, detectors, image), and the specific controls in a separated framework available just for the designed instrument (FEG setting and ESEM setting). The framework called 3D contains a tool to reconstruct a three-dimensional view of the observed surface, starting from four specimen images. As better explained in [17, 18], this tool can be developed for every instruments, then it will be available for all. The framework called TEM groups all the commands specific for the TEM instrument.

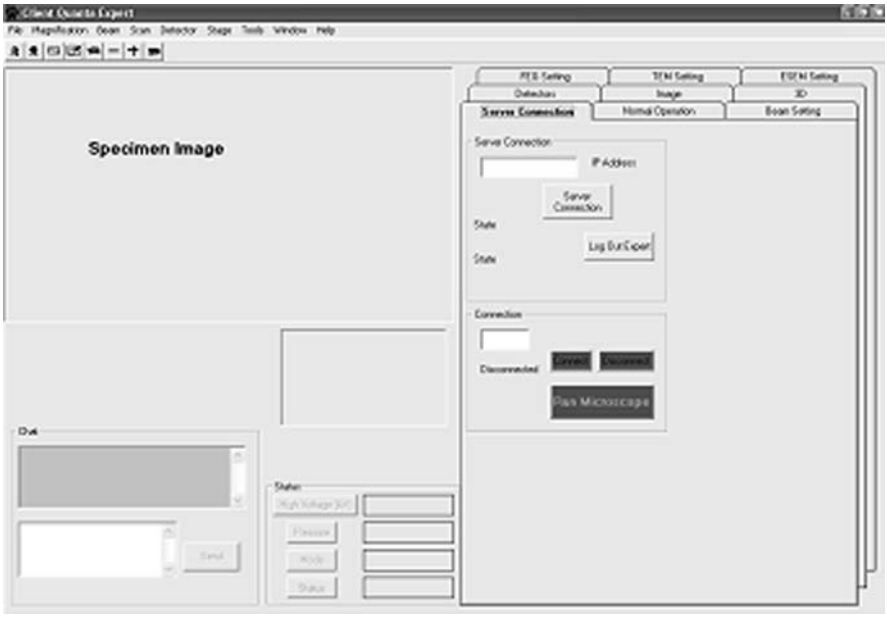


Fig. 14.3 The GUI designed for the virtual laboratory

At the moment of remote user connection, the user has to choose the instrument, and then just the commands related to the microscope selected will be enabled. The intention is to separate the GUI of the application used to control the instrument, from the GUI developed for the same instrument by the manufacturer. In this way the remote operator is not affected by the microscope manufacturer, but he is interested only in the microscope kind.

Moreover, on the basis of the user experience, some commands will be enabled and some others will not. The idea is to create three different user profiles, as done in the server/client applications previously described. The user profile will be an expert profile, with all commands enabled, a student profile, with a selected set of commands enabled, and a non-expert user, with no commands enabled.

Ideas for video streaming architecture. The most important requirement is to guarantee a real-time transmission for long time, because a microscope session can be a day long. A standard video coding algorithm (MPEG/H.264) will be exploited for the encoding of video data produced by microscopes. In this case a frame grabber card will be used, with its proprietary software development kit (SDK). This SDK will allow the interoperability between the grid structure and the video streaming system.

Considering the Telemicroscopy as a test bed for network technologies and video streaming solutions, it will also be interesting to test the video on the web TV technology and on mobile communication systems. Probably the results will not reach all requirements, but it will be possible to obtain some interesting results about the technologies, like the behavior of web TV with a non-standard video input or the delay of the mobile communication networks.

4 Perspectives

As said in the previous section, in this preliminary phase of the project we have explored a set of data grid architectures, and we chose the GT4 tool to develop the virtual laboratory here presented. Analysis and experimental tests will be followed by revisions, adaptation, integration in order to tune at best the virtual laboratory. At the best of our knowledge, the problem of real-time video streaming and interactive remote control of electron microscopes is a problem that has just started to be investigated and that deserves a lot of researches for defining a practical solution for daily laboratory services.

5 Conclusions

In this chapter we presented an ongoing project of a virtual laboratory on a grid architecture. A preliminary analysis has been reported describing the high level architecture. The proposed model makes use of state-of-the-art technologies and methodologies as well as existing standards in order to maintain a high level of

scalability and adaptability to different applicative cases. This architecture will provide general means for remote real-time control of main electron microscopes functionalities in a collaborative environment to be used for medical services, research services, and training services.

References

1. L.F. Allard and E. Voelkl. Invited tutorial on electron microscopy via the internet. In *14th International Congress on Electron Microscopy*, Cancun, MX, Sep. 1998.
2. Atlan TICC Allinace. http://www.atlanticcalliance.org/news_globallab.html
3. A. Van Balen and C.L. Morgan. Details from a distance: Seeing Angstroms at 10,000 Kilometers. In *INET Conference Proceedings*, pp. 386–397, 2000.
4. H. Bles, M. Boom, L.B. van Dijk, E. van Duin, H. Jacobs, S. Joosen, E. van Breemen, and A. van Balen. A collaborative environment for users of advanced scientific instruments. In *Third International Conference on Creating, Connecting and Collaborating through Computing*, pp. 83–90, 2005.
5. F. Berman, G.C. Fox, and A.J.G. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, 2003.
6. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.
7. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3):200–222, 2001.
8. Globus. http://www.globus.org
9. Globus Tooolkit. http://www.globus.org/alliance/publications/papers.php#anatomy
10. Grid Cafe. http://gridcafe.web.cern.ch/gridcafe/
11. G. Grimes, S. McClellan, J. Goldman, G. Vaughn, D. Conner, J. McDonald, T. Winokur, G. Siegal, and W. Fleming. Remote microscopy for hi-res real-time interactive pathology. *Advanced Imaging*, p. 13ff., Jul. 1997.
12. H. Han, H. Jung, H.Y. Yeom, H.S. Kweon, and J. Lee. HVEM Grid: Experiences in constructing an electron microscopy grid. *Frontiers of WWW Research and Development – APWeb 2006 LSCN 3841/2006*, pp. 1156–1162, 2006.
13. Im Y. Jung, In S. Cho, and Heon Y. Yeom. A stateful web service with scalable security on HVEM DataGrid. In *e–science, Third IEEE International Conference on e–Science and Grid Computing, (e–Science 2007)*, pp. 360–369, 2007.
14. LIMINA laboratory. http://dipcia.unica.it/superf/LIMINA.html
15. F. Mighela and C. Perra. Remote control for microscopy applications. In *Instrumentation and Measurement Technology Conference Proceedings*, Apr. 2006.
16. F. Mighela, C. Perra, and M. Vanzi. Long distance SEM remote control. In *The 16th International Microscopy Congress Proceedings*, Sep. 2006.
17. R. Pintus, S. Podda, F. Mighela, and M. Vanzi. Quantitative 3D reconstruction from BS imaging. *Microelectronics Reliability*, 44:1547–1552, 2004.
18. R. Pintus, S. Podda, and M. Vanzi. An automatic alignment procedure for a four-source photometric stereo technique applied to scanning electron microscopy. *IEEE Transactions on Instrumentation and Measurement*, 57(5):989–995, 2008.
19. C.S. Potter, B. Carragher, L. Carroll, C. Conway, B. Grosser, J. Hanlon, N. Kisseberth, S. Robinson, U. Thakkar, and D. Webber. Bugscope: A practical approach to providing remote microscopy for science education outreach. *Microscopy and Microanalysis*, 7:249–252, 2001.
20. Sardegna Ricerche. http://www.sardegnaricerche.it/index.php?xsl=370&s=61299&v=2&c=4209&nc=1&sc=&fa=1&o=1&asn=61294&t=3&cattmp=4205

21. E. Voelkl, L.F. Allard, J. Bruley, and D.B. Williams. (Under)graduate teaching using internet access to electron microscopes. In *Proceedings of the MSA, Springer*, vol. 3. Suppl. 2, pp. 1095–1096, 1997.
22. N.J. Zaluzec. Tele-presence microscopy: A new paradigm for collaboration using the Internet. In *Symposium on the Future of Remote Microscopy in Materials Science Education*, Carnegie Mellon University, Sep. 1998.

Chapter 15
A Network-Aware Grid for Efficient Parallel Monte Carlo Simulation of Coagulation Phenomena

M. Marchenko, D. Adami, C. Callegari, S. Giordano, and M. Pagano

Abstract Remote instrumentation services often rely on distributed control and computation. In this framework, a grid infrastructure offers the opportunity to utilize massive memory and computing resources for the storage and processing of data generated by scientific equipment. The provision of grid-enabled remote instrumentation services is also a key challenge for high-speed next generation networks and cross-layer mechanisms, between grid middleware and network protocols, should be introduced to support quality of service (QoS) both at network and at application layers. This chapter is mainly focused on distributed data computation and, more in detail, on the optimization of the time necessary to perform a simulation concerning particles coagulation phenomena. Extensive computations have been performed in a cluster environment and the collected data have been used as a starting point for the dimensioning of a proper grid infrastructure. Since computations are characterized by frequent data exchanges, this chapter investigates how bandwidth allocation affects overall performance.

Keywords DSMC · GNRB · Dynamic resource allocation · MPLS-TE

1 Introduction

Nowadays, the success of many research and industrial projects more and more depends on the availability of unique and expensive laboratory equipment or computing resources. Hence, Remote Instrumentation Services (RIS) and middleware, which allows virtualized and shared access to industrial or scientific equipment, are key factors to open up new relevant opportunities for industry, science, and business applications.

In this framework, grid technologies offer effective strategies to achieve such ambitious goals. Originally conceived to solve large-scale computation problems,

M. Marchenko (✉)
Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia
e-mail: mam@osmf.sscc.ru

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_15,

a grid may be defined as "a system which coordinates resources that are not subject to a centralized control, using standard, open, general-purpose protocols and interfaces, to deliver non-trivial qualities of service" [3, 4, 5]. In most cases, grid applications heavily stress the network used to interconnect the grid resources, so the performance of the applications is highly affected by the type of service provided by the network. Emerging e-Science applications have many networking requirements, such as cost-effectiveness, flexibility, inter-domain and internetworking dynamic provisioning, and scheduling of short-long-term as well as low-high-capacity connections. In this framework, this chapter deals with a coagulation phenomenon, that is a process of merging smaller particles (or droplets/clusters/bubbles) into bigger ones. The particles represent an aerosol (or small parts of a solid matter) that is suspended in a disperse medium, such as a gas or a liquid. In chemical engineering, phenomena of diffusion or dispersion as well as coagulation–fragmentation are of great importance in multiphase systems. Two multiphase systems of high industrial relevance, in this respect, are liquid–liquid extractions in bubble columns and flame aerosol technology for the production of nanoscale particles originating from the gaseous phase in flames. The latter process is industrially used for the production of carbon black and ceramic powders, such as fumed silica and pigmentary titanium oxide, of the order of several million tons per year. Closely related to the production of carbon black is the formation of soot particles in hydrocarbon combustion systems used for the production of energy, such as diesel engines and gas turbines. The control of these emissions is relevant, since there are evidences that they can damage the environment and human health.

To avoid the formation of soot or to make the formation of powder ceramics more efficient, knowledge of the underlying physical and chemical processes of particle formation and dynamics has to be improved. To this end, more efficient and accurate mathematical methods capable of including more detailed physical models are needed. This is also of immense economic interest, since flame synthesis represents the most inexpensive manufacturing process of nanoparticles that can be used for a wide range of new materials. All these processes have in common that the control of the particle ensemble characteristics is essential. For instance, in the case of flame aerosol synthesis the properties of the final material highly depend on the size distribution of the produced particles.

In industrial processes, particles are produced in diffusion flames under laminar or turbulent conditions. Thus, the most promising mathematical model of a coagulation process is a spatially inhomogeneous model which allows for evaluating a dependence of particles' concentrations upon space locations. This model also takes into account the diffusion and velocity shift of the particles and their initial spatial distribution.

The simulation of a coagulation process may be performed by using a cluster of nodes connected through a switched LAN with different bandwidth, but, in most cases, the large amount of data to be processed requires the use of a computing grid. Since the frequency of data exchanged among the processing nodes is really high, the time necessary for running the simulations depends on the QoS (quality of service) provided at network level. In previous works [2] some of the authors of this

chapter proposed a centralized approach based on a Grid Network Resource Broker (GNRB) to dynamically provision Bandwidth-on-Demand (BoD) services for grid applications in DiffServ/MPLS networks [6, 8]. In more detail, this chapter focuses on a comparison between performance that can be obtained for a coagulation process simulation when a cluster or a grid infrastructure is considered.

This chapter is organized as follows: Section 2 analyzes the problem of simulating a coagulation phenomenon by using a parallel implementation of the Monte Carlo algorithm. Next, Section 3 shortly describes the architecture of a GNRB, focusing, in more detail, on the path computation algorithm necessary to set up LSPs with QoS constraints. Section 4 presents the simulation scenario and the results of the analysis carried out to evaluate the performance of the coagulation phenomenon simulation when a cluster or a grid is taken into account. Finally, Section 5 contains some final remarks.

2 Monte Carlo Simulation of a Coagulation Problem

2.1 Statement of the Problem

Let us call an i-mer an aggregate of i particles of a given minimal size (or volume, mass). Then consider the following Cauchy problem for a spatially inhomogeneous nonlinear equation of coagulation:

$$\frac{\partial c_1}{\partial t} + div\left(\overrightarrow{v}\, c_1\right) = d_1 c_1 - c_1 \sum_{j=1}^{+\infty} K(1,j) c_j$$

$$\frac{\partial c_i}{\partial t} + div\left(\overrightarrow{v}\, c_i\right) = d_i c_i + \frac{1}{2}\sum_{j=1}^{i-1} K(i-j,j) c_j - c_i \sum_{j=1}^{+\infty} K(i,j) c_j, \quad i > 1$$

$$c_i(0, \overrightarrow{x}) = c_i^0(\overrightarrow{x})$$

where

- $c_i = c_i(t, \overrightarrow{x})$ – concentrations of i-mers at time point t and spatial point $\overrightarrow{x}$;
- $\overrightarrow{v} = \overrightarrow{v}(\overrightarrow{x})$ – spatially inhomogeneous velocity field;
- d_i – diffusion coefficients of i–mers;
- $K(i,j)$ – coagulation kernel;
- $c_i^0(\overrightarrow{x})$ – concentration of i-mers at $t = 0$.

We evaluate the following space integrals:

$$\varphi = \iiint_G c_i(T, \overrightarrow{x}) d\overrightarrow{x} \quad \text{or} \quad \varphi = \iiint_G \sum_{i=1}^{+\infty} i^p c_i(T, \overrightarrow{x}) d\overrightarrow{x}$$

considering the equation within time–spatial domain $[0,T] \times \Omega$, $T < \infty, \Omega \subset R^3$.

2.2 Parallel Monte Carlo Algorithm

It is very hard to solve the coagulation equation numerically because it consists of an infinite number of nonlinear equations. But there exists an effective approach to solve the equation: a Monte Carlo method known as a direct simulation Monte Carlo algorithm (DSMC) [7].

We simulate an evolution of a test particles ensemble:

$$\pi(t) = \{p_1, p_2, \ldots, p_N\}$$

where $N = N(t)$ is the number of particles (it is a decreasing function of time), a test particle p_k is a pair of parameters: $p_k = (l_k, \overrightarrow{x}_k)$,$l_k$ – a particle size and $\overrightarrow{x}_k$ – a position of the particle. Denote by N_0 the initial number of test particles at $t = 0$.

To evaluate the functional φ, we simulate L independent samples of test particles ensemble $\pi_1, \pi_2, \ldots, \pi_L$ and use a sample average

$$\varphi \approx \frac{1}{L} \sum_{l=1}^{L} f(\pi_l)$$

where f is a given function of the test particles ensemble.

To define parallel simulation on M processors, we split the spatial domain Ω into quite big non-overlapping subdomains $\widehat{\Omega}_1,\widehat{\Omega}_2,\ldots,\widehat{\Omega}_M$, indicated in the following as processors' subdomains, where subdomain $\widehat{\Omega}_m$ is processed by the mth processor ($m = 1, 2, \ldots, M$). The test particles are sorted out to the different processors' subdomains.

We also split the spatial domain Ω into S small non-overlapping subdomains $\Omega_1,\Omega_2,\ldots,\Omega_S$. They are referred to as interaction subdomains. Note that interaction subdomains are much smaller than the processors' subdomains $\widehat{\Omega}_m$. Denote by ϱ a typical volume of interaction subdomains $\widehat{\Omega}_m$. Test particles coagulate (interact) with each other only within interaction subdomains. We split the time interval [0,T] into $T/\Delta t$ subintervals (time stages) of size Δt. Over each time stage, we split the simulation into two processes: simulation of coagulation and simulation of particles transport.

Thus, the main parameters of the algorithm are the following:

- M – number of processors,
- $\{\widehat{\Omega}_m\}_{m=1}^{M}$ – set of processors subdomains,
- L – number of independent samples,
- N_0 – initial number of test particles,
- Δt – time step size,
- ϱ – typical volume of interaction subdomains.

A scheme of the simulation of a sample of the test particles ensemble is shown in Fig. 15.1 where we consider four processors and a single processor is denoted by a rectangle.

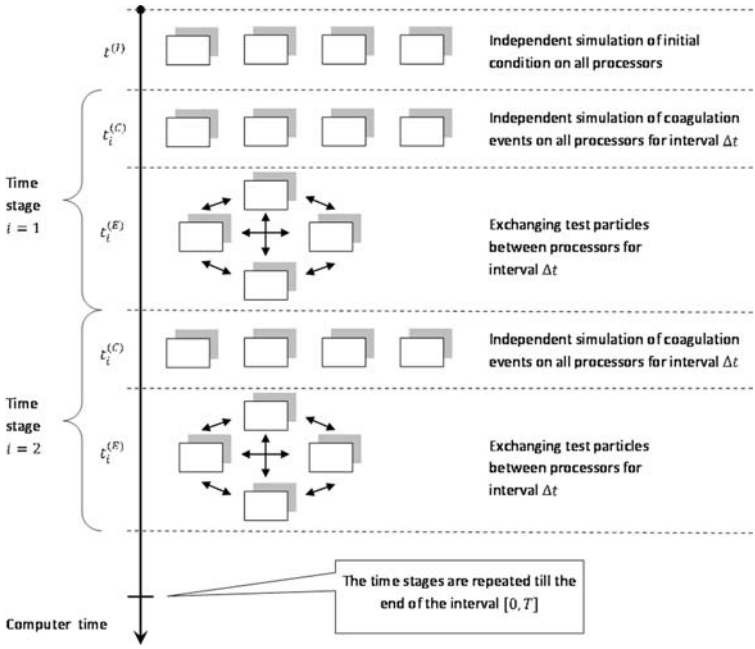


Fig. 15.1 Simulation scheme

Let us make some remarks on the simulation of a sample of the test particles ensemble:

1. The algorithm is synchronized at the end of each coagulation stage and each data exchange stage.
2. On all processors initial conditions are simulated independently. The computer time corresponding to this phase is $t^{(I)} \sim N_0$
3. The simulation of coagulation events consists in simulating the exponentially distributed time intervals between subsequent coagulation events, choosing a pair of test particles according to a special probabilistic law, and changing the phase state of test particles ensemble (see Fig. 15.2).
4. The test particles are moving according to a Brownian motion. For each test particle $p = (l, \overrightarrow{x})$, its new position $\overrightarrow{x}'$ is simulated using Euler method for stochastic differential equations with step size Δt:

$$\overrightarrow{x}' = \overrightarrow{x} + \Delta t \overrightarrow{v}(\overrightarrow{x}) + \sqrt{\Delta t d_l} \overrightarrow{\eta}$$

The computer time corresponding to simulating the coagulation is as follows:

$$t^{(C)} = \sum_{i=1}^{\frac{T}{\Delta t}} t_i^{(C)}, t_i^{(C)} = \max_{m=1,2,\ldots,M} t_{i,m}^{(C)}$$

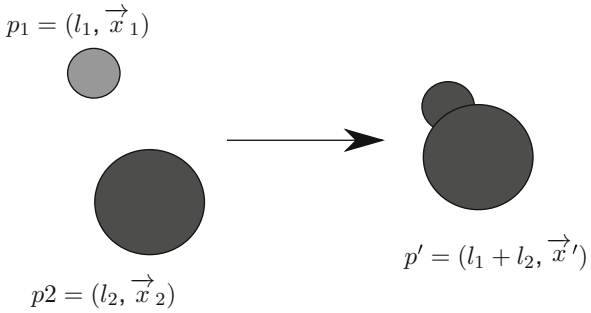


Fig. 15.2 Coagulation of two *i*-mers

where $t_{i,m}^{(C)}$ is the computer time for the *i*th time stage on the *m*th processor. The computer time corresponding to the exchange of particles is as follows:

$$t^{(E)} = \sum_{i=1}^{\frac{T}{\Delta t}} t_i^{(E)}, t_i^{(E)} = \max_{m=1,2,\ldots,M} t_{i,m}^{(E)}$$

where $t_{i,m}^{(E)}$ is the computer time for the *i*th time stage of the *m*th processor. It is known that

$$t_{i,m}^{(C)} \approx p_m \Delta t n_{i,m}$$

where the constant p_m is determined by the processor performance and $n_{i,m}$ is the number of test particles on the *m*th processor (i.e., within the subdomain $\widehat{\Omega}_m$).

Assume that it is possible to define $\{\widehat{\Omega}_m\}_{m=1}^{M}$ such that

$$t_{i,1}^{(C)} \approx t_{i,2}^{(C)} \approx \ldots \approx t_{i,M}^{(C)}$$

for each stage $i = 1, 2, \ldots, T/\Delta t$. We will call this case a balanced sequential calculations scenario.

Denote by $b_i^{(E)}$ the amount of data in bytes transferred between processors during the time stage *i*. It is possible to prove that $b_i^{(E)}$ has the following order of magnitude:

$$b_i^{(E)} \sim \Delta t N_0$$

Nevertheless, the value of $t_i^{(E)}$ shows a different behavior. Under some restrictions for the number of links between processors, it is possible to prove that the order of magnitude of $t_i^{(E)}$ is

$$t_i^{(E)} \sim \delta t N_0 M^r, r \geq 0$$

Therefore, for the balanced sequential calculations scenario, the total computational time $t_1(M)$ to simulate one sample of test particles ensemble is estimated as follows:

$$t_1(M) = t^{(I)} + t^{(C)} + t^{(E)} \sim$$
$$C_I N_0 + N_0 M^{-1}(C_f \varrho^{-1} M^{-1} + C_r) + C_E N_0 M^r$$

where C_I, C_r, C_E are constants. Denote by Φ the relative efficiency of computations:

$$\Phi = \frac{t_1(1)}{t_1(M)}$$

and assume the following dependence of N_0 on M:

$$N_0(M) \sim M^d, 0 < d \leq 1$$

Then it is possible to prove that for the balanced sequential calculations scenario the relative efficiency of computations has the following order of magnitude while $M \to \infty$:

- if $C_E = 0$, then $\Phi \sim M^d \to \infty$,
- if $C_E > 0$, $d < r$, then $\Phi \sim M^{d-r} \to 0$,
- if $C_E > 0$, $d = r$, then $\Phi \sim constant$,
- if $C_E > 0$, $d > r$, then $\Phi \sim M^{d-r} \to \infty$.

3 Network-Aware Grid Architecture

The cooperative working of a grid with a network infrastructure requires the development of a new grid middleware service, called Network Resources Manager (NRM), which can be invoked by the Grid Application Manager (GAM) to ask for different network services.

As previously stated, to achieve a grid network-aware environment, we proposed a centralized architecture where a new entity, the Grid Network Resource Broker (GNRB) (Fig. 15.3), provides two basic functionalities: network resource management and network information retrieval.

One of the most important blocks of the GNRB is the so-called Path Computation Element (PCE). Given N LSP setup requests, the basic function of the PCE is to find N network paths that satisfy the LSP QoS constraints. In particular, taking into account QoS metrics and network status information, the goal is to determine a set of paths that matches the QoS requirements, while minimizing the probability to reject other requests. To this aim, one of the most widely used solution is represented by the Wang–Crowcroft (WC) algorithm [11].

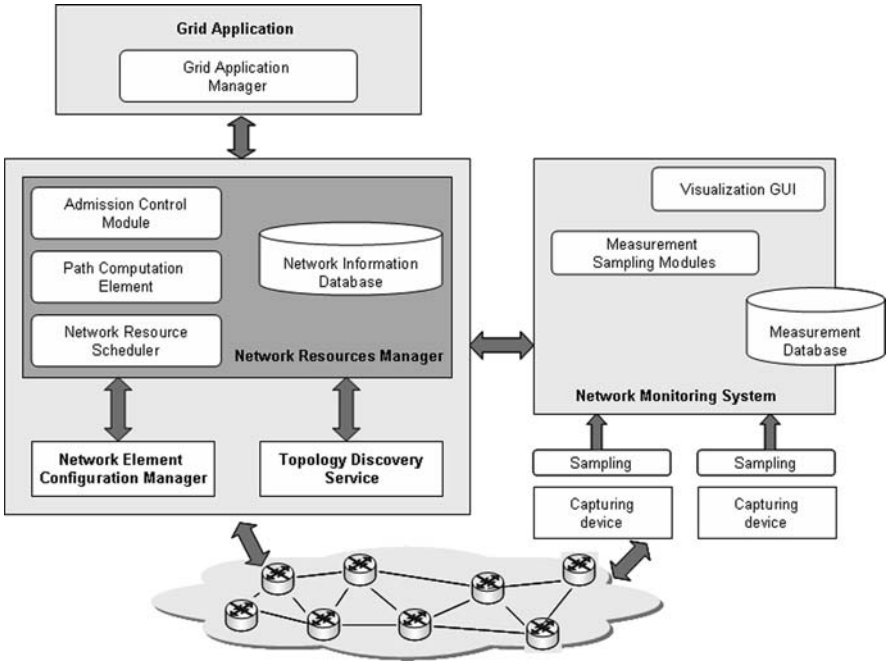


Fig. 15.3 GNRB architecture

3.1 Path Computation Algorithm

The Wang–Crowcroft (WC) path computation algorithm aims at finding a path which satisfies multiple QoS constraints, given in terms of bandwidth (B_{MIN}) and delay(D_{MAX}). Every link (i,j) is characterized by two parameters: b_{ij} (residual bandwidth) and d_{ij} (propagation delay).

The algorithm consists of the following steps:

1. set $d_{ij} = \infty$ if $b_{ij} < B_{\mathrm{MIN}}$
2. compute the path P with the minimum delay D^* (applying Dijkstra's algorithm)
3. If $D^* < D_{\mathrm{MAX}}$ select the path, otherwise the request cannot be satisfied

When a centralized NRM has to set up N LSPs with bandwidth and delay constraints, the number of LSPs that can be allocated may depend on the order in which the requests are processed by the PCE. Therefore, we have introduced the Wang–Crowcroft with Bandwidth and Delay Sorting (WCS) [1], a new algorithm, aiming to improve the performance of WC when all the path computation requests cannot be satisfied.

The basic idea behind the WCS algorithm (Fig. 15.4) is to analyze the path computation requests after re-ordering them based on their bandwidth and delay requirements.

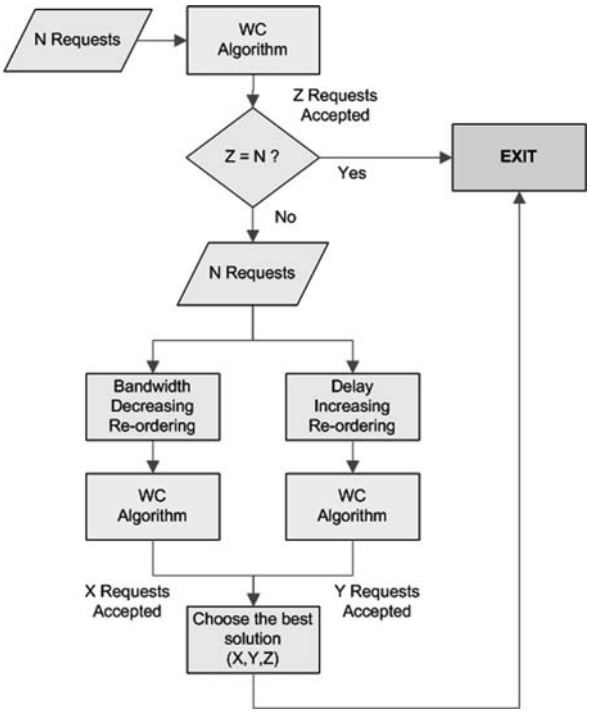


Fig. 15.4 Path computation algorithm flow diagram

In particular, given a random sequence of N path computation requests, our algorithm checks whether they can be satisfied as a whole by applying the WC algorithm. If some of them are rejected, we compute the number of requests that can be accepted when

1. they are sorted top-down according to the bandwidth B_{MIN} requirements;
2. they are sorted bottom-up according to the delay D_{MAX} requirements.

Finally, the best solution according to a pre-defined cost function (e.g., the maximum number of reservation requests that can be accepted) is chosen.

It is worth noticing that our algorithm, unlike the original version, takes into account all the components (i.e., transmission, queuing, and propagation delays) of the end-to-end delay. In particular, the queuing delay is bounded by a deterministic quantity evaluated by using network calculus theory.

4 Performance Analysis

In this section, we consider the implementation of the DSMC algorithm in a grid infrastructure. In particular, we assume that several computing nodes (called, in the following, processors) are available on a cluster and that LANs' switches are

interconnected by a DiffServ/MPLS backbone, managed by the GNRB described in Section 3.

The first issue to be solved concerns the distribution of the test particles among the M processors. The balanced sequential calculations scenario seems to be a good approach; if necessary, the appropriate load balancing procedure may be carried out.

Then, it is necessary to estimate the QoS parameters of the backbone connections among processors. The theoretical calculations reported in Section 2 refer to the asymptotic behavior (when $M \to \infty$), while, in practice, the computational resources are always limited. Therefore, it is necessary to estimate the minimal network bandwidth B_{MIN} under which

$$t_1(1) = t_1(M^*), \text{ i.e., } \Phi = 1$$

where M^* is the number of available processors.

To determine B_{MIN}, at first we fix the adequate values of ϱ and Δt. At the end we have the estimates for the values $t^{(I)}$ and $t_i^{(C)}$, $b_i^{(E)}$,$i = 1,2,\ldots,T/\Delta t$. Having all this information, we can simulate the behavior of the network to evaluate the minimal value B_{MIN} which satisfies the condition $t_1(1) = t_1(M^*)$. In the following sections, we will describe the parameters of the coagulation equation and the topology of the MPLS network (with $M^* = 4$) considered as a simple case study.

4.1 Test Coagulation Equation

In our test case, the domain Ω is a cylinder with base radius R and height H:

$$\Omega = \left\{ \overrightarrow{x} : 0 \le x_1 \le H, x_2^2 + x_3^2 \le R^2 \right\}$$

The other parameters of the coagulation equation are as follows:

$$\overrightarrow{v} \equiv 0;\ d_l \equiv const,\ l = 1,2,\ldots$$

$$K(i,j) = 1$$

$$c_i^0(\overrightarrow{x}) = \begin{cases} e^{-C\sqrt{x_2^2+x_3^2}}, & H_1 \le x_1 \le H_2 \\ 0, & x_1 < H_1 \text{ or } x_1 > H_2 \end{cases}$$

By properly choosing the value of C, it is possible to achieve the desirable degree of unbalancing for the computational load on the M^* processors. In particular, $C = 1$ leads to a totally balanced sequential calculations scenario for the choice of $\{\widehat{\Omega}_m\}_{m=1}^M$ described below.

In our case the process of particles transportation also involves the interaction with the border of the domain Ω. Namely, we introduce a reflection from the borders of the domain. From the mathematical point of view it is necessary to add appropriate boundary conditions to the coagulation equation.

Interaction domains $\Omega_1,\Omega_2,\ldots,\Omega_S$ are assumed to be small cubes with equal volume ϱ.

The value of the step size Δt is chosen to enable the particles' exchange only between the nearest-neighbor domains $\{\widehat{\Omega}_m\}_{m=1}^M$: such choice permits to reduce the interactions among processors (see Fig. 15.5) and, as a consequence, the number of LSPs to be allocated, which would be $2(M-1)$.

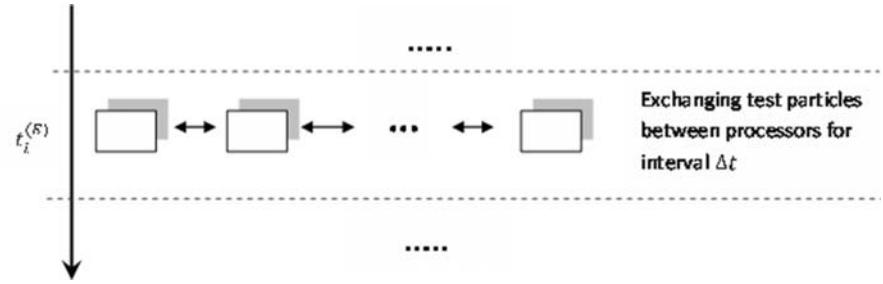


Fig. 15.5 Interaction among neighboring processors

Namely, $\{\widehat{\Omega}_m\}_{m=1}^M$ are chosen as equal volume segments of the cylinder Ω:

$$\widehat{\Omega}_m = \left\{ \overrightarrow{x} : 0 \le x_1 \le H,\ r_{m-1}^2 \le x_2^2 + x_3^2 \le r_m^2 \right\}$$

$$r_0 = 0, \quad r_m = \sqrt{\frac{m}{M}} R, \quad m = 1,2,\ldots,M$$

4.2 Simulation Scenario

As previously stated, the goal of our performance analysis is to evaluate the time necessary to carry out the simulation of the coagulation phenomenon by using a cluster or a grid. In the following, we suppose that the grid infrastructure consists of the same computing resources as the cluster and that the interconnecting network is an MPLS backbone. More specifically, in our simulation scenario, the network topology matches the GÉANT Network, a multi-gigabit pan-European data communications infrastructure, which has been deployed for research and education services. As shown in Fig. 15.6, we consider $M = 4$ processors (nodes A, B, C, and D) connected to four different network nodes and we suppose that the remote LANs are not bottlenecks.

All the other network nodes belong to a unique MPLS domain. Table 15.1 lists the bandwidth and the propagation delay associated to each link.

Simulations are based on experimental data collected on a high-performance cluster located at the Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences, Novosibirsk, Russia. The cluster consists of 80 dual processor nodes (HP Integrity rx1620, 2 processors Intel Itanium 2, 1.6 GHz, 3 MB cache, 4 GB RAM) with a peak computational

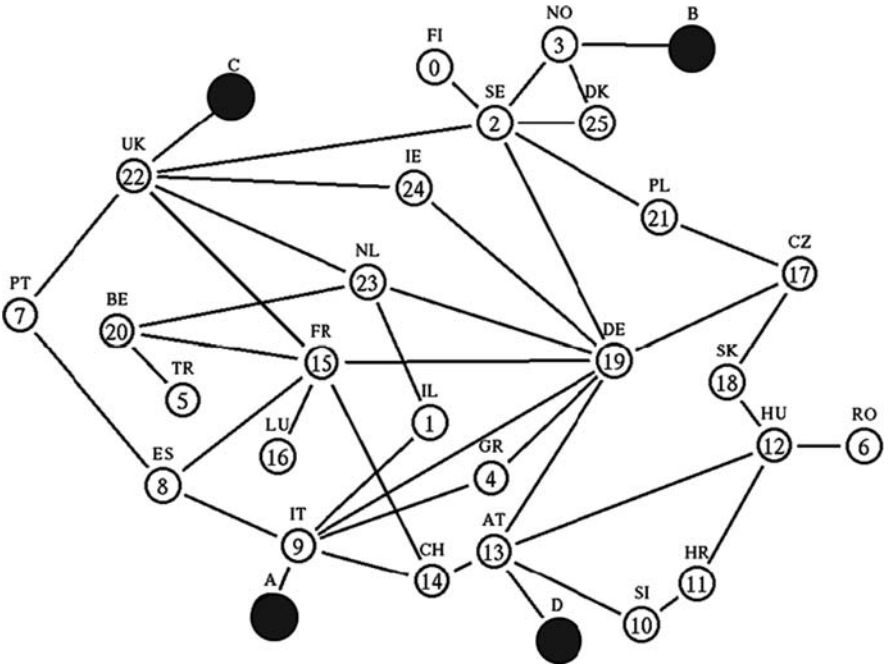


Fig. 15.6 Network topology

Table 15.1 Link metrics

Node		Node		B (Gbps)	Delay (ms)	Node		Node		B (Gbps)	Delay (ms)
IL	1	IT	9	0.155	24	DK	25	SE	2	10	3
IL	1	NL	23	0.155	31	SE	2	NO	3	10	3
TR	5	BE	20	0.155	27	ES	8	IT	9	10	21
FR	15	BE	20	2.5	7	ES	8	FR	15	10	8
GR	4	IT	9	2.5	22	FR	15	UK	22	10	4
GR	4	DE	19	2.5	24	FR	15	CH	14	10	4
SL	10	HR	11	2.5	3	DE	19	FR	15	10	4
HR	11	HU	12	2.5	4	CH	14	IT	9	10	4
AT	13	SL	10	2.5	4	CH	14	AT	13	10	4
HU	12	SK	18	2.5	3	AT	13	HU	12	10	4
SK	18	CZ	17	2.5	4	DE	19	AT	13	10	4
CZ	17	PL	21	2.5	9	DE	19	IT	9	10	8
PT	7	ES	8	2.5	9	CZ	17	DE	19	10	4
PT	7	UK	22	2.5	14	DE	19	NL	23	10	3
UK	22	IE	24	2.5	7	UK	22	NL	23	10	4
NL	23	BE	20	2.5	4	SE	2	DE	19	10	6
DE	19	IE	24	2.5	11	SE	2	UK	22	10	7
DK	25	NO	3	5	8	SE	2	PL	21	10	5

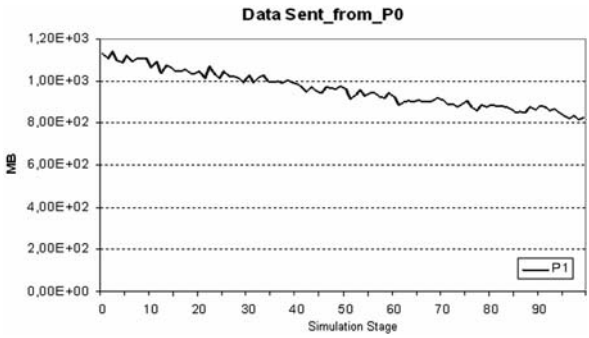


Fig. 15.7 Data sent from P0 to P1

power of 1 TFlops. Processors are connected to seven edge switches communicating through an InfiniBand switched fabric that guarantees a bandwidth of 10 Gbps between each pair of processors [11]. In particular, Figs. 15.7 and 15.8 report the amount of data exchanged by Processor 0 (P0) and Processor 1 (P1) at each stage of a simulation performed with $N_0 = 10^7$. It is relevant to highlight that due to the interaction model we are considering (see Fig. 15.5), P0 exchanges data only with P1, whereas P1 exchanges data with both P0 and P2. Graphs for P2 and P3 are not reported, because the trend is the same as P1 and P0, respectively.

Fig. 15.8 Data sent from P1 to P0 and P2

The goal is to analyze the behavior of the overall simulation time depending on the bandwidth reserved for communication among processors in the grid. In particular, taking into account the interaction paradigm among processors, it is necessary to establish three bidirectional LSPs (PO $\leftrightarrow$ P1, P1 $\leftrightarrow$ P2, P2 $\leftrightarrow$ P3), meeting bandwidth and delay constraints, by using the WCS path computation algorithm.

4.3 Simulation Results

Table 15.2 reports the parameters characterizing the allocated LSPs (ingress LER, egress LER, hops list, delay, setup time). Paths have been computed so as to satisfy their bandwidth requirements (from 0.5 to 15 Mbps) and to minimize the end-to-end delay. In all the simulations, performed by using MTENS (MPLS with Traffic Engineering Network Simulator) [9, 10], the LSPs are allocated along the same path. It is relevant to highlight that the LSPs setup time is in the order of tenths of milliseconds and therefore is negligible.

Table 15.2 Link metrics

Ingress LER	Egress LER	Hop list	Delay (ms)	Setup time (ms)
9	3	9_19_2_3	17	35
3	22	3_2_22	10	21
22	13	22_15_14_13	12	25

As clearly shown in Fig. 15.9, if the allocated bandwidth is greater than 2 Mbps, the grid assures better performance than a single processor. Moreover, even though the cluster outperforms the grid (with the same overall computational power), it is possible to significantly reduce the performance gap by increasing the amount of reserved bandwidth. For instance, in case the allocated bandwidth is equal to 15 Mbps, the difference is in the order of 50%. Moreover, since the amount of exchanged data is almost constant (see Figs. 15.7 and 15.8), it is not necessary to perform bandwidth negotiation at each stage. Therefore, the network administrator of a grid infrastructure, if requested, may assure a level of performance very close to the performance attainable in a cluster without a significant increase in maintenance costs.

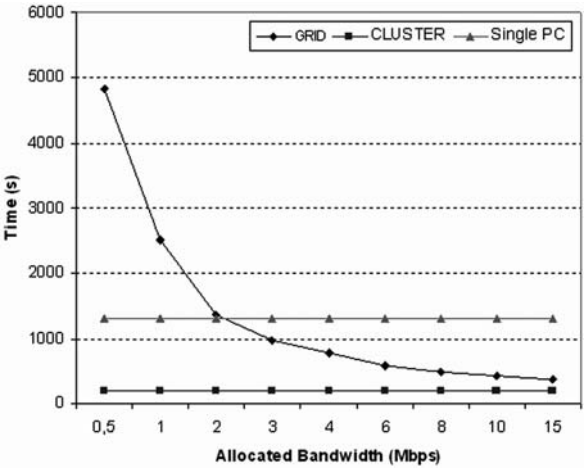


Fig. 15.9 Overall simulation time when $N_0 = 10^7$

5 Conclusions

The design and deployment of grid-enabled remote instrumentation services must handle the diversity of instruments and the critical nature of scientific equipment. In many cases, distributed data computing is strictly coupled with remote instrumentation services.

This chapter deals with the efficient parallel simulation of particles coagulation phenomena, which play a key role in some industrial and scientific applications. First of all, an analytical model for the particles coagulation process has been introduced. Due to the complexity of the problem, a direct simulation Monte Carlo algorithm has been used and an extensive simulation campaign has been carried out to investigate the efficiency of the proposed numerical approach. These simulations have been performed by means of a cluster located at Novosibirsk, Russia. Next, experimental data collected from these simulations have been used as input to properly dimension a grid infrastructure consisting of the same computational resources as the cluster interconnected by an IP/MPLS backbone. A performance analysis has been carried out to evaluate how the amount of bandwidth allocated in the network affects the behavior of the overall simulation.

Acknowledgments The work was supported by RFBR grants No. 06-01-00586 and No. 06-01-00046, President grant No. 587.2008.1 of "Leading scientific schools" program, INTAS grant No. 05-109-5267, Russian Science Support Foundation grant, RINGRID EU FP6 Project.

References

1. D. Adami, C. Callegari, S. Giordano, and M. Pagano. A new path computation algorithm and its implementation in NS2. In *IEEE International Conference on Communications*, Glasgow, Scotland, 2007, pp. 24–28, Jun. 2007.
2. D. Adami, N. Carlotti, S. Giordano, and M. Repeti. Design and development of a GNRB for the coordinated use of network resources in a high performance grid environment. In F. Davoli, S. Palazzo, and S. Zappatore, editors, *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*, pp. 295–307, Springer, New York, 2006.
3. M. Baker, R. Buyya, and D. Laforenza. Grids and Grid technologies for wide-area distributed computing. *Software–Practice and Experience Journal*, 2002.
4. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. *The physiology of the Grid: An open Grid services architecture for distributed systems integration*, June 2002. `http://www.globus.org/research/papers/ogsa.pdf`
5. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organization. *International Journal of Supercomputer Applications*, 2001. `http://www.globus.org/research/papers/anatomy.pdf`
6. F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. Multi-Protocol label switching (MPLS) support of differentiated services. In *RFC 3270*, May 2002.
7. M.A. Marchenko. Majorant frequency principle for an approximate solution of a nonlinear spatially inhomogeneous coagulation equation by the Monte Carlo method. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 21(3):199–218, 2006.
8. E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. In *RFC 3031*, Jan. 2001.

9. The OSPF–TE Network Simulator Home Page. http://netgroup-serv.iet.unipi.it/ospf-te_ns/
10. The RSVP–TE Network Simulator Home Page. http://netgroup-serv.iet.unipi.it/rsvp-te_ns/
11. Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996. http://dx.doi.org/10.1109/49.536364

Part III
Interactivity Management

Chapter 16
Practical Mechanisms for Managing Parallel and Interactive Jobs on Grid Environments

E. Fernández, E. Heymann, and M.A. Senar

Abstract The CrossBroker is a Grid Resource Management System that provides transparent and reliable support for the execution of parallel and interactive applications on a Grid environment. In this chapter, we outline the architecture of our system and describe the key mechanisms responsible for an efficient and reliable execution of parallel and interactive applications. We also present the job description language that allows the user to take profit from our system. The extensible and orthogonal mechanisms proposed give support for co-allocation of resources for various MPI implementations and the use of several tools and services for interaction with the application while running in a remote system.

Keywords Grid scheduling · Parallel · Co-allocation · Interactivity · CrossBroker

1 Introduction

Large-scale Grid computing requires a job management service that addresses new concerns arising in the Grid environment. This "job management" involves all aspects of the process of locating various types of resources, arranging these for use, utilizing them, and monitoring their state. In these environments, job management services have to deal with a heterogeneous multi-site computing environment that, in general, exhibits different hardware architectures, loss of centralized control, and, as a result, inevitable differences in policies.

E. Fernández (✉)
Departament d'Arquitectura de Computadors i Sistemes Operatius, Universitat Autònoma de Barcelona, Barcelona, Spain
e-mail: enol@aomail.uab.es

This work was done in the frame of the "int.eu.grid" project (sponsored by the European Union) and supported by the MEyC-Spain under contract TIN2007-64974.

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_16,

There are typically three main phases when scheduling a job over a Grid:

1. Resource discovery, which generates a list of potential resources to be used.
2. Information gathering on those resources and the selection of a best set.
3. Job execution, which includes file staging and cleanup.

The job management system that we have developed follows the same approach in scheduling jobs. However, our system, known as CrossBroker, is targeted to some applications that have received very little attention to date. Most existing systems have focused on the execution of sequential jobs. CrossBroker also supports interactive jobs and parallel applications mostly written with the MPI library, being able to take advantage of being executed on multiple grid sites.

From the scheduling point of view, support for parallel applications introduces the need for co-allocation. In the case of interactive applications, the main requirement is the possibility of starting the application in the immediate future, also taking into account scenarios in which all computing resources might be running batch jobs. Interactive applications also require the ability to control application output online and the ability to change the execution parameters while it is running.

The rest of this chapter is organized as follows: Section 2 briefly describes the architecture of the CrossBroker, Section 3 deals with the job specification language for jobs, Sections 4 and 5 describe the architecture that supports the execution of parallel and interactive applications, and Section 6 summarizes the main conclusions of this chapter.

2 Description of the CrossBroker Architecture

The architecture of the CrossBroker is tightly related to the software architecture of the CrossGrid [2] and int.eu.grid [7] projects. We restrict our discussion here only to those elements that are directly related to the execution of applications. When users submit their application, our scheduling services are responsible for optimizing scheduling and node allocation decisions on a user basis. Specifically, they carry out three main functions:

1. Select the "best" resources that a submitted application can use, taking into account the application requirements, as well as certain ranking criteria used to sort the available resources.
2. Perform a reliable submission of the application onto the selected resources (including co-allocation of resources if needed).
3. Monitor the application execution and report on job termination.

Figure 16.1 presents the main components that constitute our resource management services. Each grid site is composed of a cluster of machines consisting of a gatekeeper, called Computing Element (CE), and many worker nodes (WN) managed through a Local Resource Manager, such as PBS or Condor. A user submits a job to a Scheduling Agent (SA) through a User Interface (command line or Migrating Desktop). The job is described by a JobAd (Job Advertisement) using the

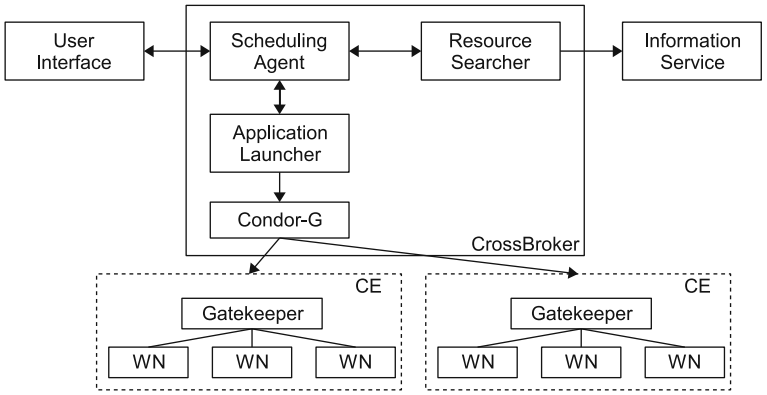


Fig. 16.1 CrossBroker resource manager architecture

Job Description Language (JDL) [11], which has been conveniently extended with additional attributes to reflect the requirements of CrossBroker applications.

Once the job has reached the SA, the Resource Searcher (RS) is asked for resources to run the application. The main duty of the RS is to perform the matchmaking between job needs and available resources. Using the job description as input, the RS returns as output a list of possible resources within which to execute the job taking into account the job requirements. The matchmaking process is based on the Condor ClassAd library [12], which has been extended with a set matchmaking capability [6]. The RS includes a repository of information decoupled of the matchmaking process, updated via an external Information Service, which can be built with Globus MDS [4] or RGMA [3]. Additional implementations for the Information Service can be supported by the use of an API included in the RS.

The SA then selects the best resource (or group of resources) from the list returned by the RS. The computing resources (or group of resources) are passed to the Application Launcher, which is responsible for the actual submission of the job. Due to the dynamic nature of the Grid, the job submission may fail on that particular site. Therefore, the Scheduling Agent will try other sites from the returned list until the job submission either succeeds or fails. The Application Launcher is also in charge of the reliable submission of parallel applications on the Grid.

Ideally, interactive and parallel applications should always run soon after submission. However, there may be situations where not all the remote resources involved in an execution are available, causing the ready resources to stay idle until all the subjobs start. The CrossBroker has a time-sharing mechanism that enables both interactive and batch jobs to share a single machine, in such a way that the interactive application starts its execution as soon as it is submitted (unless all resources in the Grid are busy executing other interactive applications) and proper co-allocation is ensured. This time-sharing scheme takes advantage of the Condor Glide-In [16] mechanism, which we extended and adapted to meet our requirements [5].

3 Job Description Language

The user describes the applications using a JDL file. The JDL is a Condor Classad [12] based language, where a set of attributes define the application to be executed. Listing 16.1 depicts an example of such file for an interactive and Open MPI application.

Listing 16.1 JDL job description

```
Executable                = "fusion_app";
Arguments                 = "-n";
JobType                   = "Parallel";
SubJobType                = "openmpi";
NodeNumber                = 12;
Interactive               = True;
InteractiveAgent          = "i2glogin";
InteractiveAgentArguments = "-p 21566:193.159.108.145";
InputSandBox              = {"fusion_app"};
Requirements              = other.GlueHostBenchmarkSI00 >= 1000;
Rank                      = other.GlueHostFreeCPUs;
```

The *Executable* and *Arguments* attributes specify the name and arguments of the application to be executed. The *InputSandBox* attribute is used to state the list of input files required for executing the application (in this example only the executable file will be transferred). Similarly, an *OutputSandBox* attribute allows the definition of output files that will be transferred from the Worker Nodes once the application has finished. The user can specify special requirements of its application with the *Requirements* attribute. Additionally, the *Rank* attribute allows the definition of preferences. Both attributes use the Glue Schema [1], since the resources are described using it. In this case, the user requires machines having a SpecInt 2000 benchmark better than 1,000 and prefers the sites with more free CPUs.

The *JobType* attribute specifies the type of application; currently, CrossBroker supports sequential, parallel [6], and workflow [10] as job types. Parallel jobs need some additional attributes:

- the *NodeNumber* attribute allows users to specify how many nodes their application will run on.
- the *SubJobType* specifies the kind of parallel application used. The SubJobType values currently supported are as follows:

  1. *openmpi*. Defines an application linked with Open MPI.
  2. *mpich*. An application linked with the MPICH library (ch_p4 device).
  3. *pacx-mpi*. An application that can run over multiple sites with PACX-MPI [9].
  4. *mpich-g2*. Inter-cluster application using the MPICH-G2 library [8].
  5. *plain*. An application that does not use any of MPI implementations above, the CrossBroker will only allocate the nodes.

- the optional *JobStarter* and *JobStarterArguments* attributes let the user specify his own method for starting the parallel job.

CrossBroker allows the execution of parallel applications on a grid running on multiple grid sites, thus using the set matchmaking capability of our Resource Searcher for the automatic search of resources. The set matchmaking capability [6] can be guided by the user with the use of the *Subjobs* attribute in JDL. This attribute specifies the different subjobs that one application is composed of and for each of them defines a number of nodes to use and any kind of special requirements. Listing 16.2 shows an example of an inter-cluster application requiring 10 CPUs, 5 of them in a cluster with machines more than 1 GBytes of RAM and 5 of them in a cluster with machines with OpenGL.

Listing 16.2 SubJobs specification in JDL

```
SubJobs = {
  [ NodeNumber = 5;
    Requirements = Member("OpenGL",
      other.GlueHostApplicationRunTimeEnvironment);]
  [ NodeNumber = 5;
    Requirements = other.GlueHostMainMemoryRAMSize >= 1024; ]
}
```

Independently of the type of application (parallel or sequential), the *Interactive* attribute defines the job as interactive and triggers the special scheduling mechanisms for such jobs. In order to provide real interactivity, application input has to be forwarded from the users (working on the executing machine) and their application (running on different nodes of the grid). The output must follow the reverse path. The *InteractiveAgent* and *InteractiveAgentArguments* attributes allow the user to specify which agent to use (see Section 5).

4 Managing Parallel Jobs

A parallel application to be executed on a grid can be executed either on a single site (intra-cluster parallel application) or on multiple sites (inter-cluster application). Whether using one or another approach depends both on the resources available and on the user-execution needs. The CrossBroker provides support for both types of applications by the use of *Application Launchers* and *JobStarters*.

Inter-cluster applications need a co-allocation service that ensures all the tasks composing the application are ready to start at the same time. The Application Launcher is the CrossBroker service responsible for providing this reliable co-allocation of applications on the Grid. Once the Scheduling Agent (SA) is notified that an inter-cluster parallel application needs to be executed, the matchmaking

process is performed to determine the site (or sites) for executing the application. When this is complete, the CrossBroker invokes the Application Launcher (AL) service with a set of resources to use for executing the parallel job. The Application Launcher follows a two-step commit protocol (detailed in [6]). This protocol ensures the proper co-allocation of the resources and that all the subjobs are started. The AL includes an API to define the details of the protocol that depend on the different applications and communication libraries. Currently, the CrossBroker provides Application Launchers for two MPI implementations created with this API, namely PACX-MPI [9] and MPICH-G2 [8]. The API allows the definition of:

- Submission method: the submission method defines how to start the subjobs in the remote resources; the Application Launcher provides Condor-G submission ready to use.
- Synchronization method: each MPI implementation uses different synchronization methods for the start-up of applications. MPICH-G2 uses Globus communication services and barriers, while PACX-MPI provides a *startup-server* for this purpose.
- Monitoring method: once the application is up and running, it must be monitored to check it finishes properly its execution. This monitoring includes also the error handling when problems arise.

Either if the application ends correctly or if there is any problem in the execution of any subjob, the AL records this in a log file that will be checked by the Scheduling Agent, which will take the correct action, in accordance with that information. This provides a reliable once-only execution of the application without user intervention. The events logged by the Application Launcher are classified as

- Scheduled event: all the subjobs are submitted and the application is waiting for the start of all the subjobs.
- Running event: the synchronization protocol has finished correctly and the application is running.
- Done event: the application has ended its execution correctly.
- Error event: once an error has appeared, the AL logs the kind of error and the remote site that has originated the error.

A helper application is used for starting the parallel application. This helper is called the *JobStarter*. The JobStarter deals with the different details of parallel application start-up that depend both on the MPI implementation and on the Local Resource Manager installed at each site. The CrossBroker has a clear interface for the JobStarter invocation composed of a set of environment variables.

The CrossBroker uses MPI-START [14] as default JobStarter for MPI jobs. MPI-START is usually installed at the sites, but in case of not being found, the CrossBroker launcher script will download it to the site automatically. MPI-START has a pluggable architecture for supporting different MPI implementations and Local Resource Managers. The most common cases used in int.eu.grid: Open MPI, PACX-MPI, or MPICH as MPI implementations with PBS, SGE, or Condor as Local

Resource Manager are fully supported by MPI-START. The user may include a different JobStarter for his jobs. This is done via the *JobStarter* and *JobStarterArguments* in the JDL file.

5 Managing Interactive Jobs

One of the main requirements for the application to be interactive is to provide a possibility for starting it in a predictable future. This requirement is very hard to fulfil in most existing Grid infrastructures, due to their batch-oriented system of submission. The CrossBroker, in contrast, includes particular features to support interactive-oriented submission of jobs, specially a job multiprogramming mechanism designed specifically for a fast start of applications [5].

Although applications are executed on remote sites, the input/output of such applications should be controlled locally, that is, from the submitting machine. This way, users can interact with their applications that are executing remotely. The CrossBroker defines an architecture for execution of interactive applications using a split execution system.

Interactive sessions are handled by a Grid Console (GC) for obtaining mostly continuous input/output from remote programs running on an unreliable network. A Grid Console is a split execution system made up of two software components: an agent (Console Agent – CA) and a shadow (Job Shadow – JS). Standard input messages go from the Job Shadow to the Console Agent, which then forwards these to applications, while both standard output and standard error messages follow the reverse path. Under this arrangement, a program can run on any networked machine and still execute exactly as if it were running on the same machine as the shadow.

We include a Condor Bypass [15] implementation of Grid Console. Any other implementation can be used under this schema; glogin [13] has also been adapted to use it as Console Agent. In the JDL shown in Fig. 16.1, an example of glogin usage is shown. The port and machine used as Job Shadow are specified in the InteractiveAgentArguments attribute of the JDL with the −p option.

Interactive Agents can be used in conjunction with parallel jobs. In this case, the CrossBroker will specify to the *JobStarter* that an agent should be used for starting the application. Figure 16.2 shows an example with all the components involved in an inter-cluster MPI and interactive application execution. The user starts the Job Shadow in his local machine and submits a JDL to the CrossBroker, arrow (1) in Fig. 16.2. The CrossBroker will start the specific Application Launcher for the MPI implementation used that will submit the subjobs to each site. This is illustrated by arrow (2) in Fig. 16.2. In this example, two sites will be used, one with processes from 0 to K and the other with processes from K+1 to N. The JobStarter will start the job in each site and will use the InteractiveAgent parameter to start the Console Agent in the first site. Only one Console Agent is required because the MPI handles internally the I/O of the whole application and redirects it to the first process. Once the application starts, the Job Shadow and Console Agent start to communicate with each other, as illustrated by arrow (3).

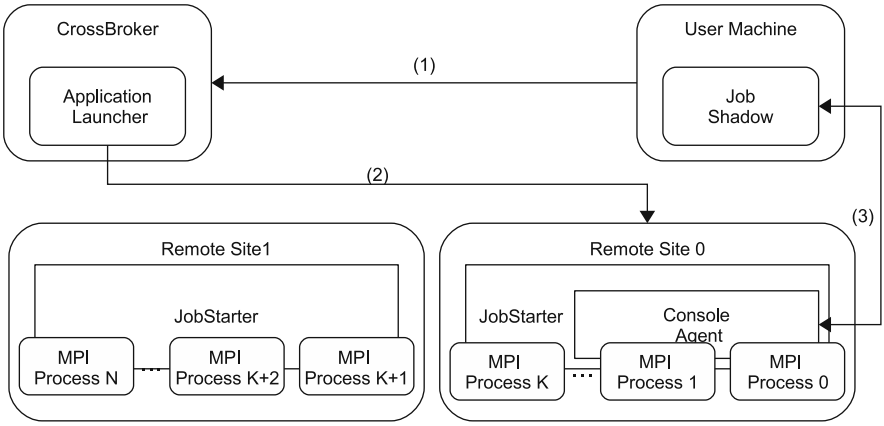


Fig. 16.2 Execution of an inter-cluster MPI interactive job

6 Conclusions

Traditionally, many scientific applications submitted to a Grid are executed in a pure batch mode. The execution of interactive and parallel applications in grid environments has received little attention to date. We have presented an architecture for a grid scheduler that allows the execution of such jobs. The implementation of this architecture is the CrossBroker. We have described the main CrossBroker components and the mechanisms that provide automatic and reliable support for interactive and MPI jobs over grid environments.

References

1. S. Andreozzi, M. Sgaravatto, and M.C. Vistoli. Sharing a conceptual model of grid resources and services. *The Computing Research Repository*, Jun. 2003.
2. M. Bubak, M. Malawski, and K. Zajac. The crossgrid architecture: Applications, tools, and grid services. In *Proceedings of the European Across Grids Conference*, pp. 309–316, 2003.
3. A.W. Cooke, A.J.G. Gray, W. Nutt, J. Magowan, M. Oevers, P. Taylor, R. Cordenonsi, R. Byrom, L. Cornwall, A. Djaoui, L. Field, S. Fisher, S. Hicks, J. Leake, R. Middleton, A.J. Wilson, X. Zhu, N. Podhorszki, B.A. Coghlan, S. Kenny, D. O'Callaghan, and J. Ryan. The relational grid monitoring architecture: Mediating information about the grid. *Journal of Grid Computing*, 2(4):323–339, 2004.
4. K. Czajkowski, C. Kesselman, S. Fitzgerald, and I. T. Foster. Grid information services for distributed resource sharing. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, pp. 181–194, 2001.
5. E. Fernández, E. Heymann, and M.A. Senar. Resource management for interactive jobs in a grid environment. In *Proceedings of the 2006 IEEE International Conference on Cluster Computing*, p. 10, 2006.
6. E. Fernández, E. Heymann, and M.A. Senar. Supporting efficient execution of mpi applications across multiple sites. In *Proceedings of the 12th International Euro-Par Conference*, pp. 383–392, 2006.
7. int.eu.grid Project web site. `http://www.interactive-grid.eu/`

8. N.T. Karonis, B. Toonen, and I. Foster. Mpich-g2: A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, 63(5):551–563, 2003.
9. R. Keller, E. Gabriel, B. Krammer, M.S. Müller, and M.M. Resch. Towards efficient execution of mpi applications on the grid: Porting and optimization issues. *Journal of Grid Computing*, 1(2):133–149, 2003.
10. A. Morajko, E. Fernández, A. Fernández, E. Heymann, and M.A. Senar. Workflow management in the crossgrid project. In *Proceedings of the European Grid Conference*, pp. 424–433, 2005.
11. F. Pazini. *Jdl attributes specification*, 2007.
12. R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pp. 140–146, 1998.
13. H. Rosmanith and D. Kranzlmüller. Glogin – a multifunctional, interactive tunnel into the grid. In *Proceedings of the 5th International Workshop on Grid Computing (GRID 2004)*, pp. 266–272, 2004.
14. S. Stork and K. Dichev. *Mpi-start*. `http://savannah.fzk.de/projects/mpi-start/`
15. D. Thain and M. Livny. Multiple bypass: Interposition agents for distributed computing. *Journal of Cluster Computing*, 5:39–47, 2001.
16. D. Thain, T. Tannenbaum, and M. Livny. *Condor and the Grid*. Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons Inc., Apr. 2003.

Chapter 17
Int.eu.grid

A Grid Infrastructure for Interactive Applications

G. Borges, J. Gomes, M. Montecelo, M. David, B. Silva, N. Dias, J.P. Martins, C. Fernández, L. García-Tarrés, C. Veiga, D. Cordero, J. López, J. Marco, I. Campos, D. Rodriguez, R. Marco, A. López, P. Orviz, A. Hammad, M. Hardt, E. Fernández, E. Heymann, M.A. Senar, A. Padee, K. Nawrocki, W. Wislicki, P. Heinzlreiter, M. Baumgartner, H. Rosmanith, S. Kenny, B. Coghlan, P. Lason, L. Skital, J. Astalos, M. Ciglan, M. Pospieszny, R. Valles, and K. Dichev

Abstract The int.eu.grid project aims to provide a production quality grid computing infrastructure for e-Science supporting parallel and interactive applications. The infrastructure capacity is presently about 750 cores distributed over 12 sites in 7 countries, and it implies a much larger management and coordination effort to guarantee high availability and fast start-up mechanisms of (batch and parallel) interactive applications. Int.eu.grid is also strongly focused on providing a user-friendly access to the grid through a powerful GUI allowing a transparent access to grid resources and the possibility to interact in real time with running applications. In this chapter we present the management policies, infrastructure services, and operation mechanisms that enable interactivity over the grid bringing a new added value to grid environments.

Keywords Grid computing · Interactivity · Resource discovering · Fast start-up

1 Introduction

The Interactive European Grid project (int.eu.grid) [7] started in April 2006 and operates an advanced production quality grid infrastructure distributed over 12 sites (clusters) in 7 countries, all connected through the GÉANT Network. The project is focused on providing user-friendly access to more than 200 users, registered in 13 project virtual organizations, with demanding parallel and interactive applications and immediate execution needs. The Collaboration emerged in a context where

G. Borges (✉)
Laboratório de Instrumentação e Física Experimental de Partículas, Lisboa, Portugal
e-mail: goncalo@lip.pt

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_17,

interactivity and visualization were heavily needed over the grid to establish a feedback channel between users and their applications. Users wanted to see what was going on with their tasks and demanded answers in seconds and not in hours.

Although largely based on the EGEE [2] architecture, the int.eu.grid new features are materialized in specific software layers acting on top of gLite [5] middleware, bringing a completely new added value to the whole grid environment. Nevertheless, although delivering a valuable service to its users, these new features also carry a huge support effort, which standard grid projects do not have to deal with, representing a challenge from the infrastructure management point of view, both for local clusters and for central services.

A strong effort has also been put in establishing infrastructure interoperability scenarios and in creating technical solutions to enable virtual organizations to operate across int.eu.grid and other major grid infrastructures at the expense of minimum configurations changes. This allows external multi-science applications to profit from the coordinated effort to support MPI and interactive environments over the grid.

This chapter describes the int.eu.grid infrastructure, covering areas such as its architecture, new deployed services, and mechanisms for the operation coordination.

2 Infrastructure Architecture

The int.eu.grid architecture, depicted in Fig. 17.1, relies on the same basic building blocks as EGEE. It proposes the deployment of a set of local services, which must

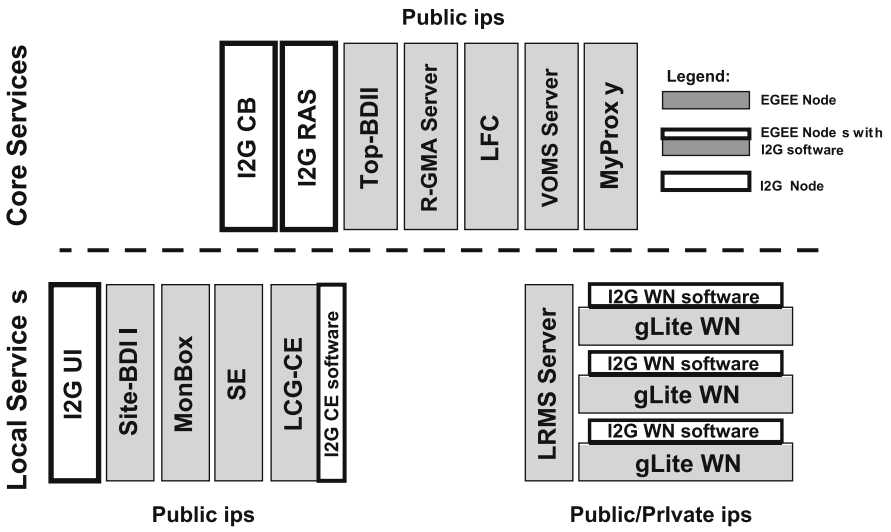


Fig. 17.1 Int.eu.grid core and local services

be present in all sites, and a set of core services, installed only in some institutes but used by all. However, to enable interactivity and the possibility to run MPI parallel applications on grid environments, the int.eu.grid collaboration has enhanced pre-existing EGEE services and deployed some additional ones. The following set of specific int.eu.grid core services must be in place:

- The *CrossBroker* [3] represents the heart of the int.eu.grid workload management system providing efficient job management scheduling for interactive and parallel jobs inside clusters and across clusters. The CrossBroker matches jobs to sites according to pre-defined criteria including local bandwidth, an important feature for users expecting to stream data to/from their running applications. It is also able to properly trigger the interactivity mechanisms developed inside the project to enable visualization and data streaming when they are invoked by the users. The CrossBroker was initially developed by the CrossGrid [6] project and is based on the LCG [10] Resource Broker and gLite WMS.
- The *Roaming Access Server (RAS)* is a service that acts as a relay between the grid and the *Migrating Desktop (MD)*, a thin Java client enabling end users access to grid resources through a graphical interface. Together, the MD and the RAS are two key components of the int.eu.grid approach to an integrated user-friendly grid environment. Both were initially developed within the CrossGrid [6] project and have been extensively improved in the int.eu.grid context.
- The remaining core services (Top Berkeley Database Information Index (Top-BDII), MyProxyServer (PX Server), Logical File Catalogue (LFC), Virtual Organization Membership Service (VOMS) and R-GMA server) are used and deployed as in the EGEE infrastructure without any additional change (apart from configuration aspects needed to fit int.eu.grid specific needs).

Regarding local services, each site must deploy a User Interface (UI), a LCG Computing Element (lcg-CE), a Storage Element (SE), a Monitoring Box (Mon-Box), and several Worker Nodes (WNs). The strongest difference comes in the UI, which provides all the int.eu.grid workload management client command tools to properly interact with the CrossBroker. Other services such as the lcg-CE and WNs just have to support a thin int.eu.grid software layer on top of the ordinary gLite services. This software layer deploys and installs the interactive agents (*(i2)glogin* [14] *and GVID* [13]), the MPI implementations supported within the project (*Open-MPI* [4] *and PACX-MPI* [8]), and the *MPI-START* [11] mechanism, a solution developed inside the int.eu.grid project to encapsulate the MPI, batch system, and cluster-specific details.

3 Infrastructure Interactivity Framework

Int.eu.grid's main focus is to enable interactive and parallel user-friendly computing over the grid maintaining the same ordinary features as other gLite-based infrastructures and trying to achieve this as seamlessly as possible. Because this chapter

is devoted to report on interactivity achievements over grid environments, we will emphasize the user-friendly access and interactivity properties (two features well connected) of the int.eu.grid infrastructure, not mentioning its powerful enhancements regarding parallel computing.

3.1 Friendly User Access

A major int.eu.grid novelty is a *Graphical User Interface (GUI)* to access the grid resources. Users can choose between the standard command line tools based on a gLite User Interface enhanced with the int.eu.grid middleware and an exclusive int.eu.grid GUI *called Migrating Desktop (MD)* [9]. While the gLite classic UI is a good solution for developers working in porting applications, since it combines the Linux development environment with the gLite and int.eu.grid capabilities, the MD GUI is a good solution for the end user who wants to focus on achieving his or her scientific goals, without bothering with the underlying grid details.

This MD is a Java-based GUI with many advantages over the command-line interface: the contents of the storage elements can be navigated in a file browser, files can be up/downloaded from and to the local computer, complex job submissions can be stored as shortcuts on a Virtual Desktop and then submitted when desired, overcoming the steep learning curve that the non-expert user has to face. Because it does not access the grid directly but over the network, connecting to a special machine called *Roaming Access Server (RAS)*, which acts as a bridge between the MD and the infrastructure, the MD does not depend on grid middleware being locally installed. Therefore, it allows accessing the grid from a diverse set of platforms and operating systems if a Java Virtual Machine is installed.

In a more technical level, the MD GUI and RAS work together in a client/server fashion. The RAS is a core service installed at a central location, with direct network connectivity as any regular server, while the MD is a client, which connects to the RAS and can be run from the private workstations of the grid users (as long as it can connect to the appropriate RAS ports). The RAS offers a well-defined set of Web services that represent the integration level for interfaces of different middleware. It provides services like job submission, job monitoring, interactive session manager, and job channel forwarding. Indeed, MD/RAS supports interactivity and real-time visualization of the jobs. For these to be possible it is needed to develop a specific plugin for each application willing to use the MD. This plugin acts as an interface enabling the bidirectional communication between a certain job and the local GUI.

The MD and RAS combination addresses issues still considered open in grid environments: it is a powerful GUI that can be started using Java from any desktop, platform, or portable computer independently of its location, thus making the grid easily available to non-experts. The MD/RAS server version in production supports Java 1.6 and enables the submission of batch, interactive, and parallel jobs using both OpenMPI and PACX-MPI (among other data management and job management grid operations).

3.2 Interactivity

The int.eu.grid *interactivity* features are implemented around the *glogin* [14] tool, which enables the users to open secure GSS TCP communication channels between grid nodes, similar to the more traditional SSH or the older RSH tools. glogin was initially developed on the CrossGrid [6] project and has been extended in int.eu.grid (*i2glogin*) to execute on GT4 middleware. Another main difference is that while glogin is submitting itself during the execution of the local instance invoked on the User Interface, i2glogin needs to be submitted using grid middleware mechanisms, commonly as a job described by a JDL file, delivering the control of the submission mechanisms to the middleware. i2glogin is fully integrated in the MD/RAS framework: the local instance of i2glogin is started on the RAS, which acts as server, and waits for incoming connections from the grid. The client part of i2glogin is submitted as grid job using a JDL where the connection point (server:port) of the local instance has to be specified as an argument. Upon startup of the grid job it connects back to the local instance and creates a secure, low-latency, bidirectional connection.

Important aspects for judging the quality of the interactivity are the speed of the data transmission as well as security provided by encryption and user authentication/authorization. These criteria are fulfilled by i2glogin and hence make it a useful generic interactivity tool.

3.3 Visualization

The visualization functionalities are based on the compression of the OpenGL [12] graphics output generated by the applications running in the clusters, which is then sent to the client through a glogin communication channel. GVID [13] performs the encoding and decoding of the graphics on both sides. To integrate it in the MD/RAS framework and follow the requirements of MD application plugins, a Java implementation of the GVID display client was developed, resulting in a Java package ready to be used by grid application developers.

The user input events such as mouse clicks generated through the application-specific MD plug-in are sent via RAS to the application running in the remote cluster. The application itself has to be adapted to react to these input events. Through this software, the job's input and output streams are sent back and forth inside this "pipe," and the user can thus watch locally in real time (as soon as it is produced) the output produced remotely, instead of waiting for the job to finish as with batch jobs. Also, the user can control the job in real time, for instance to change parameters of a simulation while it is still running.

Figure 17.2 presents how the user-friendly access, the interactivity, and the visualization frameworks are integrated in the int.eu.grid infrastructure. This concrete example shows the fusion workflow, an interactive parallel application, which simulates the thermonuclear processes taking place inside fusion devices.

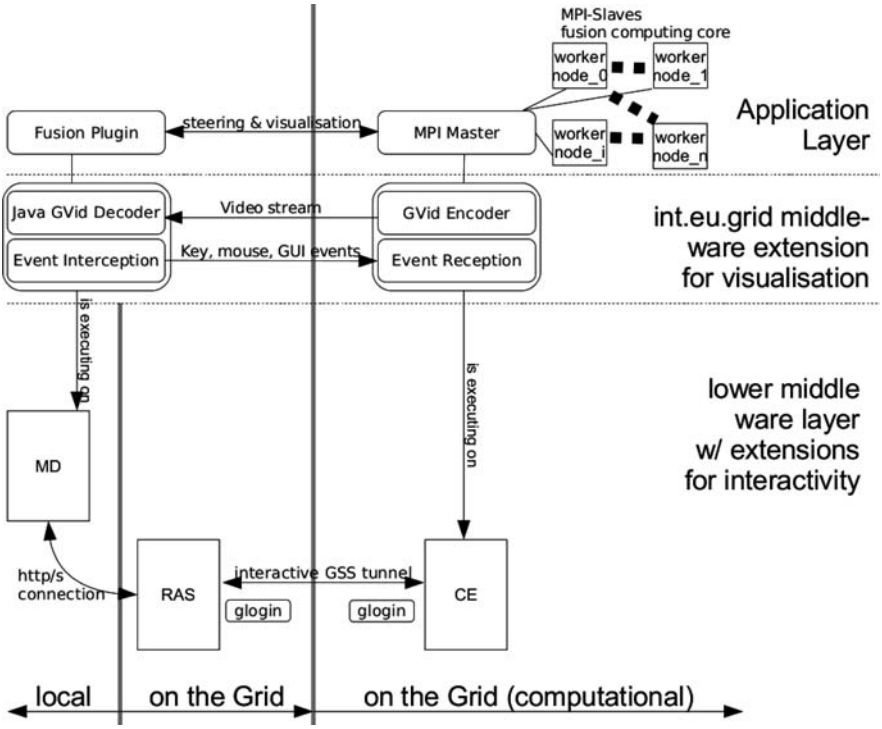


Fig. 17.2 glogin, GVID, and MD/RAS integration in int.eu.grid infrastructure (for the fusion application)

3.4 CrossBroker and Time Sharing Mechanisms

The CrossBroker supports interactivity for all types of sequential and MPI flavored jobs. Through extraordinary JDL variables, the user can define at submission time which interactive agent (and its arguments) should be used for the data streaming. If the job is recognized as interactive, the CrossBroker injects the chosen interactive agent (i2glogin) in a transparent way for the user and treats the task with higher priority. Interactive jobs are always sent to sites with free machines since they imply immediate availability of resources. However, the fact that the clusters' information is updated with a delay of a few minutes by the central information system represents a big drawback. If interactive jobs trigger this interval, they may eventually fail due to the incorrect and delayed published information.

If there are no available machines, the CrossBroker can activate a time-sharing criterion. This is a new major functionality introduced by the latest int.eu.grid middleware release, taking advantage of the *Condor glidein* mechanism [1]. When this functionality is active, Condor glidein instances are attached to the batch jobs submitted by the CrossBroker. Once started in the Worker Nodes, they take control of the remote machine independently of its LRMS, generate two logical virtual

machines, and register themselves back to it as usable computing resources. One virtual machine is used for the execution of the original batch job where the glidein instance was wrapped on, while the other can only be taken by an interactive job. The CrossBroker can submit interactive jobs directly to these free resources bypassing the gatekeeper and enabling a much faster start-up time for interactive job submissions. Furthermore, this feature allows interactive jobs to run even when a given cluster is fully occupied. The action to be performed on the running batch job when the interactive job arrives can be defined on a VO basis and can result on its death, suspension, or execution priority decrease.

Table 17.1 presents the quantitative results of a study evaluating the job start-up times through different submission chains:

- Standard job batch submission to free resources;
- glidein submission (wrapped in batch jobs) to free resources;
- Job submission to logical virtual machines built by running glidein instances.

Table 17.1 Condor glidein and response times

Mechanism	Resource searching	Resource selection	Submission to	
			Campus grid	Remote site
Free machine submission	3 s	0.5 s	17.2 s	22.3 s
Glidein submission to free machine	3 s	0.5 s	29.3 s	33.3 s
Virtual machine submission	0.5 s		6.8 s	8.1 s

One can conclude that the initial glidein submission takes a long time, but once glidein instances are in place in the WNs, it improves the start-up mechanism for interactive jobs by a factor of more than 3.

3.5 lcg-CE, JobManagers, and LRMS Configurations

Beyond the integration of int.eu.grid interactive tools in the infrastructure and the time sharing developed in the CrossBroker, some site services, such as the lcg-CE and LRMS, can be configured and tuned to optimize job scheduling and overall execution performance. At this local level, the main issue that can be addressed is the implementation of fast start-up mechanisms, i.e., the possibility of starting the application immediately.

The project interactivity framework is complemented by the fact that some sites have decided to move to the pbs JobManager, a much faster (but less reliable) mechanism as compared to the standard lcgpbs JobManager. In EGEE infrastructures, a direct submission to a remote site completes in less than 25 s (compared to around 2 min for lcgpbs), and a job submitted through the EGEE WMS completes in just over 2 min compared to around 5–6 min for lcgpbs JobManager. Nevertheless, the

"lcg" job managers have been coded for scalability instead of speed, meaning less CPU consumption when the CE handles many hundreds of jobs in parallel. The "lcg" job managers were coded to be more robust when things start to go wrong.

Another implemented solution was to enable LRMS mechanisms for immediate job execution following the *short deadtime job (sdj)* [15] approach proposed by EGEE.

4 Conclusions

We have presented the tools developed by the int.eu.grid project and their integration in the infrastructure services that enable an interactivity framework in grid environments. The project brings a new added value to the grid with its user-friendly access, interactivity and visualization features, and fast start-up mechanisms. However, apart from the technical advantages, these developments also bring a complete new set of issues never raised before, both on the operation management of the infrastructure and on the political agreements between project and VOs.

On the operation management part of the infrastructure, new services and monitoring chains had to be developed to closely monitor the infrastructure. Regarding the Service Level Agreements with VOs, a generalized priority schema and business model for grid users and jobs is still missing. However, a general concept is widely accepted: Users should run interactive jobs with limitations. Otherwise, nobody will run batch or parallel jobs if interactive jobs run immediately at no cost for the user. Interactive resource usage has to be "paid" somehow and could be implemented through a Fair Share policy with different penalty factors:

- Batch jobs worsen the priority according to the resources used.
- Interactive jobs worsen the priority faster than batch ones.
- If batch and interactive jobs run in the multiprogramming schema, priority worsening is adjusted according to the amount of CPU consumed by each one.

Things would be easier if the infrastructure remained simple (single Broker, uniform middleware, same LRMS, same policies, etc.), but normally users and institutions will try to interoperate among several grids (interactive grid will be one of them), meaning multiple brokers, multiple LRMS, different policies, etc. Some solutions are under investigation within the project, such as the development of an interactive CE, where the automatic injection of glidein wrappers for all jobs submitted from other sources, different from the CrossBroker, could be implemented.

References

1. Condor Glidein. `http://www.bo.infn.it/condor-mirror/manual/v6.6/condor_glidein.html`
2. Enabling Grid for E-Science project. `http://www.eu-egee.org/`

3. E. Fernández, E. Heymann, and M.A. Senar. Supporting efficient execution of MPI applications across multiple sites. In *Lecture Notes in Computer Science*, vol. 4128, pp. 383–392, 12th International Euro-Par Conference, Dresden, Germany, 2006.
4. E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and T.S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In D. Kranzlmüller, P. Kacsuk, and J.J. Dongarra, editors, *Proceedings of the 11th European PVM/MPI Users Group Meeting, Lecture Notes in Computer Science (LNCS)*, vol. 3241, pp. 97–104, Budapest, Hungary, 2004.
5. gLite middleware. `http://glite.web.cern.ch/glite/`
6. J. Gomes, M. David, J. Martins, L. Bernardo, A. Garcia, M. Hardt, H. Kornmayer, R. Marco, D. Rodriguez, I. Diaz, D. Cano, J. Salt, S. Gonzalez, J. Sanchez, F. Fassi, V. Lara, P. Nyczyk, P. Lason, A. Ozieblo, P. Wolniewicz, M. Bluj, K. Nawrocki, A. Padee, W. Wislicki, C. Fernandez, J. Fontan, Y. Cotronis, E. Floros, G. Tsouloupas, W. Xing, M.D. Dikaiakos, J. Astalos, B. Coghlan, E. Heymann, M. Senar, C. Kanellopoulos, A. Tirado Ramos, and D.J. Groen. Experience with the international testbed in the CrossGrid project. In *Advances in Grid Computing – EGC 2005, LNCS 3470*, pp. 98–110, Amsterdam, The Netherlands, February 2005.
7. Interactive European Grid project.
8. R. Keller, E. Gabriel, B Krammer, M.S. Muller, and M.M. Resch. Towards efficient execution of MPI applications on the Grid: Porting and Optimization issues. *Journal of Grid Computing*, 1(2):133–149, 2003.
9. M. Kupczyk, R. Lichwała, N. Meyer, B. Palak, M. Płóciennik, and P. Wolniewicz. 'Applications on demand' as the exploitation of the Migrating Desktop. *Future Generation Computer Systems*, 21 (1):37–44, Jan. 2005.
10. LHC Computing Grid project. `http://lcg.web.cern.ch/LCG/`
11. MPI-START. `http://savannah.fzk.de/projects/mpi-start/`
12. OpenGL project. `http://www.opengl.org/`
13. M. Polak and D. Kranzlmüller. Interactive Videostreaming Visualization on Grids. In J. Dongarra and B. Tourancheau, editors, *Cluster and Computational Grids for Scientific Computing, Future Generation Computer Systems, Special Section*, vol. 24, pp. 39–45, Elsevier Science, Amsterdam, The Netherlands, Jan. 2008.
14. H. Rosmanith and D. Kranzlmüller. glogin – A Multifunctional Interactive Tunnel into the Grid. In R. Buyya, editor, *Proceedings of the Grid 2004, 5th IEEE/ACM International Workshop on Grid Computing*, pp. 266–272, IEEE Computer Society, Pittsburgh, PA, USA, Nov. 2004. ISBN 0-7695-2256-4.
15. Short Deadline Jobs. `http://egee-intranet.web.cern.ch/egee-intranet/NA1/TCG/wgs/SDJ-WG-TEC-v1.1.pdf`

Chapter 18
Interactivity in Grid Computing in the Presence of Web Services

H. Rosmanith and J. Volkert

Abstract Nowadays, we experience a few research projects which concentrate on providing interactivity for grid computing. Interactivity allows event-based processing of dynamic data created while a grid job is running. This technique represents an antipode to the ruling batch processing paradigm. Batch processing means: Submit a job to a queue, process it and wait for the job's termination. Only after the job has been finished it is possible to analyse the output. Current grid middleware, such as the Globus Toolkit 4 (GT4) or Unicore 6, offers a Web services based interface to access their services, thus we have to investigate how we can provide interactivity in such an environment. In this chapter, we show a possible solution which has been proven to work by an implementation for GT4.

Keywords Grid computing · Interactive jobs · Web services

1 Introduction

Today's grid computing [9] mainly focuses on noninteractive batch jobs. Grid middleware was developed having in mind long-running, compute intensive jobs, which perform their work entirely off-line. In such a computing environment, there is no possibility for changing parameters during the runtime of the job. Only when the job has been terminated, is it possible to examine the output in a post-mortem fashion. This can, in the worst case, lead to a waste of valuable computing and research time.

This disadvantage can be resolved by enabling the user to interact with the application while it is still executing on the grid. To demonstrate this, let us assume that a long-running numerical simulation has been started in an interactive grid. In such an environment, the user is able to observe the job's output, analyse intermediate results and improve the model as needed. Instead of waiting for the job's completion, the user can restart it prematurely, should the simulation develop into a wrong or useless

H. Rosmanith (✉)
Interactive European Grid Project, GUP Linz, Austria
e-mail: hr@gup.uni-linz.ac.at

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_18,

direction. If modifying parameters is allowed too, the user can steer the simulation and thus avoid a costly restart of the entire job.

This chapter is organised as follows: Section 2 examines interactivity and identifies crucial features missing in grid middleware. Then, two solutions are demonstrated: In Section 3.1, the submission mechanism present in the *glogin* tool is ported to make use of Web Services, while in Section 3.2 a dedicated Web Service providing interactive connections is developed using a sophisticated technique.

2 Exploring Interactivity

Today, the term "interactivity" is used not only in computer science, but in other areas as well, such as, for instance, biology and medicine. Originally, the term was coined in sociology, denoting humans mutually influencing each other by actions. But what is interactivity in computer science? By interactivity we mean communication between the program and the user. In [24], six different levels of interactivity are distinguished: From pure observation of program output (the lowest level 1) to manipulation of the program with feedback (the highest level 6). Since level 6 includes all the other levels and level 6 is needed in many cases, it is clear that this level should be implemented.

Are there any special properties interactivity should have? In [4], several attributes of interactivity are defined. Although the type of interactivity was discussed with human interaction in mind, it can also be used to describe human–computer interaction. A particular attribute which specifically applies to an interactive environment is "limited look-ahead": This means that while interaction is going on, the input to be expected of an interaction partner is unknown to the other partner. During interaction, both partners increase their knowledge of each other.

Similarly, this is true for interactive data processing: Neither the program nor the user know what data will be sent by the other side, adding a "surprise value" to the data. For instance, the output of a complicated scientific simulation is usually unknown to the scientist – otherwise the simulation would not be necessary at all. On the other hand, strolling around in an already well-known virtual world is not very interactive from this point of view.

From this it follows that grid middleware, which requires that the complete input data set to a job is already available even before the job starts, can be classified as noninteractive grid middleware.

To allow users to interact with a grid job, a communication channel for sending back data is required. In combination with the output data channel, which sends data from the job back to the user, both channels form a bidirectional data flow. To assure the quality of interactivity, data has to be transferred fast enough.

The conclusion is that two basic properties important for interactivity are speed and bidirectional data flow.

Speed: With respect to speed, how can a fast data transport be ensured? Both communication partners require data exchange at a speed sufficient to satisfy the problem's requirements. Interactive data needs to be exchanged fast enough, but it

is not necessary to go faster. It is useless that a graphical application operates at a frame rate of 1,000 frames/s, when the user is satisfied by 30 frames/s.

A common obstacle found in grid computing which hinders the implementation of a fast data transport is a mechanism known as "polling", which means that the advent of data is queried in a loop. Unfortunately, using this mechanism has become quite popular these days. If done frequently, this leads to a system wasting valuable computing resources. To reduce the impact of the polling mechanism on the system, the query interval is set to a pretty high value, for instance, the Globus Toolkit [8] polls for new data every 10 s. While this reduces the impact of polling, it leads on the other hand to a less responsive system.

This problem can be addressed by a feature which has always been known in interactive programming: Event-based processing. Data is processed as it arrives. When there is no data available, the program is put asleep. This guarantees an almost immediate reaction on the reception of an event, usually faster than any reasonable polling interval, while at the same time it keeps system load on a minimum.

Bidirectional data flow: In order to allow the user to interact with a program, data from the input devices such as keyboard, pointer or readings from laboratory apparatus such as microscopes, need to be sent to the program while it is still executing. In order to display data the program is currently generating, an output channel is required, too. Only when both input and output channels are present, we can talk of a bidirectional data flow. The combination of input and output channel, also known as forward and feedback channel, forms an "Elementary Interaction Loop" (EIAL) [5]. In interactive grid computing, the EIAL is expanded to include grid resources or even cross grid boundaries.

3 The Approach of Web Services

Examining current popular middleware, we observe a transition to Web Services [1], also including the submission of jobs. When the Globus Toolkit finally switched from version 2 to version 4, the proprietary, pre-Web Service protocol (pre-WS GRAM) was replaced by a protocol using Web Service interfaces (WS GRAM). Unicore [7] was refined in the same way: The pre-Web Services version of this middleware was modified so that now version 6 uses Web Services. In order to keep up with this development, it is necessary to examine how to support interactivity in the Web Services environment. The following sections deal with this question.

3.1 A Separate Bidirectional Data Channel

When using the Globus Toolkit 2 (GT2), an interactive connection can only be obtained by providing a separate data channel. The reason is that once the *globus-gatekeeper* has started a *globus-job-manager*, data from the command to execute always passes through the job-manager. The job-manager starts the command with the standard output (stdout) file-descriptor redirected to the GASS-cache, using *fstat(2)* to determine whether the file has changed. This led to the development of the

glogin [21, 22] tool, which implements a secure bidirectional (thus interactive) data channel to provide features such as pseudoterminals (pty), standard input, output and TCP port forwarding.

For better understanding, a short description of *glogin*'s mode of operation follows. To create the interactive data channel, *glogin* allocates a TCP port and invokes a remote copy of itself on the grid, passing information about the TCP port to the remote copy. The remote copy allocates another TCP port and writes its connection parameters to standard output. Then, the remote copy creates a child process and terminates. This is required since otherwise the output data is stuck in the GASS-cache for 10 s, thus delaying command execution for this amount of time. Unfortunately, this revealed a weakness in local schedulers [20]. Both local and remote *glogin* processes try to establish a connection to each other. The connection established first is used as interactive data channel, while the second connection is discarded. Data transported over this connection are secured using the GSSAPI [15, 29]. Details can be found in [21].

With the advent of the Globus Toolkit 4 (GT4), *glogin* has been expanded to additionally make use of Web services. For command execution, *glogin* utilises the same services as *globusrun-ws*, which are the *ManagedJobFactory Service*, the *ManagedJob Service*, the *ManagedExecutable Job Service*, the *Delegation Service* and the *Subscription Manager Service*. With these services, a remote copy of *glogin* is invoked to establish the secure interactive channel. Once the channel is available, *glogin* works as already described.

globusrun-ws offers "data-streaming". The term implies a constant flow of data, suggesting that *globusrun-ws* is an interactive tool. However, this is not the case. The reason is that data for its connection is not transported using the event-based approach. Rather, it is using *gsiftp* to poll for the advent of new data every 10 s. Furthermore, a feedback channel for interactively supplying data from user to the job is not present; thus, a bidirectional connection is not available.

3.2 Providing a Dedicated Web Service

One disadvantage of a separate communication channel is that it might fail due to firewall issues. Inbound and/or outbound connectivity might be blocked, a port range (like in `GLOBUS_TCP_PORT_RANGE`) might not be configured, the clients may be locked in behind a private network, possibly using network address translation (NAT) for outbound connectivity. Fortunately, there is one particular connection that can be relied upon: the Web Service connection itself. Thus, we investigate how interactivity can be provided utilising Web Services.

3.3 Preliminary Experiments

In GT4, Web Services are implemented using the Java technology. Data is transported using HTTP or HTTP-S. To examine the feasibility of providing an

interactive data stream in this environment, an experimental client/servlet was written for the Tomcat Java Servlet container [28]. With respect to the Request/Reply mechanism, HTTP can be seen as a unidirectional protocol: A server can never initiate a request. The interactive data stream therefore uses two such HTTP connections. In the client, this has been implemented by two threads handling the upstream and the downstream separately, while on the server side, two servlets are started independently, as shown in Fig. 18.1.

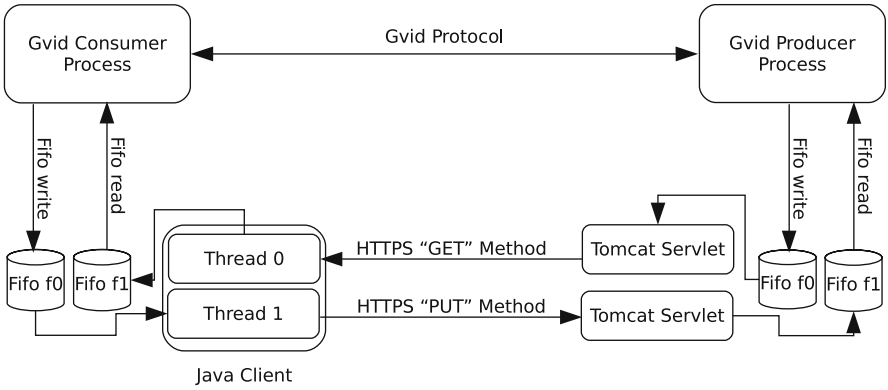


Fig. 18.1 System architecture of "GVid over Tomcat" experiment

On both sides, the communication endpoints have been connected to a named pipe, allowing arbitrary programs to attach to them. The experiment then was carried out by connecting a video stream generator such as GVid [14, 16] and a matching video stream consumer to it. The result was very promising: the Java implementation was about as fast as the native C implementation.

3.4 An Interactive Service using Globus WSRF

When trying to realise a solution with the same technique for the Globus "Web Service Resource Framework" (WSRF), the result was pretty disillusioning. While the experiment in Section 3.3 provides a constant flow of data, the new solution did not generate such a flow. The reason is that the Globus WSRF is based on exchanging messages using SOAP/XML – thus, additionally encoding the video stream from experiment 3.3 in SOAP would be necessary. The crucial point is that each message is transported over a separate connection. The reply a Web service sends has to be created as a whole before it can be transported to the requester. Afterwards, the connection is terminated; the state of a Web service is kept by the WSRF in a *resource* [25]. It is not possible to transport a reply in parts, re-using the same connection for yet unkown data. Transporting interactive data with the WSRF would result in a flurry of connects/disconnects, even if the Web service is repeatedly invoked within the same client. Therefore, this approach is not feasible.

This poses an evident question: If constantly reconnecting is unusable, is it possible to preserve an existing connection? In fact, this can be done by creating a copy of the socket descriptor in both the client and the Web Service. Once the SOAP message exchange is completed, the copy of the descriptor can be used for interactive data streaming. Since the copy exhibits the same properties as the original, data can pass through it unhindered by a possible firewall.

The crux of the matter is to obtain the socket descriptor in the Web service. The service routine only receives the message, but not the descriptor the message was received on. Getting the socket object by other means, for instance, by creating a subclass of *java.net.Socket*, or "asking" the Globus WSRF to pass the connection to the Web service is impossible. Modifying the WSRF or the AXIS engine [2] is out of the question – software built on top of another software should not require changes to the lower layers.

Since the Java interface does not pass the required information, it is necessary to resort to a different method. Using a machine-oriented language such as C, it is possible to examine *all* file descriptors of a program. This method is implemented in a supporting routine (named *ws-supp* for further reference). The *ws-supp* is using the Java Native Interface (JNI) to communicate with the Java part of the Web Service.

The method used first to identify the file descriptor belonging to the Web Service (WS) invocation was to pass the TCP address tuple in the SOAP message itself. This does, however, not work for two reasons: First, messages passed to a WS at the client side need to be built completely before the connection is established, because WS creates the connections implicitly when sending a message. Theoretically, this could be solved using "message-rewriting on the fly", which again does not work if the message is signed. Second, this method would not work at all in the presence of network address translation (NAT), where the client is part of a private subnet.

Instead, a cookie-based connection matching mechanism is used. Here, a "cookie" is a 128 bit random string. The client generates the cookie and stores it in the SOAP message. When sending the request, a duplicate of the corresponding file descriptor is created. The service accepts the message and creates the *ws-supp* routine as a separate process, then completes the WS-invocation by returning an empty reply. The client receives this reply and sends the cookie once again using the duplicate. The *ws-supp* routine, which has passively been monitoring all socket descriptors in question, can now exactly identify the matching socket descriptor.

Cookie-based connection matching also solves the problem of not being able to relate a connection to a Web Service invocation in the presence of multiple connections. It has been observed that when performing a stress-test and flooding the Globus service container with requests, a particular Web Service instance "sees" all connections, even those not belonging to its own message. Of course, it is impossible to access these other connections in Java, but from the system's point of view, gaining access to the data can easily be done. For instance, to silently intercept data, it is sufficient to use the *recv()* system call. This could eventually be considered a security problem; however, we have not investigated the source of this behaviour (probably because Web Services are implemented as Threads).

3.5 Secure Connection Re-Establishment

To protect the connection from a "man in the middle attack" [17] possibly exchanging connection cookies and thus hijacking an interactive session, a second cookie, the "process cookie" is used. The client creates a process cookie, stores it in the SOAP message and sends the message to the Web Service. When a secure transport (such as HTTP-S) is used, we can be sure that no "man in the middle" can decipher the message, thus only the client and the Web Service know the content of the cookie. After identifying the connection matching the connection cookie, the connection is only used in a secure fashion. A new security context based on the X.509 certificates present is created. Using this secure connection, both peers exchange their process cookie, making sure that *only* they know the content of the process cookie. Only when both sides recognise matching process cookies, communication is allowed to continue. Figure 18.2 shows the complete mechanism.

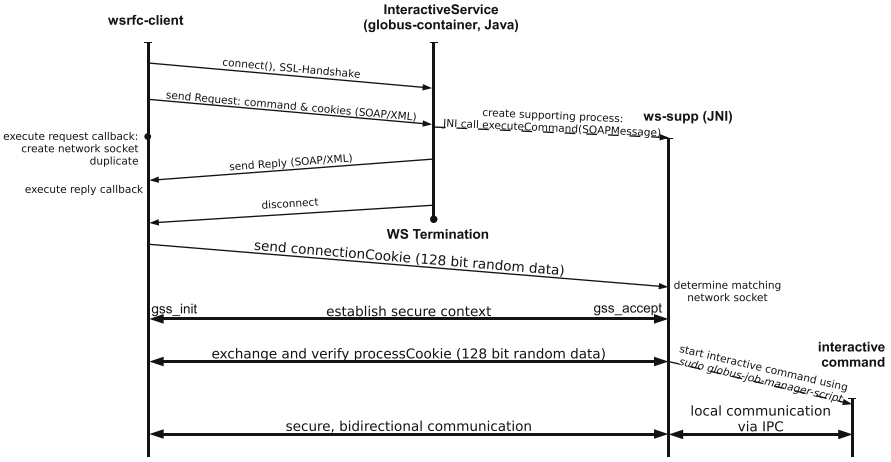


Fig. 18.2 Web service based invocation of an interactive job

Using process cookies protects against attacks from a third party not being part of the virtual organisation (VO). However, it cannot defend from "the enemy within", a malicious user being part of the same VO. In large VOs, where users even might not personally know each other, this could become a problem.

If an attacker tricks the *ws-supp* routine and the client to establish a security context with an attacking process instead of contacting each other directly, encrypted messages can be decrypted by the attacking process, recorded or even modified, decrypted again using the security context of the attacker and then sent to the intended process. To protect against this kind of attack, a further check is necessary. Since the InteractiveService expects that a user connects to its own process only, the "distinguished name" (DN) of the communication peer in the security context can be used for this purpose, because forging the peer DN would imply that an attacker has successfully corrupted the corresponding grid certification authority

(CA). Checking of the DN can be done easily in the *ws-supp* routine, because the Web Service Resource Framework (WSRF) supplies the DN of the communication peer used when invoking the Web Service. This DN can later be used to compare against the peer DN received when re-establishing the security context over the duplicated network connection.

Unfortunately, this only works in the *ws-supp* routine, which makes use of the information provided by the Java WSRF. The client, on the other hand, is implemented in C and based on GT4.0, which does not provide this information. Only with GT4.2, the identity of the peer can be queried. To overcome this problem without requiring an update to GT4.2, additional measures are taken. Instead of creating a process cookie, a 128 bit key and a 128 bit "initial vector" (IV) for symmetric encryption are created and passed safely in the SOAP request. These values are then used to protect the establishment of the security context, employing the AES encryption standard [6], providing "GSS over AES" data format. This technique is safe, since an attacker cannot know the values of the key and the IV. As soon as the security context has been established this way, the "GSS over AES" encryption is not needed any more and the system falls back to plain GSS data transfer.

3.6 Executing a Command with the InteractiveService

Prior to implementing the *InteractiveService*, it is necessary to explore how Globus executes commands using the *globusrun-ws* tool. This mechanism is described in the following paragraph.

Examining the source code and configuration files, it can be seen that the *sudo* Unix command is called from the *JobManagerScript*. Using *sudo* has become necessary in GT4 because the user-id of the container process is not executing with superuser privileges anymore. Being a "setuid" command, *sudo* has to be used to switch to the user invoking the service. The service user has previously been identified using the gridmap-file, and using an authorisation value of "gridmap" in the Web Service configuration file `security-config.xml`. In the *sudo* configuration file, `/etc/sudoers`, the container user is allowed to execute a particular command with specific parameters only: the `globus-gridmap-and-execute` command using the `grid-mapfile`, executing a command located in the $GLOBUS_ HOME/libexec/ directory. The *JobManagerScript* uses this mechanism to invoke the *globus-job-manager-script.pl* Perl script in order to submit a particular command. The command to execute is passed to the *globus-job-manager-script* by a file. This file is really just a declaration of a particular Perl hash-array variable. Since the *globus-job-manager-script* is written in Perl, the job description can easily be read in this way and the job manager can access the variable without parsing a XML or, even worse, proprietary, self-invented job description format. Once the job manager has submitted the requested command, execution is about to begin.

Instead of reinventing the wheel and therefore avoiding changes in the system, the implementation of the *InteractiveService* exactly abides by the Globus

implementation. Additionally, it keeps the connection to the Web Service client in order to make it available to the interactive command.

Beside the connection and process cookies, the SOAP message also contains the command and its parameters. To interactively execute the command, the *ws-supp* routine creates a separate process. This is required since the interactive command does not necessarily know about grid security needs. It reads and writes data in plain-text, while in contrast the *ws-supp* routine protects its communication with GSS. Data flowing forth and back between *ws-supp* and interactive command is thus encrypted/decrypted as required. If, on the other hand, plain-text communication is considered secure enough, the *ws-supp* routine could use the "File Descriptor Passing" technique described in [26] and terminate afterwards. However, this would only work if the command to execute is started on the same machine. "File Descriptor Passing" obviously does not work in a networked environment, such as the Portable Batch System (PBS).

When submitting a command, the execution mechanism present in the Globus Toolkit closes all descriptors between the *ws-supp* routine and the interactive command. This is probably done because of security reasons, but on the other hand makes it difficult to hand over the socket descriptor to the command. First, *sudo* closes all file descriptors beside 0,1,2, then the job manager takes over and closes the remaining file descriptors. After that, it opens the values specified in the job description file for *stdin, stdout* and *stderr*. This circumstance can be used by the *ws-supp* routine to pass objects suitable for opening by the job manager, while on the other hand, accepting communication for these objects itself. Obviously, a *named pipe* is required for each of the standard IO descriptors.

3.7 Secure Local Communication

When the container process creates the named pipes, their ownership is set accordingly. This implicates that the permission bits of the pipes are required to be "world"-accessible, since the grid-mapped user shares no file access privilege with the container user. This offers a "window of vulnerability" which allows a local user to arrange a "man in the middle attack." This can be done by an attacking process by connecting to the named pipes, deleting them from the file system and recreating them. Although the original named pipes have been deleted, the connection between the attacking process and the container process is still alive. A process that connects to the named pipes now will then contact the attacking process instead, which can then record or modify any data intercepted. A solution to this problem could be to encrypt the data between the *ws-supp* routine and the interactive command, but this imposes other problems.

Is there another solution which does not require additional programming, a solution which grants access to the named pipes only for two users at the system (kernel) level? In fact, using POSIX ACLs (access control lists) [12] can provide the required functionality. The container user creates the named pipes, but additionally grants read-access on stdin, and write-access on stdout and stderr to the service user.

POSIX ACLs are available in all major Unix flavour: Solaris and FreeBSD as well as Linux have them. Considering that most of today's grid computing takes place on Linux systems: POSIX ACLs are ready to use on all major Linux distributions, even in Scientific Linux 3, which uses the old 2.4 kernel. It is possible to activate ACLs even without rebooting.

The only downside is that the *globus-job-manager-script* is not ACL aware. Even though calling *open(2)* on the object would succeed, the job manager reports that it has no read permissions. A system call trace reveals that the *stat(2)* system call is used to check the permission bits only, without honouring the additional access granted by the ACLs. The correct way is to use the *access(2)* system call instead. This behaviour is documented in the *perlfunc* manual page, saying that one single line is sufficient to change to using the *access* method. Thus, adding a `use filetest 'access';` line to the $GLOBUS_LOCATION/lib/perl/Globus/GRAM/ JobManager/fork.pm is the only change required.

One might say that using ACLs is quite some effort just to protect named pipes. On the other hand, even the Globus Toolkit would benefit from using ACLs. The reason is that in GT4, the container user is usually an unprivileged account, executing commands on behalf of the grid-mapped users with the help of the *sudo* command as mentioned above. Both users require access to the same objects, therefore these objects need "world"-accessibility. This also applies for the job description file generated by the middleware, in particular, by one of the *Managed Job* services. Any local user can write a script which does a "race copy" of the contents of Globus' temporary directory – this is where the job descriptions are stored. This allows unauthorised users to inspect commands, arguments, and environment settings and file locations private to grid jobs. Using ACLs, GT4 could prevent this.

4 Future and Related Work

The CrossGrid Project [27] aimed at grid-based interactive visualisation. Interactivity was introduced by allowing a human in the processing loop, but due to the way the system was designed, the response was very slow. As an unplanned by-product, the *glogin* tool emerged. In the Interactive European Grid Project (*int.eu.grid*) [13], the *glogin* tool was modified such that the middleware extensions can invoke the required functions separately [23]. *int.eu.grid* is using *gLite* [11] middleware, which is still using the pre-Web Service approach. This could change in the future. In this case, *int.eu.grid* could be adapted to make use of the *InteractiveService* approach.

With respect to interactivity, several utilities have been built. One example is the ubiquitous "secure shell" (SSH) [30]. In the grid environment, the SSH has been expanded to support grid-related authentication, resulting in a GSI-enabled SSH (GSI-SSH). Within the Globus Toolkit 4 (GT4), the GSI-SSH is now available by default. The disadvantage of this approach is that a GSI aware ssh server (ssh daemon, sshd) needs to be installed, configured and activated on the target host. In contrast to this, *InteractiveService* is a light-weight approach, which is installed in the Globus container. Executing jobs by means of GSI-SSH results in access to

the host without being subject to scheduling mechanisms. Hence, GSI-SSH cannot replace the usual job submission mechanism.

VISIT/GS [19] is an architecture for online visualisation and steering in the Unicore grid. To get interactive access to the job, VISIT/GS uses SSH. Alternatively, it should be examined how the Web Service approach as described in this work can be realised for the Unicore 6 middleware.

The "Interactive Grid Architecture for Application Service Providers" (I-GASP) [3] provides interactivity for the grid by utilising "Virtual Network Computing" (VNC) [18] to display graphical output. In contrast to this, the way to provide interactivity as described in this work does not require a graphical display – however, it can support graphical programs by providing an interactive transport.

Grid portals, such as e.g. the "Grid ENabled Interactive Environment" (GENIE) [10], offer an "interactive environment." Here, the attribute "interactive" entirely is related to the environment, but not the job. Such portals can be considered an interactive shell around a noninteractive job execution environment. The methods described in this chapter provide interactivity not for a grid environment but for the grid jobs.

This chapter focuses on interactivity in a running application. A related, but nevertheless important property of interactivity is the requirement to start applications almost immediately. In a grid, waiting several minutes for a job to start is not uncommon, even though when resources are idle. One reason for this behaviour is that components such as schedulers and grid information systems still are not aware of interactivity and use a polling-based approach to distribute their information, instead of transporting events as they arrive (event-based approach). If many such components work together, passing data to each other in a chain, each polling delay will add to the total delay. Let us assume that in a particular grid component, the polling interval delays the processing of an event for t_w seconds, while transmitting the event to the next component lasts for t_x seconds. If events arrive at arbitrary time (in between the polling interval), the average delay for an event sent to component i is $t_{\text{mean}}(i) = \frac{1}{2}t_w(i) + t_x(i)$ until it reaches component $i + 1$, if a uniform distribution can be assumed. Actually, the real distribution is an exponential distribution, but if most (for instance, 99%) of the events arrive within the interval limits, the distribution can be approximated by a uniform distribution.

If $n + 1$ grid software components work together in an "event delivery chain", component 1 being the event source and component $n + 1$ being the event sink, the average delay is $t_{\text{mean}} = \sum \frac{1}{2}t_w(i) + t_x(i)$.

We can take for granted that $t_w(i) \gg t_x(i)$, thus, using the event based transmission approach, the delay can be reduced to a term near $\sum t_x(i)$. However, this is not the focus of this chapter and it is left as future work.

5 Conclusions

In this chapter, it has been shown and proven by implementation that interactivity with the new kind of Web Service based middleware is possible. Two methods for

the Globus Toolkit 4 have been demonstrated: First, making the pre-WS GRAM mechanism present in the *glogin* tool use WS-GRAM instead and second, providing a dedicated Web Service, the *InteractiveService*. While both methods provide a fast, bidirectional data channel, the second method has some advantages over the first one. Depending on the environment, establishing a separate data channel can last as long as the polling interval. In contrast, the *InteractiveService* usually takes only about 900 ms to start. Furthermore, one can never be sure whether a separate data channel can be established at all, since the creation of such a connection might be blocked, for instance by a firewall. In contrast, the *InteractiveService* creates a duplicate of the existing Web Service connection, activating the duplicate once the Web Service invocation is terminated. Being blocked by a firewall in this case is much less likely.

It is obvious that the new method to provide a fast interactive channel can also be used beneficially to access remote scientific instruments.

With respect to the current proof of concept, no effort has been put in optimising the secure data transfer, once the connection has been established. Data is read from the connection unbuffered, and when communicating with the interactive command executing in the grid, the *ws-supp* routine is using a buffer size of 4,096 bytes only. From this it follows that the performance of data-intensive applications might suffer. Performance optimisation was, however, not the scope of this work.

With respect to data transfer performance, much effort has been put in *glogin*'s code to achieve a high transfer speed. One might be tempted to implement these functions in the *InteractiveService*, too, but there is a better way. Instead of recognising two competing approaches in *glogin* and the *InteractiveService*, the latter can be used as an additional, alternative submission mechanism and thus get "the best of two worlds." This is left as future work.

Acknowledgments We would like to thank our colleagues Paul Heinzlreiter for proofreading and Peter Praxmarer for discussing the technical details of the implementation.

The work described in this chapter was supported in part by the European Union through the FP6-2005-Infrastructures-7- contract No. 031857 project "Int.eu.grid".

References

1. G. Alonso. *Web services: concepts, architectures and applications*. Springer, 2004.
2. Apache Axis (Apache eXtensible Interaction System). `http://ws.apache.org/axis/`, `http://ws.apache.org/axis2/`
3. S. Basu, V. Talwar, B. Agarwalla, and R. Kumar. Interactive grid architecture for application service providers. Technical Report, Mobile and Media Systems Laboratory, HP Laboratories Palo Alto, July 2003.
4. S. Brand. *The Media Lab: Inventing the Future at M.I.T.*, p. 46. Penguin, 1988.
5. S. Cronholm. Need for action oriented design and evaluation of information systems. *Human – Computer Interaction, Theory and Practice (Part I)*, pp. 306–310, 2003.
6. J. Daemen and V. Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer-Verlag, 2002.
7. D. Erwin and D. Snelling. UNICORE: A grid computing environment. *Lecture Notes in Computer Science*, 2001.

8. I. Foster. Globus Toolkit Version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, vol. LNCS 3779, pp. 2–13. Springer-Verlag, 2006.
9. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufman, 1999.
10. GENIE: Grid ENabled Interactive Environment. `http://www.npaci.edu/online/v5.5/genie.html`
11. gLite: Lightweight Middleware for Grid Computing. `http://glite.web.cern.ch/glite`
12. IEEE 1003.1e and 1003.2c. *Draft Standard for Information Technology–Portable Operating System Interface (POSIX)–Part 1: System Application Program Interface (API) and Part 2: Shell and Utilities, draft 17 (withdrawn)*, Oct. 1997. `http://wt.xpilot.org/publications/posix.1e/`
13. Interactive European Grid Project. `http://www.interactive-grid.eu`
14. T. Köckerbauer, M. Polak, T. Stütz, and A. Uhl. GVid – video coding and encryption for advanced Grid visualization. In J. Volkert, T. Fahringer, D. Kranzlmüller, and W. Schreiner, editors, *Proceedings of the 1st Austrian Grid Symposium*, volume 210 of `books@ocg.at`, pp. 204–218, Austrian Computer Society Schloss Hagenberg, Austria, 2006.
15. J. Linn. *Generic Security Service Application Program Interface*, vol. RFC 2743. Internet Engineering Task Force, Jan. 2000.
16. M. Polak and D. Kranzlmüller. Interactive Videostreaming Visualization on Grids. *Cluster and Computational Grids for Scientific Computing*, 24(1):39–45, Jan. 2008.
17. E. Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison–Wesley, 2001.
18. T. Richardson, Q. Stafford-Fraser, K. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, Jan/Feb 1998.
19. M. Riedel, W. Frings, S. Dominiczak, T. Eickermann, D. Mallmann, P. Gibbon, and T. Dussel. VISIT/GS: Higher Level Grid Services for Scientific Collaborative Online Visualization and Steering in UNICORE Grids. In *Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07)*, 2007.
20. H. Rosmanith, P. Praxmarer, D. Kranzlmüller, and J. Volkert. Towards Job Accounting in Existing Resource. In *Schedulers: Weaknesses and Improvements High Performance Computing and Communication (HPCC 2006)*, pp. 719–726, Munich, Germany, 2006.
21. H. Rosmanith and J. Volkert. glogin – Interactive connectivity for the grid. In *Distributed and Parallel Systems: Cluster and Grid Computing (DAPSYS 2004), Austrian-Hungarian Workshop on Distributed and Parallel Systems*, pp. 3–12, 2004.
22. H. Rosmanith and J. Volkert. Traffic forwarding with GSH/GLOGIN. In *13th Euromicro Workshop on Parallel, Distributed and Network-Based Processing (PDP 2005)*, pp. 213–219, 2005.
23. H. Rosmanith, J. Volkert, R. Valles, F. Serrano, M. Plociennik, and M. Owsiak. Interactive fusion simulation and visualisation on the grid. In *Sixth International Symposium on Parallel and Distributed Computing (ISPDC)*, Hagenberg, 2007.
24. R. Schulmeister. *Taxonomy of Interactivity in Multimedia – a contribution to the current Metadata-Discussion*. 2002. (German) it+ti, Ausgabe 4.
25. B. Sotomayor and L. Childers. *Globus Toolkit 4: Programming Java Services*. Morgan Kaufmann Publishers, 2006.
26. R.W. Stevens. *Advanced Programming in the Unix Environment*, pp. 479–490. Addison-Wesley, 1993.
27. The Crossgrid Project. `http://www.crossgrid.org`
28. Tomcat Home Page. `http://jakarta.apache.org/tomcat`
29. J. Wray. *Generic Security Service API Version 2: C-bindings*, volume RFC 2744. Internet Engineering Task Force, Jan. 2000.
30. T. Ylönen. SSH secure login connections over the Internet. In *Sixth USENIX Security Symposium*, pp. 37–42. SSH Communications Security Ltd., 1996. `http://www.usenix.org/publications/library/proceedings/sec96/ full_papers/ylonen/`

Chapter 19
Fusion Simulations, Data Visualization Results and Future Requirements for the Interactive Grid Infrastructure

F. Castejón, D. Lopez-Bruna, J.M. Reynolds, A. Tarancón, R. Valles, and J.L. Velasco

Abstract Physics simulations can be very resource demanding, and the fusion ones are a good example. Grid computing, which is continuously improving its features, is a promising architecture for the execution of these simulations. On the other hand, the visualization of generated data could be significantly extended by the use of an interactive grid environment in which users could manage and interact with the big amount of remote stored data. Some fusion problems have already been proved into this kind of framework, but some other in-development approaches can show benefit from current features and requirements for new future grid adaptations.

Keywords Grid computing · Fusion · Simulation · Interactive · Visualization · Langevin · Spectral elements

1 Introduction

Usually grid applications are essentially serialized, but most of the computations needed in Physics are parallel. It is clear that massive parallel applications are not well suited for grid architecture, but there are tasks that are neither serial nor massively parallel. The fact that the network bandwidth is continuously increasing points toward a near future were some parallel applications could be *gridified*. In particular, here we are going to discuss simulations for high temperature fusion plasma.

This chapter is structured as follows: in Section 2, after a brief introduction to fusion, there is a short explanation of the theoretical models that use the fusion

F. Castejón (✉)
Instituto de Biocomputación y Física de Sistemas Complejos. Universidad de Zaragoza 50009 Zaragoza, Spain
Laboratorio Nacional de Fusión – Asociación Euratom/Ciemat 28040-Madrid, Spain
e-mail: francisco.castejon@ciemat.es

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_19,

codes. In Section 3 the use case proposed and the suggested architecture are introduced. In Section 4 the proposed applications are introduced. Finally, the conclusions come in Section 5.

2 Fusion Introduction and Theoretical models

Fusion offers the prospect of a long-term safe and environmentally friendly energy source. It is particularly attractive because it uses abundant and delocalized fuel: water and lithium. The basic idea is to obtain energy from the union (fusion) of light atom nuclei. The mass of the components after the fusion is less than before and the exceeding mass m is transformed into kinetic energy following the famous expression $E=mc^2$, where c is the speed of light. In order to overcome the electrostatic repulsions among atoms, high energies are needed. This makes necessary to increase enormously the temperature of the fuel, leading it to a plasma state. This brings the problem of maintaining such a hot plasma without melting the container. The most promising way is to generate a "magnetic bottle" by means of a set of coils and to contain the fuel without touching it. But this is a very hard task, because many instabilities grow up and transfer the heat of the fuel to the container vessel. Active research is being carried out currently on finding the best configuration of coils, how to inject energy, and other physical and engineering parameters of the system. One way to get an approximation to the problem is simulating the behavior of the reactor. For more details about the Physics of magnetically confined plasmas, see [1].

The behavior of the plasma can be described with high degree of approximation by the guiding center equation for every species:

$$\frac{d}{dt}\left[f_a(\vec{r},\vec{v})dV\right] = \sum_b C_{ab}(f_a(\vec{r},\vec{v}),f_b(\vec{r},\vec{v}))dV \tag{19.1}$$

Essentially, the distribution function (the probability of finding a particle in a determinate position and velocity) inside the plasma evolves following magnetic field lines, with small perpendicular drifts. The collision term, in the rhs of the equation, takes into account the local electrostatic interaction among particles.

Equation (19.1) can be expressed as

$$\frac{\partial \hat{f}_a}{\partial t} + \vec{\nabla}_{\vec{r}_{GC}} \cdot (\vec{u}\hat{f}_a) + \frac{\partial}{\partial \lambda}(a_\lambda \hat{f}_a) + \frac{\partial}{\partial x^2}(a_{x^2}\hat{f}_a) = \sum_b \mathcal{L}_b \hat{f}_a \tag{19.2}$$

Here $\vec{r}_{GC}$ denotes the position of the ions and λ and x^2 their velocity. This equation describes the evolution under several non-restrictive assumptions (guiding-centre approximation, long-range Coulomb collisions; see [2] for more details). The magnetic and electric fields are included in the equation via the functions a_λ, a_{x^2}. Its detailed functional form is written in [2].

The exact solution of this equation gives us the evolution of the plasma but, for complex geometries, cannot be analytically solved, so a numerical method is needed. This is not an easy task, because the distribution function evolves in a five-dimensional space: three spatial and, thanks to the guiding center approximation, only two velocity coordinates. Two methods are exposed here.

2.1 Langevin Approach

The equation to solve is non-linear, and of high complexity, since $\mathcal{L}_b$ depends on f. The *Langevin approach* is to linearize the equation, ignoring this latter dependence. The evolution of a subset of the plasma (*test ions*) inside a plasma considered as a stationary thermal bath is then studied [12, 3].

Now, Eq. (19.2) is equivalent to solve a closed set of five-coupled stochastic differential equations (SDE):

$$\frac{\mathrm{d}\vec{r}_{\mathrm{GC}}}{\mathrm{d}t} = \vec{a}_{r_{\mathrm{GC}}} \tag{19.3}$$

$$\frac{\mathrm{d}\lambda}{\mathrm{d}t} = \left[a_\lambda + a_\lambda^{\mathrm{Ito}}\right] + b_\lambda \xi_\lambda \tag{19.4}$$

$$\frac{\mathrm{d}x^2}{\mathrm{d}t} = \left[a_{x^2} + a_{x^2}^{\mathrm{Ito}}\right] + b_{x^2} \xi_{x^2} \tag{19.5}$$

Solving this SDE means to calculate a large number of trajectories of the *test ions* in the presence of magnetic and electric fields and collisions with a background plasma of a given temperature and density (Fig. 19.1).

From a computationally point of view, if the electric field does not evolve in time, and the collision term only has into account the interaction with a fixed particle background, the simulation can be divided into blocks of particles with no mutual interaction. In this case, the grid architecture is the ideal environment. If the background and the electric field evolves, the maximum data transfer among nodes fixes the maximum rate of update of these quantities.

2.2 Direct Approach

Another way is to solve directly the guiding center equation (19.1) [9, 10]. First, a 3D spatial mesh adapted to the magnetic geometry of the problem is generated (Figs. 19.2 and 19.3). The mesh is composed of hexahedral elements and, in every element, the distribution function is expressed as a truncated spectral expansion:

$$f(\vec{r},v',\lambda) = \sum_{i=0}^{N_v} C_i(\vec{r}) J_i(v',\lambda) e^{-v'^2} \tag{19.6}$$

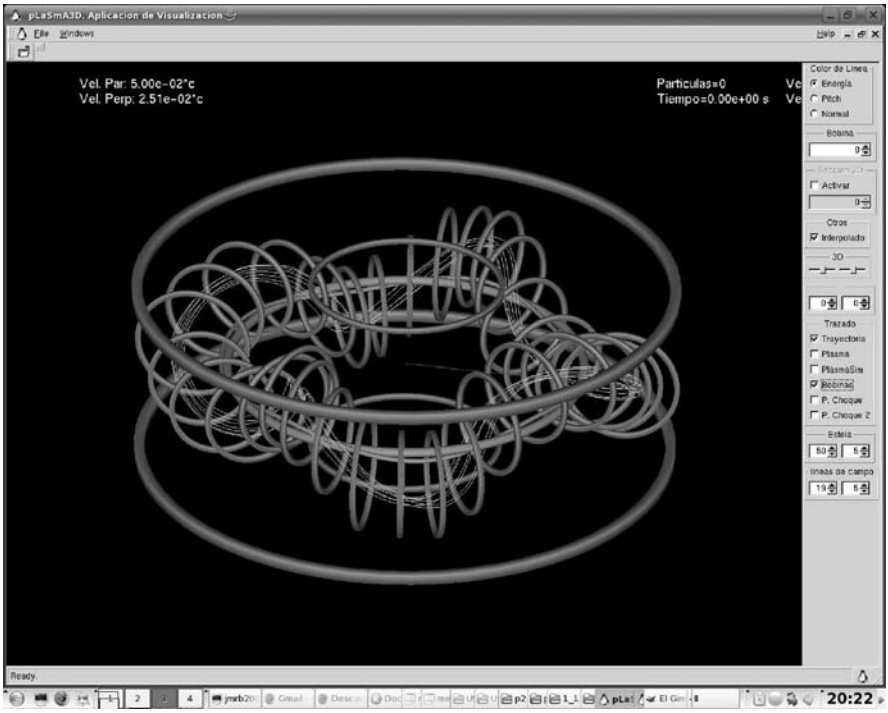


Fig. 19.1 Some of the trajectories in a Langevin simulation

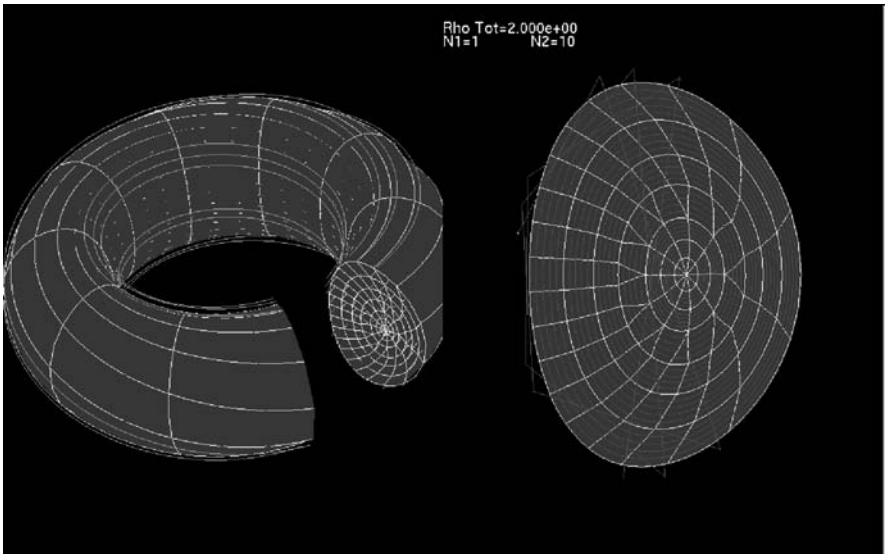


Fig. 19.2 A Tokamak-type device mesh

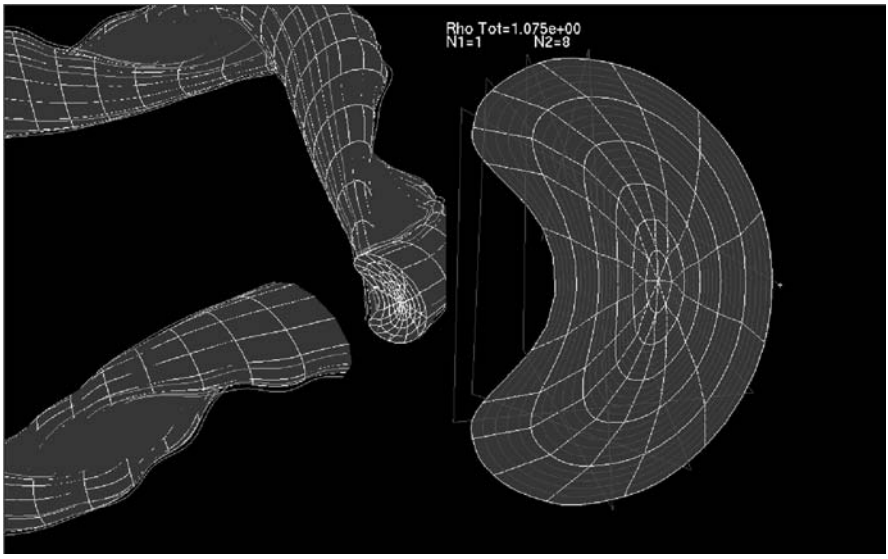


Fig. 19.3 A Stellarator device-type mesh

The solution of the equation is obtained in the spectral space inside each element, but taking into account the interaction with neighborhood ones: at the interfaces, conservation and stability are ensured through a Roe flux Riemann solver to ensure conservation and stability [7]. From a computational point of view, the task could be parallelized sharing out the mesh amongst the nodes. At every iteration of the main loop, the interaction among hexahedrals must be taken into account through communication among nodes. So, an equilibrium must be found among the number of hexahedral elements per node and the order of the spectral expansion (total work per iteration per node) in function of the existing hardware architecture.

3 Adaptation of IVISDEP to Int.eu.grid Framework

3.1 Use Case

As an example of the Langevin approach (Section 2.1), we show the already ported and integrated IVISDEP in the int.eu.grid project [4].

It is an application that the user can interact with, viewing the results of the simulation in a reasonable interactivity time, running it in the worker nodes (WN) of the grid.

In this interactive grid scenario, the main objective is to distribute the whole number of trajectories to be calculated among as much WNs as possible. This way, we can compute a greater number of trajectories at once and obtain more detailed results. This approach is possible due to the assumption of no interaction among the

different trajectories. This issue is already taken into account in the resolution of the SDE thanks to the definition of background ion distribution.

3.2 Architecture

In the int.eu.grid scenario, the application is started from Migrating Desktop (MD) [6], which is a roaming virtual Java desktop to facilitate the access to the grid. It is the entering point to the grid and at the same time it is the layer where the final visualization is achieved. Gvid [8] is the plug-in which makes the visualization possible inside the Java window and Glogin [11] is the channel through which the whole communication between the Roaming Access Server (RAS) and the grid directly takes place (Fig. 19.4).

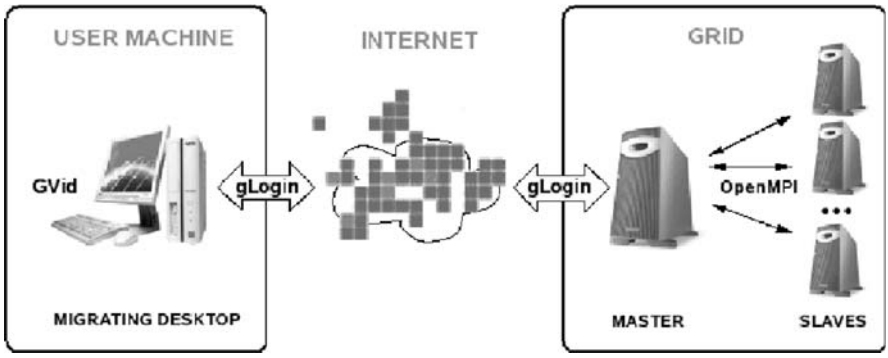


Fig. 19.4 Parallelization of the simulation

The application needs to be steered remotely from the Gvid plug-in in the MD. The graphical interface part has been modified to become a GLUT compliant application, in order to be able to receive callbacks and changes of parameters during the simulation, using the remote callbacks implemented inside a Remote Glut library.

The fusion application fits a Master–Slaves scheme, in which the Master is in charge of the visualization issues and of gathering, in a synchronized way, the data calculated by the Slaves. The latter are fully dedicated to the calculation of new positions of the particles and to sending the data to the Master to be visualized. The ISDEP computing core was restructured to distribute the computation among the WNs. This new distributed system was based on MPI [5] because of the fact that it provided the features needed to manage the data calculated in the different machines.

This whole infrastructure working together is what makes the application capable of being executed in an interactive grid environment, taking advantage of the great amount of computing power of the resources.

As technology is evolving continuously, different kinds of fusion simulations could be successfully adapted to this sort of infrastructures. In the next section we suggest one possible application in our research field.

4 Proposed Architecture

4.1 Direct Approach

Here, we are going to describe with more detail the numeric method outlined in Section 2.2 to show how it could be possible to *gridify* this massive parallel application.

Suppose that the electric and magnetic fields are fixed in the simulation so the communication among the nodes comes only through the interaction of hexahedrals (Fig. 19.5). The order of the spatial polynomials is N_p and the velocity spectral expansion has N_v modes. The number of data elements to be shared between two hexahedrals in every iteration is $\left(N_p^{1/3}+1\right)^2 N_v$ in both directions and the cost of the computation in the interior of the hexahedral grows up as $O\left(N_p^2 N_v^2\right)$ [9]. For a given spatial resolution, it is equivalent to have many hexahedrals with low-order polynomials or to have less hexahedrals with higher order polynomials, so N_p can be adjusted to increase the efficiency of the problem implementation.

Quantitatively, for $N_v = 100$ and $N_p = 27$, the iteration of 100 hexahedrals takes a computation time of the order of 1 s/iteration in a typical P4 machine. The communication among two hexahedrals is 16 spatial nodes/hexahedral surface * 100 velocity modes * 8 bytes/data = 12,800 bytes/hexahedral surface. If every hexahedral has communication with other elements in different nodes by two surfaces, the total data transferred in one iteration is 12,800 B/hexahedral surface * 100 hexahedral * 2 hexahedral surfaces/hexahedral = 2.4 MB so the bandwidth is 2.4 MB/iteration *

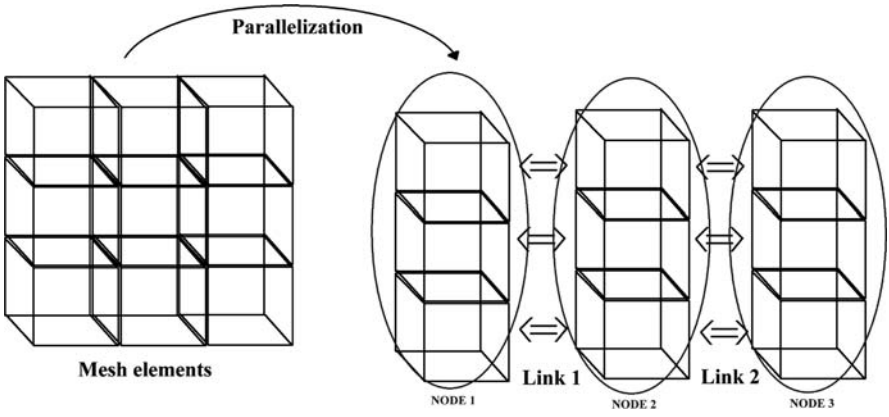


Fig. 19.5 Parallelization of the simulation

1 iteration/s = 2.4 MB/s. So the bandwdith is fixed, independently of the total number of hexahedrals that can increase with the machine complexity or the precision required but shared to the nodes in clusters of 100 elements. So, in this example, the only requirement is to have a constant full-duplex 2.4 MB/s link among nodes connected in the mesh topology.

4.2 Data Management and Visualization in the Grid

After the simulation runtime, it is necessary to analyze the data generated. Sometimes, it is not an easy task to know "a priori" what data is going to be interesting for the scientific study; so, the more data can be visualized the more information can be extracted about what would be interesting to analyze. The problem is the big amount of data generated and shared among nodes of the grid or the hardware platform.

A possible future research could be to develop an infrastructure capable of getting the data from the storage elements, analyze them on WN, and interact with a visualization architecture that would be a node with high performance graphic hardware or a "distributed" platform for the graphic generation (Fig. 19.6). So the user would have a visualization hardware, for example, a projection stereo system. In the grid part the data for the visualization system comes from the graphic generation platform that interacts with the data managing/analyzing platform. This last platform takes the data from the storing elements. With this architecture, the user could access and visualize remotely all the data generated by the simulations.

Quantitatively, suppose that we have the data generated by the simulation of the example of the previous section and we want the full distribution function in

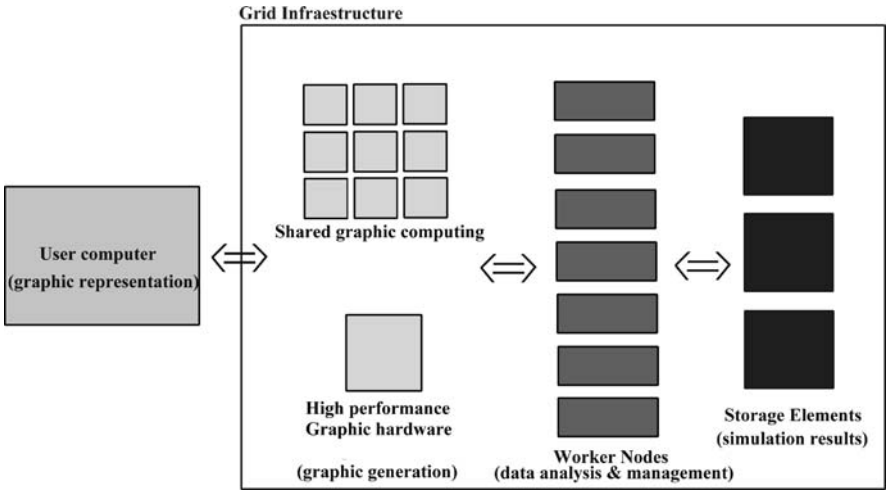


Fig. 19.6 System architecture overview

a set of selected times, suppose 1,000 instants. It is necessary to store for every time and hexahedral: 27 spatial modes * 100 velocity modes * 8 bytes/data=21,600 bytes. Suppose that the mesh is composed by 5,000 hexahedrals, so the total data stored in the simulation is 21,600 B/hexahedral instant * 5000 hexahedrals * 1000 instants=101 GB. If many simulations are going to be usually run and analyzed, it is easy to understand the real "usability" of the proposed platform.

5 Conclusions

In this chapter the possibility to use the grid infrastructure for parallel demanding tasks has been proposed. An introduction to plasma physics codes has been sketched and their requirements, from the point of view of the bandwidth, have been estimated.

The conclusion is that the grid could support the required bandwidth and introduce a feasible scenario for the implementation of some codes that are not massively parallel but are considered parallel demanding. Spectral element codes and real-time data visualization and management are the ideal candidates.

References

1. A.H. Boozer. Physics of magnetically confined plasmas. *Reviews of Modern Physics*, 76(4):1071–1141, 2005.
2. F. Castejón, L.A. Fernandez, J. Guasp, V. Martin-Mayor, A. Tarancón, and J.L. Velasco. Ion kinetic transport in the presence of collisions and electric field in TJ-II ECRH plasmas. *Plasma Physics and Controlled Fusion*, 49(6):753–776, 2007.
3. F. Castejón, A. López-Fraguas, A. Tarancón, and J.L. Velasco. The particle flux structure and the search for a flux-expansion divertor in TJ-II. *Plasma Fusion Research*, 3:S1009, 2008.
4. F. Castejón, J.M. Reynolds, F. Serrano, A. Tarancón, R. Valles, and J.L. Velasco. Fusion plasma simulation in the interactive grid. *Journal of Computing and Informatics*, 27:261–270, 2008.
5. E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and T.S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. *Lecture Notes in Computer Science*, 3241:97–104, 2004.
6. M. Kupczyk, R. Lichwała, N. Meyer, B. Palak, M. Płóciennik, and P. Wolniewicz. Applications on demand as the exploitation of the migrating desktop. *Future Generation Computer Systems*, 21:37–44, 2005.
7. Y. Liu, M. Vinokur, and Z.J. Wang. Spectral difference method for unstructured grids I: Basic formulation. *Journal of Computational Physics*, 216:780–801, 2006.
8. M. Polak and D. Kranzlmüller. Interactive videostreaming visualization on grids. *Cluster and Computational Grids for Scientific Computing, Future Generation Computer Systems, Special Section, Elsevier Science*, 24(1):39–45, 2008.
9. J.M. Reynolds, D. Lopez-Bruna, J. Guasp, J.L. Velasco, and A. Tarancón. A new code for collisional drift kinetic equation solving. In *Bifi 2008 III International Conference*, 2008.
10. J.M. Reynolds, D. Lopez-Bruna, J. Guasp, J.L. Velasco, and A. Tarancón. Simulation drift kinetic electron-ion equation with collisions in complex geometry. In *35th EPS Plasma Physics Conference*, 2008.

11. H. Rosmanith and D. Kranzlmüller. Glogin - A Multifunctional Interactive Tunnel into the Grid. In R. Buyya, editor, *Proceedings of the Grid 2004, 5th IEEE/ACM International Workshop on Grid Computing*, pp. 266–272. IEEE Computer Society, 2004. ISBN 0-7695-2256-4.
12. J.L. Velasco, F. Castejón, L.A. Fernandez, V. Martin-Mayor, A. Tarancón, and T. Estrada. Ion heating in transitions to CERC in the Stellarator TJ-II. *Plasma Physics and Controlled Fusion*, 48:065008, 2008.

Chapter 20
Interactive Grid-Access Using MATLAB

M. Hardt, M. Zapf, and N.V. Ruiter

Abstract At Forschungszentrum Karlsruhe an ultrasound computer tomograph for breast cancer imaging in 3D is under development. The aim of this project is the support of early diagnosis of breast cancer. The process of reconstructing the 20 GB of measurement data into a visual 3D volume is very compute intensive. The reconstruction algorithms are written in Matlab. As a calculation platform we are using grid technologies based on gLite, which provides a powerful computing infrastructure.

This chapter will show the way how we access the grid from within Matlab without the requirement of additional licenses for the grid. It will describe the tools and methods that were integrated. We will conclude with a performance evaluation of the presented system.

Keywords Interactive · Grid · Matlab · Distributed computing

1 Introduction

At Forschungszentrum Karlsruhe a 3D ultrasound computer tomograph (USCT, Fig. 20.1) for early breast cancer diagnosis is currently developed [2, 7] . In the Western world every 10th woman is affected by breast cancer, which has a mortality rate approaching 30%. The long-term goal of the project is to develop a medical device for use in a hospital or at doctor's surgery, applicable for screening and location of breast cancer already at an early stage. Therefore USCT must be useable as simple and quick as today's X-ray devices.

The current USCT demonstrator consists of 384 ultrasound emitters and 1,536 receivers. For each scan of the volume, a single emitter – one at a time – emits a spherical wave that is recorded by all 1,536 receivers. This delivers approximately 600,000 amplitude-scans or A-scans. For increased accuracy, the same volume can

M. Hardt (✉)
Steinbuch Centre for Computing (SCC), Forschungszentrum Karlsruhe, Karlsruhe, Germany
e-mail: marcus.hardt@icur.fzk.de

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_20,

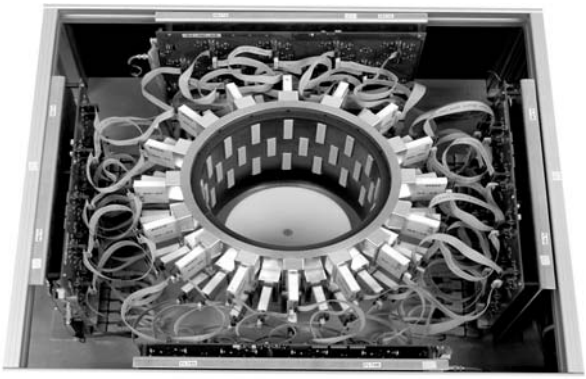


Fig. 20.1 Top view of the current demonstrator for 3D USCT

be scanned with the emitter–receiver combination rotated in six different positions, resulting in a total of approximately 3.5 million A-scans or 20 GB of data.

The availability of suitable algorithms that translate the measured data into a 3D volume is the crucial part of the whole system. The USCT group at FZK therefore develops not only the hardware but also these reconstruction algorithms. The strategic development platform for this is Matlab.

Matlab^{TM}[4] is a problem-solving environment, widely used in the fields of science and engineering for prototyping and development of algorithms. Matlab is usually bound to running on only one CPU in one computer, limiting the size of the problem that can be solved.

For best performance of USCT, the core of the reconstruction algorithm was rewritten in assembler, utilising the *Single Instruction Multiple Data* (SIMD) processor extension and a minimal amount of memory transfers [9]. Despite these optimisations, the computation time grows to very large values (see Table 20.1), depending on the number of A-scans (N_{A_scans}) and the desired output voxel resolution (N_{Pixel}). The computational complexity is of the order $O(N_{A\_scans} \cdot N_{Pixel})$

Table 20.1 The required computation time depends linearly on the number of A-Scans and the number of pixels of the requested image. Time values are given for computation on a Pentium IV 3.2 GHz (single core) with 2 GB RAM

Image type	N_{A_scans}	Input	N_{Pixel}	Output	Time
Slice image	6 144	35 MB	$4\,096^2$	128 MB	39 min
Quick vol.	$3.5 \cdot 10^6$	20 GB	$128^2 \cdot 100$	12.5 MB	36.6 h
Medium vol.	$3.5 \cdot 10^6$	20 GB	$1024^2 \cdot 100$	1.25 GB	10 d
Maximal vol.	$3.5 \cdot 10^6$	20 GB	$4\,096^2 \cdot 3\,410$	426 GB	145.8 a

For coping with the required amount of computational resources, a uniform interface to compute resources (data and CPU) is being created. The target platform is gLite as it provides access to the largest infrastructure for scientific applications [1].

1.1 The Grid Middleware gLite

We use a gLite installation with the interactive extensions as well as the Message Passing Interface (MPI) that are provided by the Interactive European Grid [6]. This infrastructure is described in detail in [3]. The gLite [5] middleware provides an interface to allocate heterogeneous compute resources on a "per job" basis. Within this so-called job paradigm, one job consists of a description of job requirements:

- dependent input data that has to be already on the grid
- operating system requirements
- memory or CPU requirements
- hardware architecture requirements

and a self-contained piece of software that will be executed at a remote batch queue. For job submission gLite provides a specialised computer: the User-Interface (UI). This computer contains client software for submitting jobs as well as the application that is going to be executed in the grid. For submitting a job, the UI forwards it to the ResourceBroker (RB) ((1) in Fig. 20.2). The RB has access to several sources of monitoring to find the resource with the best match to the user's job-description. This resource is typically the headnode of a cluster or Compute Element (CE) in gLite terminology. The CE forwards (3) the job to one of the cluster or WorkerNodes (WN) where the job will be executed. If output was created on the WN either it can be stored and registered in the grid for later use or it has to be transported back along the path (3), (4), (5).

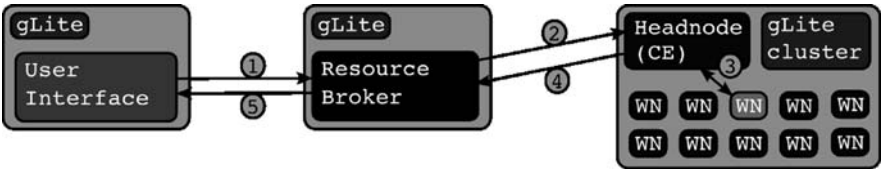


Fig. 20.2 The diagram shows the path of a grid job using the gLite infrastructure. Before execution can start on the WorkerNode (WN), data and application have to be transferred from the User Interface (UI) via the ResourceBroker (RB) (1) to the CE (2) and further to the WN (3). Output has to be transported back along the same path ((3) (4) and (5))

1.2 Limitations of gLite

The infrastructure described above is known to scale from large to very large numbers of resources (see Fig. 20.3). Being designed to meet these scaling requirements, the usability was not the primary focus. This becomes apparent when trying to use this grid infrastructure with applications for which gLite was not intentionally designed. Examples for such applications could be a scientific problem-solving environment (PSE), e.g. Matlab or a spreadsheet calculation program (e.g. Microsoft

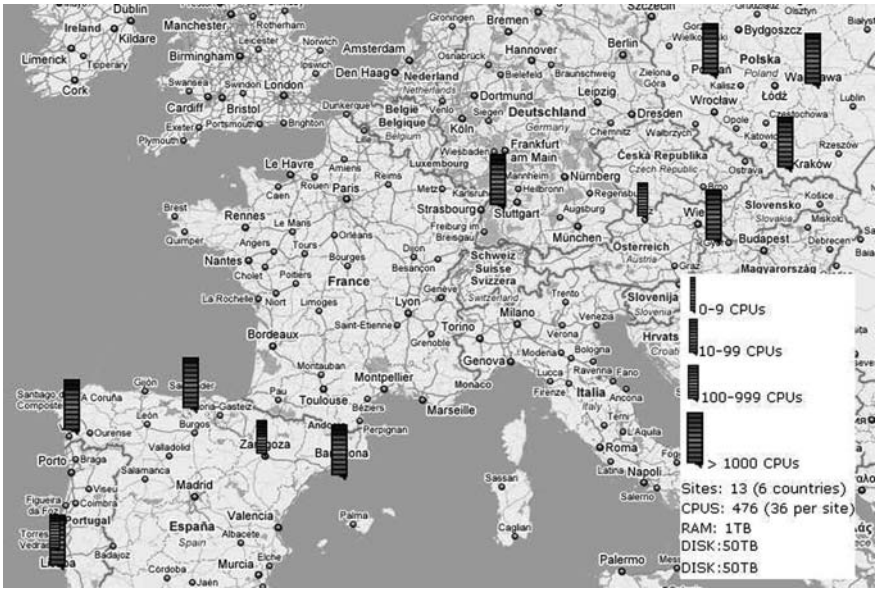


Fig. 20.3 European part of the LCG grid. The figures below date from October 2007 and comprise the worldwide installation

Excel). The differences between this style of applications and the ones that gLite was designed for are manyfold.

A typical software consists of two parts: A graphical user interface (GUI) and a backend that processes computations based on the user's commands. The application can utilize remote resources in the grid. While the GUI would run at the users PC, the compute intensive part of the application would make use of the resources in the grid. For our example of the Excel application this would mean that only the spreadsheet computation runs remotely, while the main part of the application remains local.

The gLite job paradigm, however, requires a different structure of applications. This is because first the startup of a job in gLite has an overhead in the range of $10\,s$. Second a job is an application, i.e. a self-contained application has to be sent to the grid, started on the assigned WorkerNode, to compute the problem, terminate and return the result. Finally, the gLite does not support direct communication between the user and his job.

For our example this means that we had to submit the whole Excel application as a job, wait until the computation of the spreadsheet is finished, transfer the result back and then take a look at the result.

Quintessentially, the lack of support for direct communication makes interactive usage of the grid very difficult and API-like calls of complex tasks virtually impossible.

An additional issue is related to the fact that we utilise computers in many different organisational domains and countries. This leads to the problem that software dependencies might not always be fulfilled. This results in the job to be aborted.

Finally, software licenses are not always compatible with distributed computing. While some applications are licensed on a per-user base, that one user can run on as many resources as he likes, other applications require one running license per entity. Many scientific problem-solving environments (e.g. Matlab) require one license per computer.

This contribution describes how these limitations are overcome. In the following sections we describe the improvements of the functionality of gLite. We will describe the proposed solution in detail in Sections 2 and 3. Furthermore, we will prove that the described solution works by presenting the results of a performance evaluation that we conducted using our solution in Section 4.

2 Improving Grid Access

Our goal is being able to use grid resources in a remote procedure call (RPC) like fashion. For this we want to install an RPC tool on gLite resources. This section describes our way how to efficiently allocate gLite resources and how we install the RPC tool.

2.1 Pilot Jobs

This concept of Pilot Jobs is based on sending a number of small jobs through the gLite middleware. Upon start on the WorkerNode (WN) each pilot job performs initial tests. Only when these tests are successful the pilot job connects back to a central registry or to a client.

Using this scheme avoids the pitfalls that might occur at various levels. Any reason for failure, no matter if it is related to the middleware, to network connectivity problems or to missing software at the destination WorkerNode will cause the pilot job to fail. Therefore only those jobs that start up successfully will connect back to the registry or client. In order to compensate for the potential failure rate, we submit around 20% more jobs than required. This will eventually lead to enough available resources.

Another advantage of pilot jobs is that an interactive access to the WorkerNodes is possible, because they connect back to a central location and thus provide direct network connectivity. Furthermore, the inefficient handling of scientific data (see Fig. 20.2) can be done via the direct network connection as well. This considerably improves the usability of gLite.

2.2 GridSolve

GridSolve [8] is currently developed at the ICL at University of Tennessee in Knoxville. It provides a comfortable interface for calling remote procedures on

the Grid (GRPC). It supports various programming languages on the client-side (C, C++, Java, Fortran, Matlab, Octave).

The interface provided by GridSolve is very simple to use. The required modifications in the source code can be as simple as:

```
y = problem(x) => y = gs_call('problem', x)
```

This modification will execute the function `problem` on a remote resource. GridSolve selects the remote resource from a pool of previously connected servers. The transfer of the input data (`x`) to the remote resources and the return of the result (`y`) are handled by GridSolve.

Asynchronous calls of `problem` are also possible. The actual parallelisation is done by taking advantage of this interface. Doing this is the task of the software developer.

GridSolve consists of four components:

- `The gs-client` is installed on the scientist's workstation. It provides the connectivity to the GridSolve system.
- `The gs-server` runs as a daemon on all those computers that act as compute resources. In the gLite context these are the workernodes. When started up, the server connects to the `gs-agent`.
- `The gs-agent.` It runs as a daemon on a machine that can be reached via the Internet from both, the scientist's workstation and from the gs-servers on the WorkerNodes.
- `The gs-proxy` daemon can be used in case WorkerNodes or the client reside behind firewalls or in private IP subnets.

Upon start of the servers, they connect to the agent and report the problems that they can solve. Whenever the client needs to solve a problem it asks the agent which service is available on which server and then directly connects to it and calls the remote service.

GridSolve provides client interfaces to various programming languages like C, C++, Java, Fortran and to problem-solving environments like R, Octave and Matlab so that users can continue using their accustomed environment. Detailed information about how GridSolve works is provided in [8].

3 Integration of GridSolve and gLite

On the one hand we are motivated to use gLite because of the large amount of resources provided and on the other hand we want to use GridSolve because it provides easy access to remote resources. The integration of both platforms promises an API-like access to resources that are made available by gLite.

One challenge originates from the fact that the developments of the grid-infrastructure middleware gLite and the gridRPC middleware GridSolve have progressed simultaneously without mutual consideration.

Therefore neither system was designed to work with the other one. In order to integrate both software packages, certain integrative tasks have to be performed.

To keep these tasks as reproducible and useful as possible, we have developed a set of tools and created a toolbox that we named *giggle* (*G*enuine *I*ntegration of *G*ridSolve and *gLite*).

3.1 Giggle design

We want to use the gLite resources within the GridSolve framework, hence we have to start GridSolve servers on a number of gLite WorkerNodes. This is done by submitting pilot jobs to start the `gs-server` daemon, which allocates resource for the gs-clients that send the actual compute task.

Giggle simplifies the pilot job mechanism for the user by providing predefined jobs that are sent whenever the user requests resources. Using the interactive grid extensions, it is possible to allocate several computers that are either scattered across the grid or confined to one cluster or a combination of both with just one job.

Every single pilot will be started on a WorkerNode (WN) by gLite. It is impossible to know which software is installed at that particular WN. This is why giggle downloads and installs a pre-built binary package of GridSolve together with the most commonly used libraries. Currently these packages are downloaded from a webserver.

For enhanced speed of startup time and network throughput, caching is supported. Furthermore, shared filesystem clusters benefit from gained installation speed, because in that case only one shared installation per cluster is carried out. After the installation, the `gs-server` is started. It connects to the `gs-agent` and is thereafter available for the user. In order to accomplish this, we have defined infrastructure servers that carry out specific tasks:

- The user has access to the *developers workstation*. This is typically his own desktop or laptop computer. Giggle and the GridSolve client (`gs-client`) have to be installed.
- A *webserver* is used to store all software components which have to be installed on the WorkerNodes by giggle. We cannot ship these components via gLite mechanisms because then all packages are transferred via two additional computers, as can be seen in Fig. 20.2.
- The *servicehost* runs the GridSolve agent (`gs-agent`) and optionally, the proxy (`gs-proxy`). A separate host is currently required because of specific firewall requirements which cannot be easily integrated with the security requirements of a webserver. Technically the servicehost can run on any publicly accessible computer, e.g. the above-mentioned webserver or the int.eu.grid RAS server.

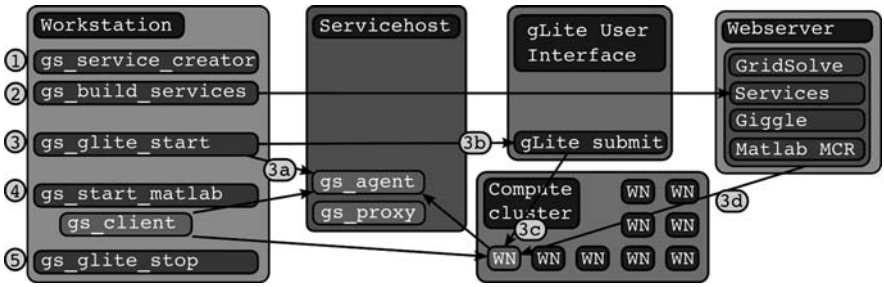


Fig. 20.4 Architectural building blocks of giggle and their interaction

Figure 20.4 shows these three additional computers and their interaction with the gLite hosts.

3.2 Giggle Tools for Creating Services

Creating services (i.e. making functions available remotely) via the GridSolve mechanism requires in principle only a few steps: definition of the interface and a recompilation using the GridSolve provided tool *problem_compile*.

However, while developing our own services for GridSolve on gLite, we found that it is better to include our services into the GridSolve build process and recompile the service together with the whole GridSolve package. This is more stable when installing the service and GridSolve in a different location on several computers. We also found that this approach holds when modifications to the underlying GridSolve sources (e.g. updates, fixes) are done. Thus, we have facilitated this process (with the tools gs-service-creator and gs-build-services) within giggle.

- `gs-service-creator` ((1) in Fig. 20.4) generates a directory structure that contains the required files for creation of a new service. The gs-service-creator is template based. Currently two templates – plain C RPC and Matlab compiler runtime RPC – are supported. Furthermore, a mechanism is included that supports the transport of dynamically linked shared libraries to the server.
- `gs-build-services` (2) is the tool that organises the compilation of GridSolve and the service. It hides the complexity of integrating our service into the GridSolve build system. The tool compiles the sources in a temporary location, where the GridSolve build specifications are modified to integrate the new service. The compilation process also recompiles GridSolve. This makes the build procedure longer, but we found that this ensures a higher level of reproducibility, flexibility and robustness against changes of the service or the underlying version of GridSolve. The output of the build process is a package (.tar.gz) file that contains the service. Optionally a GridSolve distribution tarball can be created. After a successful build, the generated packages have to be deployed on the webserver to be available for download by grid jobs.

3.3 Giggle Tools for Resource Allocation

The gLite resource allocation is done via the gLite User Interface (UI). We provide three tools that avoid logging into the UI and manage the involved gLite jobs. Furthermore the authentication to the grid is taken care of.

- `gs-glite-start` (3) launches the resource allocation. This tool starts a chain of several steps: First it starts up the GridSolve components (`gs-agent` and optionally the `gs-proxy`) on the servicehost (3a). Then it instructs (3b) the gLite User Interface (UI) to submit a given number of gLite-jobs (3c) to the resources of the interactive grid project. The jobs download (3d) and install required dependencies and start the GridSolve server. The `gs-server` connects to the `gs-agent`. Then the WorkerNode is available to the user via the GridSolve client.
- `gs-glite-info` can be used to display a short summary of the gLite jobs.
- `gs-glite-stop` (5) When the user is done using the grid, gs-stop-grid frees the allocated resources and terminates the GridSolve daemons. Otherwise resources would remain allocated but unused.

Currently these tools rely on password-less ssh in order to connect to the gLite User Interface machine. There the user commands are translated to gLite commands.

3.4 Giggle Tools for the End-User

`gs-start-matlab` (4) configures a user's Matlab session so that it can access GridSolve resources. This involves configuring Matlab to find the local GridSolve client installation as well as directing the client to the previously started `gs-agent`.

Please note that Matlab is only taken as an example representative of many other applications. Being available for Java, C, Fortran, Octave and more the GridSolve client can be used from various programming languages in different applications.

4 Measurements

We have implemented a test-scenario, using pilot jobs on the int.eu.grid infrastructure in the following way:

We submit the pilot jobs to int.eu.grid. This can easily be done via our GUI, the Migrating Desktop or from the gLite commandline. The int.eu.grid Resource Broker – the CrossBroker – selects the sites where the jobs are sent to. Once a pilot job is started at a particular WorkerNode, an installation procedure is carried out. It downloads and installs GridSolve, as well as the functions that we want to run on the WorkerNode. This way we minimise the data that requires to travel the path along CE, RB and UI. After the installation, the GridSolve middleware is started.

From this point on the user can see the WorkerNodes connected using the GridSolve tools. In our case using `gs_info`, since we use the Matlab client interface.

The performance evaluation is based on the CPU intensive loop benchmark. This benchmark allows to specify the number of loops or iterations and the amount of CPUs to be used. Advantages of the loop benchmark are

- no communication: this is an important factor for scaling
- linear scaling: two iterations take twice as long as one
- even distribution: every CPU computes the same amount of iterations.

With this benchmark we measured the performance characteristics of our solution. This will resemble both: the overhead introduced by GridSolve and an effect that originates from the different speeds of the CPUs that are assigned to the problem. This effect may be called unbalanced allocation.

Within this series of measurements we can modify three parameters:

- Number of gLite WNs. This is different to the number of CPUs, because we have no information about how many CPUs are installed in the allocated machines. GridSolve, however, uses all CPUs that are found.
- Number of processes that we use to solve our problem.
- Amount of iteration that we want to be computed.

We chose iterations between 10 and 1,000, which correspond to 6 s and 10 min runtime on a single Pentium IV-2400 CPU. The iterations were distributed to the resources provided by the production infrastructure of int.eu.grid. Resources were allocated at SAVBA-Bratislava, LIP-Lisbon, IFCA-Santander, Cyfronet-Krakow and FZK-Karlsruhe. Typically we have allocated 5–20 WorkerNodes (WNs), most of which contain 2 CPUs. The precise number of CPUs is unknown.

We measured the time to compute the iterations over the number of CPUs used. Measurements were repeated 10 times for averaging purposes. The graphs show the inverse of the computing time as a function of the number of processes into which we divided the benchmark. Out of the 10 repetitions of each measurement we show the average of all measurements as well as the maximum and minimum curves. The difference originates from the dynamic nature of the grid. If one server exceeds a time limit, it will be terminated. In this situation GridSolve chooses a different resource where this part is computed again. This leads to a prolonged computation of the whole benchmark, hence a difference between the “min” and the “max” curve. The min curve resembles a better result, while the max curve stands for the longest runtime of the benchmark. We refer only to the theoretical and the min curve further in this discussion. The graphs also show a theoretical curve, which is based on the assumption of ideal scaling (i.e. if doubling the number of processes, the result will be computed in half the time).

We can observe a difference between the maximum and the minimum curve. The reason for this difference is the unbalanced allocation of CPU speeds that we have mentioned above. Since these effects are of a random nature it is not possible to compare them between the different graphs.

Results are shown in Figs. 20.5, 20.6, and 20.7.

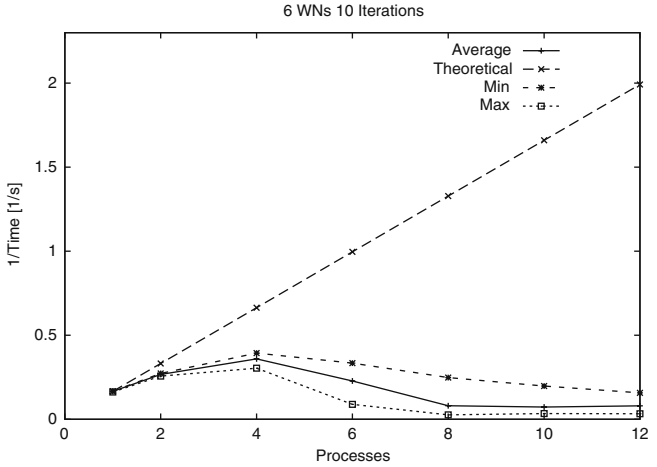


Fig. 20.5 6 gLite WNs allocated. 10 iterations require 6 s on a Pentium-IV CPU

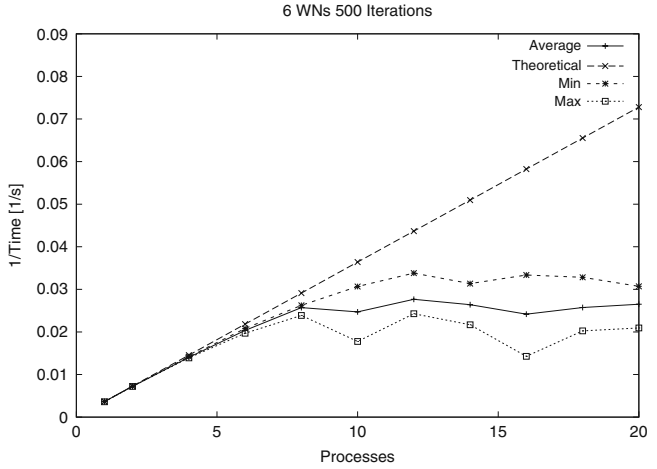


Fig. 20.6 Five hundred iterations run 5 min on a Pentium-IV CPU, 30 s on the grid

The comparison between a measurement with a low number of iterations (Fig. 20.5) and one with more iterations (Fig. 20.6) (6 s and 5 min, respectively) reveals the overhead that is caused by GridSolve. We observed that this overhead depends on the amount of CPUs that we utilise and occurs mostly during submission and collection of results. This is also indicated by the increased computing time for more than 4 CPUs in Fig. 20.5.

In all measurements we can find one point at which the curve stops following the theoretical prediction and continues horizontally. This indicates that the compute

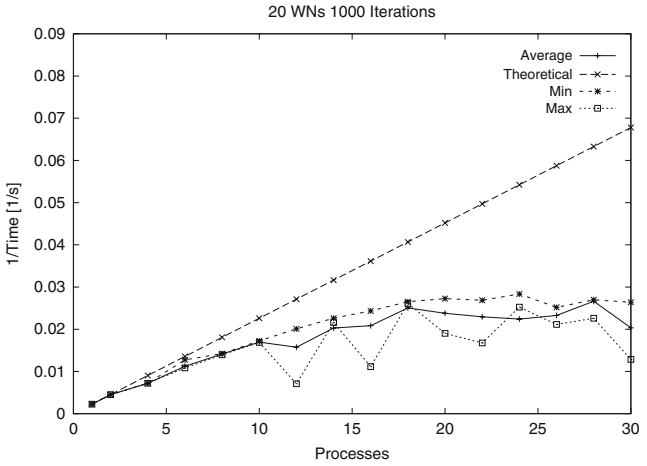


Fig. 20.7 With 20 WNs allocated and 1,000 iterations we can the system scale to 18 processes

time does not decrease even if further increasing the number of parallel processes. This is because at this point the number of available CPUs is smaller than the number of processes. The theoretical curve does not take this effect into account.

In Figs. 20.6 and 20.7 we can see that even the mincurve grows slower than the optimum (theoretical) curve. This is due to the heterogeneous nature of the grid. In our pool of resources, we have CPUs with different speeds. GridSolve allocates the fastest CPU first and only uses slower CPUs when all fast CPUs are already busy. As the theoretical curve is extrapolated from the performance on only one CPU, it resembles the ideal scaling behaviour. In Fig. 20.6 this can be observed at 6 and more processes, in Fig. 20.6 we can see this effect already starting with 4 processes.

5 Conclusions

gLite provides a powerful infrastructure for computing. However, it is very complex and difficult to be used by non-gLite experts.

Based on the experience of distributing an embarrassingly parallel application we provide a solution for two weak points of gLite using pilot jobs. With this solution the job abortions become tolerable. Furthermore, using GridSolve we have been able to show that it is possible to integrate the distributed gLite resources into standard programming languages, using the problem-solving environments Matlab as an example.

Our performance evaluation reveals that scaling of resource usage works in principle, but certain obstacles are still to overcome. The main obstacle regarding proper scaling is, however, the fact that the underlying resources disappear sometimes, so that the failed part has to be computed again. We expect a performance increase of a factor of two or more by simply improving the reliability of resources.

We have reached the goal of using the gLite infrastructure in an interactive, API-like fashion from problem-solving environments like Matlab.

References

1. CIO.com. Seven wonders of the IT world. http://www.cio.com/article/135700/4
2. H. Gemmeke and N.V. Ruiter. 3D ultrasound computer tomography for medical imaging. *Nuclear Instruments and Methods*, pp. A580, p. 1057–1065, 2007.
3. J. Gomes, G. Borges, M. Hardt, and A. Hammad. A Grid infrastructure for parallel and interactive applications. *Computing and Informaticsi*, Vol. 27, 173–185, 2008.
4. The Mathworks. Website. www.mathworks.com http://www.mathworks.com
5. EGEE Project. gLite website. http://glite.web.cern.ch/glite
6. Interactive European Grid Project. Website. http://www.interactive-grid.eu
7. R. Stotzka, J. Würfel, and T. Müller. Medical Imaging by Ultrasound–Computertomography. In *SPIE's International Symposium Medical Imaging 2002*, pp. 110–119, 2002.
8. A. YarKhan, J. Dongarra, and K. Seymour. GridSolve: The evolution of network enabled solver. In P. Gaffney and J.C.T. Pool, editors, *IFIP TC2/WG 2.5 Working Conference on Grid-Based Problem Solving Environments*, pp. 215–226, Springer, 2007. http://tinyurl.com/3bma6h
9. M. Zapf, G.F. Schwarzenberg, and N.V. Ruiter. High throughput SAFT for an experimental USCT system as matlab implementation with use of simd cpu instructions. *Medical Imaging 2008: Ultrasonic Imaging and Signal Processing*, 6920(1):692010, 2008. http://link.aip.org/link/?PSI/6920/692010/1

Chapter 21
Collaborative Interactivity in Parallel HPC Applications

Interactive Computational Steering of Grid Applications

M. Riedel, W. Frings, T. Eickermann, S. Habbinga, P. Gibbon, A. Streit, F. Wolf, and T. Lippert

Abstract Large-scale scientific research often relies on the collaborative use of massive computational power, fast networks, and large storage capacities provided by e-science infrastructures (e.g., DEISA, EGEE) since the past several years. Especially within e-science infrastructures driven by high-performance computing (HPC) such as DEISA, collaborative online visualization and computational steering (COVS) has become an important technique to enable HPC applications with interactivity and visualized feedback mechanisms. In earlier work we have shown a prototype COVS technique implementation based on the visualization interface toolkit (VISIT) and the Grid middleware of DEISA named as Uniform Interface to Computing Resources (UNICORE). Since then the approach grew to a broader COVS framework. More recently, we investigated the impact of using the computational steering capabilities of the COVS framework implementation in UNICORE on large-scale HPC systems (i.e., IBM BlueGene/P with 65536 processors) and the use of attribute-based authorization. In this chapter we emphasize on the improved collaborative features of the COVS framework and present new insights of how we deal with dynamic management of *n* participants, transparency of Grid resources, and virtualization of hosts of end-users. We also show that our interactive approach to HPC systems fully supports the necessary single sign-on feature required in Grid and e-science infrastructures.

Keywords Scientific visualization · Computational steering · COVS · VISIT · UNICORE

M. Riedel (✉)
Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany
e-mail: m.riedel@fz-juelich.de

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_21,

1 Introduction

Many e-science applications within Grids aim at simulation of a scientific domain-specific problem, and *parallel computing* is widely accepted to be an essential tool for the solution of these complex scientific problems. Such simulations of physical, chemical, biological or economical processes provide insights into phenomena that either cannot be addressed by experimental methods or would require experiments that are too expensive or dangerous. Grid applications that use parallel computing techniques for simulations have to use computers with multiple processors that are able to jointly work on one or more specific problems at the same time.

In many parallel simulations, for instance in *computational fluid dynamics (*CFD*)*, astrophysics, or weather and climate research, the tasks of simulation and visualization of the results are usually done as independent steps. First, a researcher performs a simulation for some fixed amount of computational time, and then analyses the computed results in a *separate post-processing step*, for instance by viewing the results in a dedicated visualization application.

On the contrary, several scientists have indicated a higher level of satisfaction by using an *interactive access and control* of their Grid applications during their run-time. That means it is often more efficient and more enlightening to perform both steps – simulation and visualization – at the same time by also steering certain parameters of applications during run-time. Hence, the scientists are able to observe the intermediate steps during the computation of the simulations (*online visualization*) and can interact with an ongoing parallel simulation to influence its computation (*computational steering*). Some examples where interactive computational steering is used are for parameter space exploration, in engineering applications to kick-start numerous events, calibrations of simulations, and several other use cases.

In this contribution we will highlight certain design concepts of the Collaborative Online Visualization and Steering (COVS) framework and its reference implementation within UNICORE [17], which allows for interactive access to Grid applications. While many works around COVS were published [15, 14, 12, 13, 16], this chapter emphasizes on features of how we deal with dynamic management of n participants, Grid transparency in terms of hostnames and ports to satisfy end-users without technical knowledge (e.g., single sign-on).

This chapter is structured as follows. Following the introduction the scene is set in Section 2 where we present the evolution of the COVS framework reference implementation in UNICORE and its core building blocks. Section 3 then highlights COVS framework features to seamlessly manage collaborative interactive Grid applications. Section 4 describes two parallel HPC applications that make use of the framework while a survey of related work is described in Section 5. The chapter ends with some concluding remarks.

2 Collaborative Online Visualization and Steering (COVS) Framework

The COVS framework enables HPC-based Grid applications with interactivity (i.e., computational steering) and visualized feedback mechanisms. In earlier work [15], we have shown a prototype COVS technique implementation based on the visualization interface toolkit (VISIT) [19] and the Grid middleware UNICORE. VISIT is a communication library that provides a loosely coupled approach between a parallel simulation on the one hand and its visualization on the other.

In principle, VISIT transports specific data sets between the visualization and the simulation and thus conveys scientific and additional information such as interactive (steering) commands. Following a broader requirement analysis for COVS [14], we observed that one of the most crucial design goals should be to minimize the load on the steered simulation and to prevent failures or slow operations of the visualization from disturbing the simulation progress on supercomputers. This leads to a design of VISIT where the e-science applications (simulations) act as a client which initiates all operations like opening a new connection, sending data to be visualized or receiving new interactive steering parameters. VISIT provides several components and most notably a VISIT server (integrated in visualizations), a VISIT client (integrated in simulations), and additionally a VISIT Collaboration Server and the VISIT Multiplexer to deal with collaborative scenarios.

With the successful integration of VISIT, our approach grew to a mature COVS framework implementation, whose evaluation [12] proved that the approach taken is feasible and provides sophisticated performance. Since then, we observed a trend toward higher degrees of parallelism by employing a larger number of moderately fast processor cores. As a result of these developments, supercomputer Grids will have to integrate peta-scale high-end computational resources with many cores in the future, which implies that also our COVS framework is used with new kinds of applications. Therefore, we investigated in [13] the impact of using the interactive computational steering capabilities of the COVS framework implementation in UNICORE on large-scale HPC systems (e.g., IBM BlueGene/P JUGENE with 65536 processors).

More recently, we presented in [16] an extension of the COVS framework with attribute-based authorization capabilities. This was particularly motivated by challenges that arise in geographically dispersed visualization sessions, creating a demand for a more fine-grained authorization based on attributes of end-users such as VO/project membership or different roles (e.g., steerer, collaborator, participant).

The current COVS framework reference implementation architecture is shown in Fig. 21.1 [12]. It illustrates a collaborative scenario with two geographically dispersed participants (i.e., client tier A and B). Both run on their machine a scientific

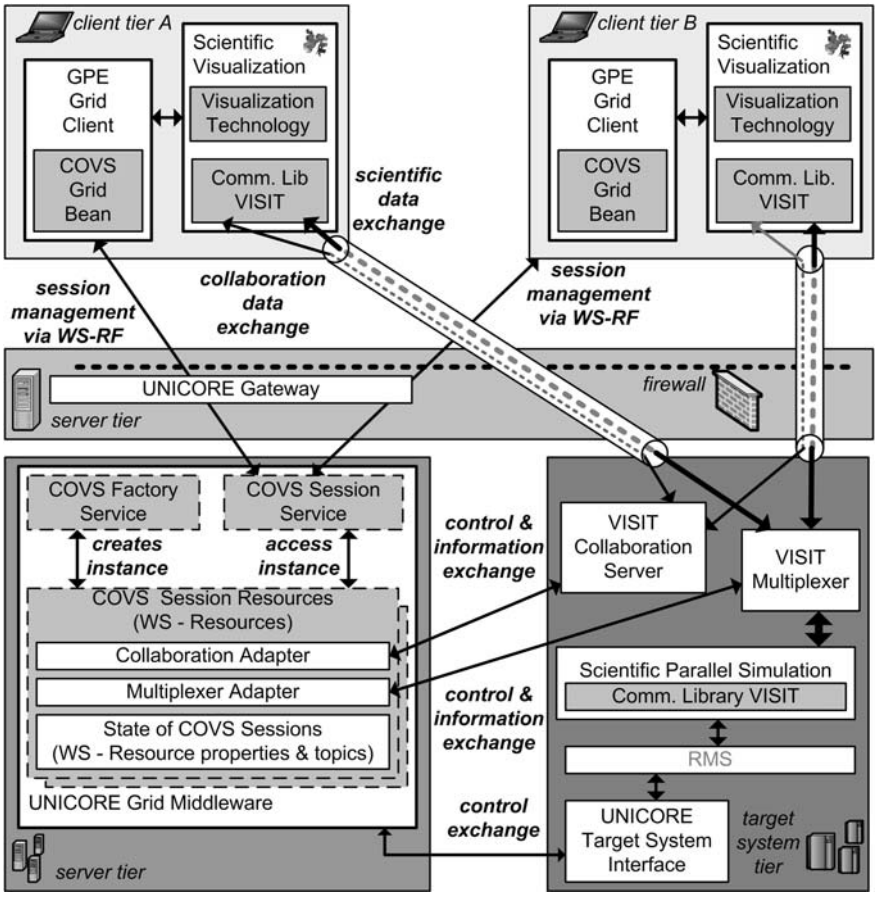


Fig. 21.1 COVS framework reference implementation in the UNICORE Grid middleware

visualization that interacts with a COVS GridBean plug-in, which extends the GPE UNICORE Grid client [11].

The Grid client is used to submit computational jobs to the Grid and to access two dedicated COVS services that are compliant with the Web Service Resource Framework (WS-RF) standard [20]. According to the factory pattern of WS-RF, the client is used to call a COVS Factory Service, which creates COVS Session resources that are in turn accessible via the COVS Service. Hence, one instance of such a session resource represents a collaborative visualization session. This session resource can be used to manage different participants by controlling the VISIT Multiplexer and the VISIT Collaboration Server. While the VISIT Multiplexer is responsible to distribute the outcome of one parallel simulation (i.e., scientific data) to *n* participants, the VISIT collaboration server is used to exchange information (e.g., turn angle of viewpoints) between the *n* participants. Both are using SSH connections

using the strong security features of UNICORE 6 more deeply explained in the next Section.

3 COVS Framework Features to Manage Interactive Grid Applications

The previous section introduced the core building blocks of the COVS framework implementation and described that the scientific data, steering data, and the collaboration data is transferred via secured dedicated connections with binary wire encoding using VISIT to achieve satisfactory performance. To ensure fast establishment of bi-directional connections as well as satisfactory performance, the Grid middleware of the COVS framework must allow for interactive access to the Grid resources bypassing the typically deployed resource management systems (RMSs) on supercomputers that do scheduling. In addition, the previous section also illustrated that the Grid application (i.e., parallel simulation) submission and management of collaborative sessions use Web service calls that in terms of the overall performance are non-critical. Nevertheless, the COVS framework implementation exposes an overview of the participants and their connection status during a COVS session to all participants. This includes information about the performance and status of each connection to identify bottlenecks.

It is also important to mention that the applicability of interactive steering depends on the type of the Grid application. The time for one computational step should not take too long in order to avoid long-winded online visualizations (no direct feedback) and to see the impact of steering almost immediately. On the other hand, the time for one computational step should not be too short, in order to give the end-user of COVS the chance to steer the parameters based on the current status, before the next data from the simulation is displayed. In order to allow for steering, the parallel Grid simulation must be parameterized in a way that allows the change of values during its run-time.

Since the SSH connections are critical, we give insights in this section on how they are created by still providing single sign-on. Hence, the seamless integration into Grids places the requirement on COVS components to remain single sign-on, which avoids specific logins or tedious password requests.

Also, this section will describe in detail how transparency of Grid resources in terms of hostnames and ports are realized within the COVS framework. Hence, the COVS framework implementation is typically well embedded in a broad Grid or e-science infrastructure by the meaning of hiding the fact that resources are physically distributed. This transparency includes differences in security policies, data representation, and how a resource is accessed when using services of the COVS framework.

Both described fundamental features of COVS are necessary in order to provide high levels of usability for end-users that have no technical Grid understanding like many e-scientists that are experts in their scientific domain but are not experts in distributed system technologies.

3.1 Secure Data Transfer and Interactive Access

The COVS framework relies on the usage of a bi-directional connection for secure data transfer between the parallel simulation and the visualizations with low latency. Lessons learned from earlier work [2] in the context of COVS revealed that if interactive steering data is transferred through several Grid components (e.g., Grid services) the overall latency will considerably increase compared to a direct connection. Therefore, the implementation of the COVS framework is based on SSH connections for the transfer of scientific and interactive steering data, also because firewalls often allow access via SSH to the protected systems (i.e., supercomputers) running parallel simulations.

Figure 21.2 illustrates that the visualization makes a connection request to the COVS Grid Bean (1). Afterward, and the COVS Grid Bean generates an SSH session key pair on the fly (2) and transports the public key to the UNICORE Grid site (3). The UNICORE Grid middleware in turn adds the key into the *$HOME/.ssh/authorized_keys* file (4) using a previously installed UNICORE software resource (sw-resource) [17] *visit_init.pl*.

The Grid client gets the resource-specific details such as the username, hostname, or the location of the VISIT *vttproxy* from this script as well (5). These pieces of information and the key pair are then transferred to the scientific visualization application (6). In particular, this application integrates the VISIT server of the communication library VISIT, which uses the transferred details to make a SSH connection request to the SSH daemon of the UNICORE Grid site (7). The daemon in turn checks whether the used session key is present in the *$HOME/.ssh/authorized_keys* file (8). After approval, the *vttproxy component* of VISIT is started and thus a bi-directional connection can be established (9).

Finally, the parallel simulation uses the VISIT client that sends the data to the *vttproxy* (10), which in turn forwards the data over the secure SSH connection to the visualization using the VISIT protocol. The *vttproxy* acts as a proxy for the visualization located in the same security domain as the simulation. The simulation can still decide when to connect to the *vttproxy*.

After the establishment of the connection, the sw-resource *visit_cleanup.pl* is used to remove the session key from the *$HOME/.ssh/authorized_keys* file. The key transfer and its addition to this file raise the demand for interactive access so that the establishment of the SSH connection is not scheduled via a Resource Management System (RMS) typically deployed on Grid resources. This is realized by using the INTERACTIVE label within the UNICORE configuration for the two sw-resources *visit_init.pl* and *visit_cleanup.pl*.

Furthermore, the usage of the Grid middleware UNICORE ensures that the implementation provides authentication and authorization of users. In particular, the Grid job from the GPE Client, which includes the execution of sw-resources, can only be executed if the user of the Grid client is correctly authenticated and authorized at the enhanced NJS [18] backend of UNICORE. Hence, and the implementation gains the benefit of the UNICORE security infrastructure since a non-authorized user is not able to establish the SSH connection and thus cannot use it for visualization.

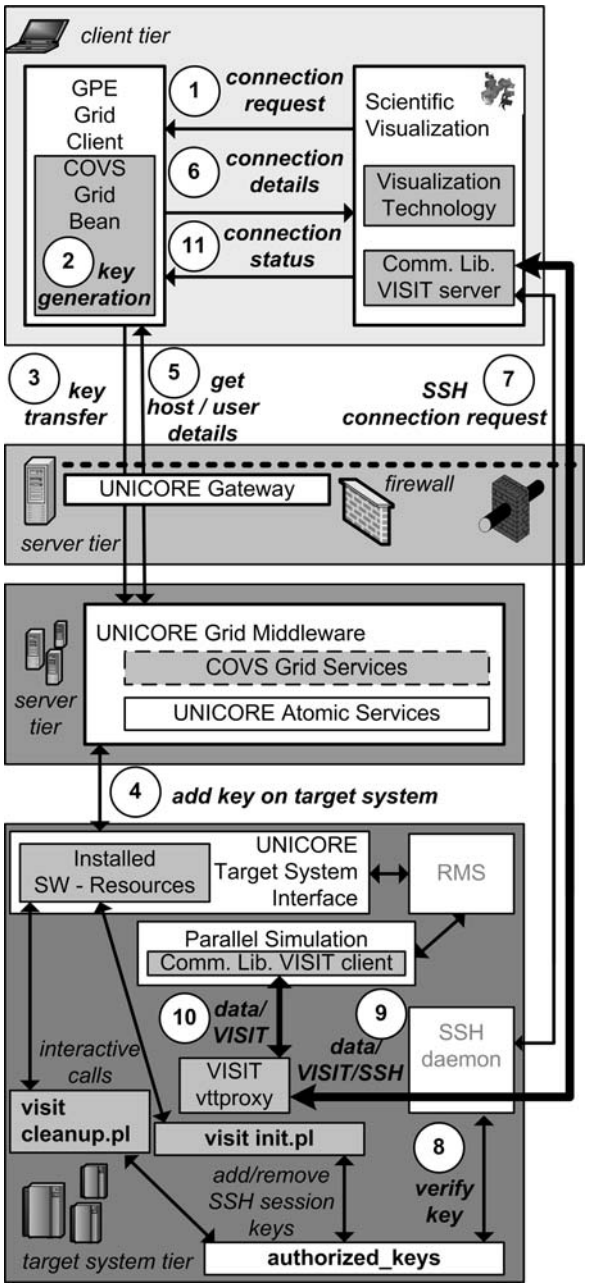


Fig. 21.2 Using interactive calls for the establishment of SSH connections

Also, the installation is very lightweight, because both sw-resources are simple perl-scripts that run directly on the supercomputer. Note that the same principle of establishment is also used in collaborative visualization sessions between the multiplexer and several visualizations also using the *vttproxy* component in between.

3.2 Naming Service for Decoupling COVS Components

One of the fundamental ideas of the COVS framework is to hide the complexity of the Grid from its end-users. That leads to the demand that they do not have to deal with hostnames, usernames, or other environment-specific details. This section provides details of how exactly the connections on the network-level between all the different components are realized by still hiding the necessity for hostnames, usernames, and ports from end-users. Hence, after the successful establishment of the SSH tunnel and start-up of the vttproxies as described in the previous section, the used VISIT components including the VISIT multiplexer, VISIT collaboration server, VISIT servers, and VISIT client must exactly know where to send data. In other words, these components require exact hostname and port information form each other.

The key to the understanding of this internal information exchange is a simple naming services provided by the VISIT toolkit named as the Service Announcement Protocol (SEAP) server [19]. This server allows for a decoupling of the parallel simulation and their visualizations. It basically maps a *seapservice:seappassword* to a set of information that comprises a hostname and port. Here, the *seapservice* is used for the identification of a client tier or simulation, while *seappassword* is used for the identification of a COVS session. Hence, it does not deal with authorization based on passwords.

One of the major requirements of the COVS framework is to prevent disturbances of the simulation progress by any online visualizations that are connected to it. Therefore, the parallel simulation itself connects to the VISIT multiplexer component, which in turn needs to connect to multiple visualizations using the vttproxies of each visualization, as shown in Fig. 21.3. Hence, the parallel simulation needs to know the hostname and port of the vttproxies that forward the data through the SSH connections to the visualizations.

Another key to the understanding relies in the COVS Grid Bean implementation that interacts with the COVS Grid service, which exposes ongoing COVS session information via WS-Resource properties [20]. In the context of the SEAP server, the COVS Grid Bean is used to set up a COVS session, which leads to the creation of a *seappassword* (e.g., svc1) for this session. This identifier is kept internally within the implementation; instead, the end-users see a human readable name for this session, which is mapped on the internal identifier.

In the context of Fig. 21.3, some end-user (e.g., client tier C) has already created such a session that is internally identified by *svc1* as the *seappassword*. The human readable name related to this identifier is exposed via WS-Resource properties to

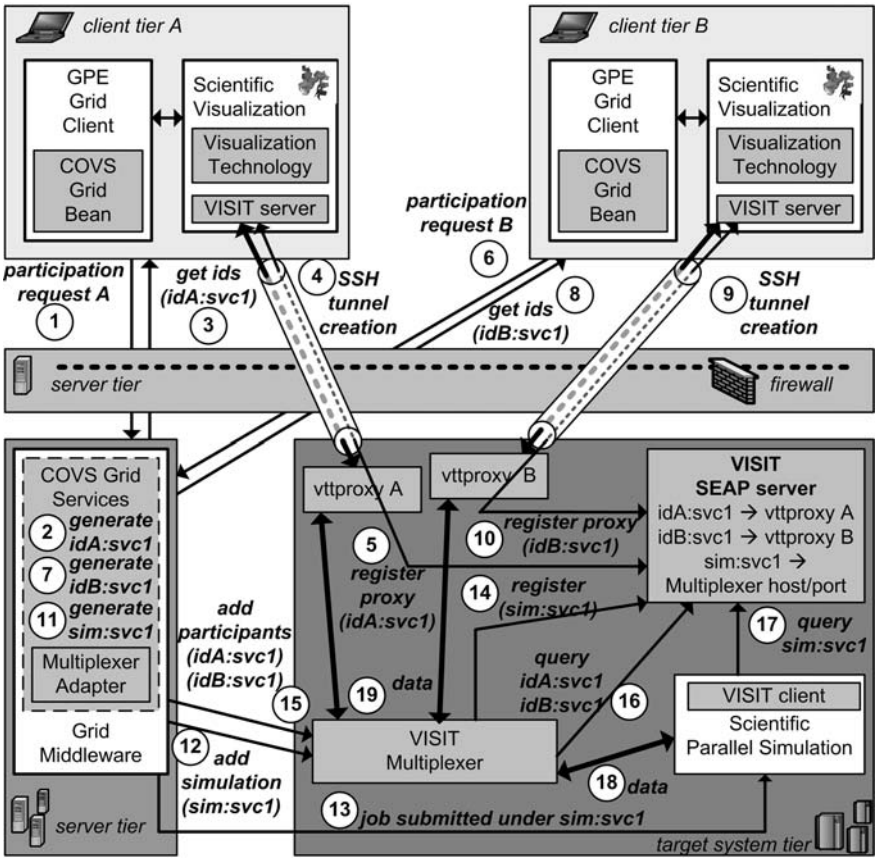


Fig. 21.3 Using the VISIT SEAP server as naming service. Two geographically dispersed participants are dynamically connected using seapservice:seappassword abstractions. SEAP information exchanges are also tunneled via SSH where necessary

the client tier A and B. Therefore, the COVS Grid Bean shows this particular session and both client tier A and B are able to request for participation in this particular session as follows.

As illustrated in Fig. 21.3, client tier A makes a participation request for a specific session at the COVS Grid service using Web service message exchanges (1). Internally, this session is identified by *svc1*. Next, the COVS Grid service generates a *seapservice idA* for this participant (2). Hence, the combination *idA:svc1* identifies a particular user that participates in a specific COVS session. This combination is sent back to the COVS Grid Bean (3). Afterward, the COVS Grid Bean creates the SSH tunnel using the mechanisms described in the previous sections (4). In order to publish the location of the started vttproxy A, the combination *idA:svc1* is registered at the SEAP server (5) and maps to the specific hostname and port of the vttproxy A.

The same steps occur when client tier B makes a participation request (6). This leads to the generation of the combination *idB:svc1* (7). Furthermore, this combination is sent back to the client tier B (8). Afterwards, the COVS Grid Bean on client tier B creates a SSH tunnel (9) and registers the location of vttproxy B under the combination *idB:svc1* at the seap server (10). Note that all these pieces of information are transferred via secure Web service message exchanges or using the secure SSH tunnel.

While the end-users on client tier A and B wait for connection to the simulation, the parallel simulation job is submitted through the Grid middleware UNICORE, for instance by an end-user at client tier C (not in the figure). This leads to a generation of the *sim:svc1* combination at the COVS Grid services (11). Hence, this combination identifies a particular simulation that provides the data for a specific interactive COVS session. This information is given as an input to the VISIT multiplexer (12).

Furthermore, this particular simulation is submitted to the underlying Grid resource by providing the *sim:svc1* combination for identification of the simulation (13). The multiplexer registers the combination *sim:svc1* at the SEAP server that maps on the location of the multiplexer (hostname and port) (14). This is necessary to provide the simulation with information where to send the data. But before any data is sent, the combinations *idA:svc1* and *idB:svc1* are added to the multiplexer (15). Therefore, the multiplexer is able to query for the exact location of vttproxy A and vttproxy B using these combinations (16) and to establish a connection to them.

In the meanwhile, the parallel simulation queries the SEAP server using its identification combination *sim:svc1* in order to know the exact location of the multiplexer (17). This information is then used to send the scientific data of the parallel simulation to the multiplexer (18). Next, the multiplexer sends this data via the connections to the vttproxies (19) that in turn forward the data to the corresponding visualizations through the SSH tunnel. Of course, this established mechanism works also for the interactive steering control information that is using the same bi-directional channels and identification combinations.

Also, Fig. 21.3 reveals the unique design of VISIT since the VISIT servers are integrated into the visualizations at the client-side, while the VISIT client is integrated into the parallel simulation. This makes the *parallel simulation act as a client* that initiates all operations like opening a connection or sending the data to be visualized or receiving new interactive steering parameters. Finally, the exact hostname and port of the SEAP server itself are typically configured within an environment variable at the sites that run VISIT-enabled components. To sum up, by using the SEAP server within the COVS framework implementation all hostnames and ports of COVS session participants are dynamically configured at run-time, which allows for maximum flexibility for ad-hoc participating end-users.

3.3 Enable Collaboration with Naming Service

Similar to the decoupled communication with the VISIT multiplexer and vttproxies as described in the previous section, the VISIT collaboration server uses also the

SEAP naming service for decoupled communication. The collaboration server is used to exchange collaboration data between the multiple visualizations that participate in a COVS session. In particular, the CollaborationAdapter within the COVS Grid service starts the VISIT collaboration server that in turn registers itself under a combination of *collaseapservice:collaseappassword* at the SEAP server. This means the combination maps to the correct hostname and port of the collaboration server on the target system tier.

In addition, the *collaseapservice:collaseappassword* combination is also exposed via WS-Resource properties of the COVS Grid service. Therefore, the COVS Grid Bean gets this information via WS message exchanges and transfers it to the scientific visualization via the COVS GridBean. After this transfer is done, the scientific visualization uses this information to query the SEAP server through the SSH tunnel in order to get the correct port and hostname of the collaboration server. Hence, the scientific visualization establishes a connection to the collaboration server through the SSH tunnel using the SEAP server. This allows for dynamic configurations of the VISIT collaboration server; for instance, it can be deployed on another host than the VISIT multiplexer (decrease load on host), thus making it necessary to establish another SSH tunnel to this particular host.

To sum up, all collaboration data that is exchanged between the collaboration server and multiple visualizations is securely tunneled via SSH tunnels, including the SEAP information exchange.

4 Interactive Computational Use Cases

In this section we review how the COVS framework implementation is used in interactive use cases with parallel HPC applications that solve n-body problems. N-body problems appear in many scientific domains such as astrophysics, plasma-physics, molecular dynamics, or fluid dynamics. N-body problems are concerned with determining the effect of forces between bodies (i.e., particles) [21]. In the domain of astrophysics several scientists use the COVS framework implementation with the Nbody6++ [10] code, which is a parallel variant of the Aarseth-type N-body code nbody6 suitable for N-body simulations on massively parallel resources.

The scientists use the framework with different subversions of the main code Nbody6++ for different areas of astrophysical research. These areas are dynamics of star clusters in galaxies and their centers, formation of planet systems, and dynamical evolution of planetesimal. In this context, the interactivity enabled via the framework allows for the exploration of parameter spaces through computational steering. In addition, the collaborative session support allows new forms of collaboration and experience exchange with geographically dispersed e-scientists within the AstroGrid-D, which is the astrophysical VO within the German National Grid D-Grid [4].

Another interactive use case in the area of e-science applications that represent N-body problems are several routines of the Pretty Efficient Parallel Coulomb solver

(PEPC) [9]. Plasma-physics scientists use the COVS framework implementation with these massively parallel codes that use a hierarchical tree algorithm to perform potential and force summation of *n* charged particles in a time O(n log n). This allows for mesh-free particle simulation on length- and timescales usually possible only with particle-in-cell or hydrodynamic techniques.

One particular use case of this code is for the simulation of a particle accelerator via laser pulses and the interactive approach of COVS realizes an interactive control of the laser target configuration. This in turn is very useful to verify start-up parameters such as the initial alignment of laser and target or to perform quick test runs with a small set of particles as a prelude to full-blown production runs on large-scale systems.

In the context of large-scale systems (e.g., peta-scale systems) we foresee that any many-body code which is attempting to model macroscopic systems will benefit from higher particle numbers. This is because on the one hand it improves the overall statistics by better reproducing the mathematical set of equations used to describe the system. On the other hand the large-scale systems permit simulation of larger, often more realistic systems (i.e., a galaxy in astrophysics).

Basically, we learn from both our COVS framework use cases that there is no upper limit on the number of simulation particles, but some codes are still in the process of getting scalable toward high amount of cores. This is particularly the case for codes that keep local copies of certain global data and are thus rather memory bound. But this could be overcome with some code restructuring, which should permit scaling up to high amount of processors. In this context, also computational steering methods of the COVS framework have to be revised, especially in terms of the well-known master-slave pattern. This means the master collects the data from all processors and sends it to the visualizations. The current trend is that more than one masters are used to collect the data and to send the data out, but this is still work in progress and has to be researched more deeply in terms of scalability toward, peta-scale systems.

5 Related Work

In the context of our work and the related work in the field interactivity mostly refers to the capability of a framework for computational steering. The UK RealityGrid project provides a steering library [1] that enable calls which can be embedded into its three components that are simulation, visualization, and a steering client. While the COVS framework is loosely coupled to the Grid middleware UNICORE and could be in principle used with any other WS-RF compliant Grid middleware (e.g., Globus Toolkit [3]), the RealityGrid steering library is much more tightly integrated with the Imperial College e-Science Networked Infrastructure (ICENI) [7].

Related work in the field can be also found in the NAREGI project [8] that offers a visualization system that consists of a visualization library and a Grid visualization API [5]. Even if this framework is also WS-RF compliant and thus similar to our

work, the steering and interactive capabilities are rather limited (e.g., turn views on the simulation).

Finally, another well-known work with interactive behavior is the Grid Enabled Visualization Pipeline (GVID) [6], which offers interactive capabilities via Event-Encoders that run on thin clients and sends steering commands to the simulation. The major difference to our approach is that it is not seamlessly integrated as a higher level service into a common Grid middleware.

6 Conclusions

This chapter describes how the computational steering and online visualization with the COVS framework implementation can be used within todays Grid and e-science infrastructures. The fundamental benefit of our approach is that the COVS framework implementation can be used by any parallel application that is instrumented with the VISIT toolkit to gain interactive functionality in terms of computational steering.

According to our evaluations we proved that the SSH connections and the VISIT protocol that is used between all VISIT components provide reasonable low latency to interactively steer a HPC application conveniently. This also implies the real-time feedback per visualizations to observe changes in the application during run-time according to the steered parameters.

Also, we have shown how the SEAP server is used within our framework to solve the Grid transparency issues. By using the SEAP naming service within the COVS framework implementation all hostnames and ports of COVS session participants are dynamically configured at run-time, which allows for maximum flexibility for ad hoc participating end-users that have no technical knowledge about the distributed nature of the Grid.

To conclude, our interactive approach with a dedicated data and steering channel provides much better performance than using XML-based Web service message exchanges for large data transfer, while Web services are well suited for the management of the connections and the control of collaborative COVS sessions.

References

1. J. Cohen, A.S. McGough, J. Darlington, N. Furmento, G. Kong, and A. Mayer. RealityGrid: An integrated approach to middleware through ICENI. *Philosophical Transactions of The Royal Society A*, 363:1817–1827, 2005.
2. Th. Eickermann, W. Frings, P. Gibbon, L. Kirtchakova, D. Mallmann, and A. Visser. Steering UNICORE applications with VISIT. *Philosophical Transactions of the Royal Society Journal*, 2005.
3. I. Foster. Globus Toolkit version 4: Software for Service-Oriented Science. In *Proceedings of IFIP International Conference on Network and Parallel Computing*, vol. LNCS 3779, pp. 213–223. Springer-Verlag, 2005.
4. German National Grid Initiative D-Grid. `http://www.d-grid.de`

5. P. Kleijera, E. Nakano, T. Takei, H. Takahara, and A. Yoshida. API for grid based visualization systems. In *GGF 12 Workshop on Grid Application Programming Interfaces*, 2004.
6. T. Köckerbauer, M. Polak, T. Stütz, and A. Uhl. GVid – Video coding and encryption for advanced grid visualization. In *Proceedings of the first Austrian Grid Symposium*, Linz, 2005.
7. A. Mayer, S. McGough, N. Furmento, J. Cohen, M. Gulamali, L. Young, A. Afzal, S. Newhouse, and J. Darlington. ICENI: An integrated Grid middleware to support e-Science. *Component Models and Systems for Grid Applications*, pp. 109–124, 2005.
8. NAREGI Project. `http://www.naregi.org`
9. S. Pfalzner and P. Gibbon. *Many-body Tree Methods in Physics*. Cambridge University Press, 1996. ISBN-10: 0521019168.
10. E. Khalisi R. Spurzem. *Nbody6 Features of the Computer Code*, 2003. `ftp://ftp.ari.uni-heidelberg.de/pub/staff/spurzem/nb6mpi/nbdoc.tar.gz`
11. R. Rateringa, A. Lukichev, M. Riedel, D. Mallmann, A. Vanni, C. Cacciari, S. Lanzarini, K. Benedyczak, M. Borcz, R. Kluszcynski, P. Bala, and G. Ohme. GridBeans: Supporting e-Science and grid applications. In *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*, Amsterdam, 2006.
12. M. Riedel, Th. Eickermann, W. Frings, S. Dominiczak, D. Mallmann, T. Düssel, A. Streit, P. Gibbon, F. Wolf, W. Schiffmann, and Th. Lippert. Design and evaluation of a collaborative online visualization and steering framework implementation for computational grids. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, Austin, USA, 2007.
13. M. Riedel, Th. Eickermann, S. Habbinga, W. Frings, P. Gibbon, D. Mallmann, F. Wolf, A. Streit, Th. Lippert, F. Wolf, W. Schiffmann, A. Ernst, R. Spurzem, and W.E. Nagel. Computational steering and online visualization of scientific applications on large-scale HPC systems. In *Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, Bangalore, India, 2007.
14. M. Riedel, W. Frings, S. Dominiczak, Th. Eickermann, T. Düssel, P. Gibbon, D. Mallmann, F. Wolf, and W. Schiffmann. Requirements and design of a collaborative online visualization and steering framework for grid and e-Science infrastructures. In *Online Proceedings of German e-Science Conference*, Baden-Baden, 2007. `http://edoc.mpg.de/display.epl?mode=doc&id=316630&col=100&grp=1414`
15. M. Riedel, W. Frings, S. Dominiczak, Th. Eickermann, D. Mallmann, P. Gibbon, and Th. Düssel. VISIT/GS: Higher level grid services for scientific collaborative online visualization and steering in UNICORE grids. In *Proceedings of 6th International Symposium on Parallel and Distributed Computing 2007 (ISPDC2007)*, Linz, Austria, 2007. ISBN 0-7695-2936-4.
16. M. Riedel, W. Frings, S. Habbinga, Th. Eickermann, D. Mallmann, A. Streit, F. Wolf, Th. Lippert, Extending the collaborative online visualization and steering framework for computational grids with attribute-based authorization. In *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing* (GRID), Tsukuba, Japan, pp. 104–111, 2008.
17. A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder. UNICORE – From project results to production grids. *Grid Computing: The New Frontiers of High Performance Processing, Advances in Parallel Computing*, 14:357–376, 2005.
18. UNICORE Website. `http://www.unicore.eu`
19. Visualization Interface Toolkit (VISIT). `http://www.fz-juelich.de/zam/visit`
20. Web Services Resource Framework (WSRF) Technical Committee (OASIS). `http://www.oasis-open.org/committees/wsrf`
21. B. Wilkinson and M. Allen. *Parallel Programming*. Prentice Hall, 1999. ISBN 0-13-671710-1.

Chapter 22
Interactive and Real-Time Applications on the EGEE Grid Infrastructure

E. Floros and C. Loomis

Abstract EGEE is currently the flagship European grid project having established the largest grid infrastructure worldwide. This infrastructure is routinely used on a daily basis by more than 13,000 users coming from a large number of scientific disciplines, including but not limited to high energy physics, fusion physics, computational chemistry, life sciences, earth sciences, astronomy, and astrophysics. Many of these grid-enabled applications require real-time interactions with the end users. This chapter presents the capabilities offered within EGEE for supporting interactive applications. More specifically, we will see that interactivity requires support from multiple layers of a grid architecture starting from the grid middleware itself, the grid site configuration, and the network infrastructure. Moreover, user- and VO-level solutions have been proposed to overcome limitations posed from the lower middleware layers. Lastly, we present a number of real-life applications exploiting interactivity on the EGEE grid.

Keywords Interactivity · Real-time applications · Grid · EGEE project

1 Introduction

Grid computing has become an important tool for a wide range of scientific disciplines used to perform daily research that exploits a large amount of distributed computing resources. International research projects and collaborations have been established in the past years with the aim to develop software tools and deploy large scale grid infrastructures to enable scientific research on the grid.

The enabling grids for E-sciencE (EGEE) [8] is presently the largest international grid project, having established the biggest grid infrastructure spanning three continents and bringing together scientists from more than 40 countries. This infrastructure has attracted numerous scientific disciplines including high energy physics (HEP), life sciences, earth sciences, computer science, astronomy, astrophysics,

E. Floros (✉)
GRNET, Athens, Greece
e-mail: efloros@grnet.gr

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_22,

and computational chemistry. Each discipline has diverse requirements for the grid infrastructures. These requirements have often tested the grid architecture. In some cases the grid successfully fulfilled the requirements; in other cases certain compromises had to be imposed on the users in order to work on the grid.

Real-time interactivity and fast application responsiveness are requirements that have been proven especially demanding for the grid infrastructures. Interactivity is required, for instance, by applications implementing remote instrumentation techniques. In these applications, software running on the grid is used to steer scientific instruments and other devices connected to the network. This software in turn interacts in real time with the end users who provide input for the steering activities, requiring near real-time responsiveness from them. Real-time interaction has challenged the grid architecture and has spawned various projects that have worked specifically on improving interactivity on the grid.

In this chapter we present the basic challenges that the grid has to face in order to improve interactivity and we explain why this is a non-trivial problem. Then we focus on the EGEE project. We detail the support currently provided by the EGEE grid not only focusing on the middleware capabilities but also analyzing the problem from the network and application points of view.

The rest of the chapter is organized as follows: In Section 2 we present the EGEE project and we describe the grid and middleware architectures it has established underlying the problem with interactivity. In Section 3 we analyze the various approaches that have been implemented to enable real or pseudo-real-time interaction on the EGEE grid. In doing this, we follow a bottom-up approach by first showing how the middleware itself can facilitate interactivity and then moving up through the layers of the architecture successively to the site configuration of local schedulers and to the application and user-level approaches. We devote Section 4 to analyzing the network factor in improving application interaction, presenting the work performed in the project in this direction. In Section 5 we present real-life applications currently running on EGEE and take advantage of the types of interactivity analyzed in Section 3. Finally we summarize the presented work in Section 6 with a reference to related work.

2 The EGEE Grid Infrastructure

2.1 EGEE Project

The enabling grids for E-sciencE is the leading grid computing project co-funded by the European Commission. The project's main objective is to deploy and operate a large-scale, production grid infrastructure for e-Science. This infrastructure currently comprises 260 production sites spread over 48 countries, providing a rough total of 80,000 CPU cores and more than 20 petabytes of storage. This infrastructure is routinely used on daily basis by more than 13,000 users grouped into more than 200 virtual organizations (VOs) running around 140,000 jobs per day. The project's

first phase was initiated in 2004 and currently has entered its third and final 2-year phase with 42 partners, involving more than 120 beneficiaries coming from 48 countries.

In addition to grid operations, the project provides support and training to end users and large scientific communities. EGEE actively engages with application communities, beginning with high energy physics (HEP) and Biomedicine at the start of EGEE but expanding to support many other domains. These include astronomy, computational chemistry, earth sciences, finance, fusion, and geophysics, with many other applications currently evaluating the infrastructure. EGEE also works with industrial users and industrial service providers to ensure technology transfer to business.

2.2 Middleware Architecture

The gLite middleware [12] provides the core functionality available on the EGEE grid infrastructure. Including software developed both by EGEE itself and by other projects, gLite provides a core software stack on which to run scientific applications. For the end user, gLite offers high level services for scheduling and running computational jobs, accessing and moving data, and obtaining information on the grid infrastructure as well as grid applications, all embedded into a uniform security framework.

gLite provides a two-layer batch-oriented grid. The first layer comprises a centralized workload management system (WMS) that receives requests to execute an application on a large pool of distributed resources hosted in the different gLite sites. Each site provides at least a computing element (CE) service which is the head of the local computing cluster and a number of associated batch queues. When the application reaches a site, a local batch system like PBS, Torque, LSF, or Condor takes control and is responsible for allocating the appropriate number of worker nodes (WNs) and initiating the application execution. The WMS is coupled with a logging and bookkeeping (LB) service, which is responsible for storing information about the job execution lifecycle. The user can optionally bypass the first layer of batch scheduling by explicitly defining at submission time the target site for execution. In this case only the local resource manager system (LRMS) is engaged in the source selection process. Nevertheless, it is still responsibility of the centralized WMS/LB node to monitor the progress of application execution and initiate data transfers between the CE and the end user workstation (usually referenced as a user interface or UI).

2.3 Grid Interactivity

Production grid infrastructures have traditionally targeted high-throughput batch processing. The main execution unit of such grids, and EGEE grid in particular,

is the job. A job abstracts a software application bundling the application executable together with additional artifacts like support data, configuration files, execution parameters, and other metadata. Typically, once a job leaves the UI the user has limited interaction capabilities. Most of the times he/she can manually query the execution state (scheduled, running, completed, aborted, and failed) and, upon successful termination, retrieve the output back to the UI.

This model of work is satisfactory for a large class of applications. HEP, for instance, which was one the first scientific communities to exploit EGEE grid and still remains one of the project's strongest players, has limited requirements for job interaction. Rather, HEP applications are interested in batch processing of large amounts of scientific data with little or no restrictions regarding execution time length and real-time interaction with the application. Notice also that the above puts limited requirements regarding scheduling times, that is the time that the job stays in the queue before it is allocated to a specific WN for execution.

Nevertheless, since grid has attracted more and diverse scientific disciplines, the requirements for interactivity have increased significantly. Many applications are not content with the one-off job submission pattern, but rather require real-time interactivity with the application while the latter is still in the running state. Examples of such applications are, for instance, remote instrumentation applications in astronomy that require continuous access, monitoring, and management of scientific instruments connected to the grid. Numerical simulations in fusion physics need steering, for instance, to alter the simulation parameters thus moving the execution model between different states. Medical data management applications have similar needs. The requirement for interactivity also influences the scheduling time restrictions placed on jobs. Most of the time it does not make sense to wait for many hours in order to start an application that in the end will interact in real time (or more accurately in pseudo-real time) with the end user.

3 Enabling Interactivity on the EGEE Grid

3.1 Interactive Jobs

Until recently, the gLite WMS supports a basic mechanism to run interactive jobs by explicitly defining this in the respective job description file (JDL), where all the details regarding job execution are defined from the user side. By declaring a job to be interactive, the WMS upon the application instantiation on the target WN will open three channels providing access to the process's standard streams (stdout, stderr, and stdin). This pseudo-real-time interaction is sufficient only for console-type interactions where the end user inputs commands from the stdin and interacts with the applications from the stdout and stderr. This does not support, for instance, interaction through user-defined channels or protocols.

From the point of view of the end user experience, usually it takes time for the messages to go back and forth between the user and the application leading to a

limited style of interaction. For this reason, this type of job has limited usability and is useful only for test runs and potential simple debugging of an application. For this reason it has already been decided that future versions of gLite will discontinue support for interactive jobs, which will become a deprecated feature.

3.2 Data Management Interactivity

A more interesting middleware supported capability is that of job perusal, which provides a data-level interactivity. In job perusal the user can, at any moment while the job runs, retrieve the output and any partial results produced so far by the job. This can help significantly in monitoring the progress of an application execution. More importantly, it can be extremely useful in cases that a job fails but partial results are still useful and can be used for input for the next runs. Job perusal is activated by setting the JDL job perusal attribute to true and optionally setting a time interval between each snapshot that is taken from the output files. The job is submitted normally using the *glite-wms-job-submit* command. After the job has been accepted for scheduling by the WMS, the user uses the *glite-wms-perusal* with the *set* parameter to define the file that he/she would like to follow and while running the same command with the *get* parameter to retrieve the partial results of a file whose perusal has been activated.

The drawback of this approach is apparently the fact that one can only retrieve output and not alter the input or steer the execution from stdin as with the interactive jobs. Another problem is that jobs perusal places increased pressure and load on the WMS; thus the user should limit the number of perusal files, otherwise there is a risk of performance degradation on the WMS.

3.3 Support from the Logging and Bookkeeping Service

The logging and bookkeeping (LB) service tracks jobs managed by the gLite WMS. It gathers events from various WMS components in order to give a higher level view of the job status. Typically logging information is placed and updated from the WMS components, reflecting the execution status of the job and potentially flagging problems and failures providing details that can help track possible problems either of the application or of the middleware itself. The user interacts with the LB service using the *glite-wms-job-status* command that returns the latest info regarding the job status as reported by the LB.

A rather underutilized capability is that the job itself can set status information on the LB. This gives the ability to the programmer to define and monitor custom status information about the application. Interactivity is through the UserTag event. A user tag is an arbitrary "name-value" pair that can be used to assign additional information to a job. In principle this information can be more fine-grained and detailed and can provide a better understanding of the application's status, behavior, and potential problems. This interaction can be of two ways:

- The application can set logging values while running, using the *glite-lb-logevent* command, which can be queried by the user.
- The user manually updates logging values with the same command that the application queries in specific (developer defined) checkpoints and adjusts its execution path appropriately.

One obvious drawback with this approach is that the application has to be altered in order to support this kind of interactivity. Also the interaction semantics are rather limited since they are depending on a simple, name-value pair and do not support streaming type of interactions like in the interactive jobs approach.

3.4 Low Latency Scheduling

The time a job remains in the queue comprises a major bottleneck for interactive and real-time interactions. Many applications have the requirement to minimize the time between job submission (job leaves the UI) and job execution (job starts running on the WN) or in other words the middleware should provide low latency scheduling capabilities. In order to face this challenge, EGEE formed the short deadline job (SDJ) working group [10] to investigate the problem and propose ways to improve the situation. By definition a SDJ:

- is a job with deadline constraint,
- has short execution time, and
- is unexpected and urgent.

Typically, SDJs have a relatively short execution time and they have to complete within a reasonable amount of time. In other cases the results produced by SDJs have temporal characteristics and are valid only within a limited time frame. Late execution and thus generation of results might not be sufficient for many applications (e.g., a fire propagation model or a flood simulation). For the above cases, best-effort execution services are not sufficient.

The SDJ working group has worked on the above requirements and has proposed local queue configuration that minimizes the waiting time. In these LRMS queues a SDJ will be scheduled immediately or fail. Unfortunately, this capability remains underutilized since there is still limited adoption from EGEE production sites. Nevertheless, there is an increased pressure from various user application communities to support them, and thus we expect the number of sites supporting SDJs to increase in the near future.

3.5 Pilot Jobs

More of a concept than a concrete technology, many applications have utilized the pilot job approach to achieve fast job initialization and increased execution throughput. A pilot job (also known as application agent) does not execute a simple run

of an application; rather, it is used to acquire, retain a WN and reuse it for multiple runs of the same or even different program executables. This means that the pilot job may not release a WN for days allocating resources from the grid even if there is no real processing taking place at a given moment. This characteristic of pilot jobs has rendered them to be considered unfair with respect to the grid computing philosophy and that excessive utilization of this approach might lead to grid resource saturation in cases that a large number of resources are allocated from the same application in order to enable multiple runs based on a user-level or VO-level scheduling policy. Moreover, in many cases the same pilot job might be used by different users overriding in essence the grid security mechanisms and policies and introducing potential security holes.

Despite the skepticism and the criticism they have received due of the above deficiencies, pilot jobs have been used for production purposes by higher level grid application programming environments like DIRAC [6] and DIANE [18]. Recently, *glexec* [14] has been introduced as an official tool for supporting pilot jobs on the EGEE infrastructure and is currently under certification for inclusion in one of the upcoming gLite releases. It alleviates the security problems of pilot jobs by switching the security context of the job depending on the user running the program through the pilot job. Obviously the problem of fairness still remains and it is up to the user or the VO to avoid over-exploitation of this capability.

3.6 Interactivity Through Support Tools

Many applications follow user-level approaches in order to overcome the lack of sufficient interactivity support from the middleware. One such example is the *glogin* and *i2glogin* [20] tools developed by GUP Linz and extensively used by the applications running in the *int.eu.grid* [16] project. The glogin package provides support for interactive connection to a running process. It enables port forwarding, in order to tunnel efficiently any protocol through the firewalls blocking worker nodes. Essentially glogin allows interactive login into any CE or WN where one is allowed to run a job.

4 Network Support

As with all grid applications, network performance is a principal bottleneck for achieving high execution speed. This is especially true in the case of interactive applications and generally those requiring fast responsiveness like remote instrumentation applications. EGEE builds upon GÉANT [9], the pan-European Research Network, that connects 34 countries providing the backbone among 30 National Research and Education Networks (NRENs). GÉANT provides mechanisms to request high network quality of service (QoS) through service level agreements (SLAs) that provide premium IP (PIP) services. PIP is a high-priority, low latency

QoS service for high quality demanding IP flows. PIP services reduce an IP packet transmission time by setting higher priorities for it in the intermediate nodes. Most of the time this leads to better response times and higher bandwidth utilization. Technically, PIP offers guarantees for one-way-delay (OWD), internet packet delay variation (IPDV), packet loss and capacity.

The drawback of PIP is the complexity to establish SLAs since it is a multi-step procedure with many of the steps requiring manual intervention. In EGEE, applications that may wish to utilize it should have to go through ENOC (EGEE network operations center). ENOC is responsible for handling GÉANT SLAs on behalf of EGEE and the participating user communities. Moreover, after a PIP is established there are still difficulties in monitoring potential network problems.

In practice, experiments conducted within EGEE in collaboration with the GridCC project [13], indicated that PIP currently provides little advantages compared to best-effort IP services (the default offered by GÉANT). This can be primarily attributed to network under-utilization rendering best-effort services sufficient for most applications. Nevertheless, this may change in the future as the number of applications running on the grid increases and the network bandwidth begins to saturate. Moreover, we are still missing some good application examples with high interactivity requirements that will exploit the full potentials of PIP. In the future we plan to invite such user communities to pursue the establishment of SLAs and experiment with PIP.

5 Example Applications

5.1 Fusion

The fusion scientific community was one of the first to develop interactive applications on the grid. One of the most successful is the Interactive Visualizer of an Integrator of Stochastic Differential Equations in Plasma (IVISDEP) [3]. The application simulates a great number of particles whose positions are calculated among a large number of nodes in the grid. IVISDEP is a parallel (MPI) application which has been ported to the grid both in the context of EGEE and int.eu.grid projects. Interactivity is required to steer the result visualization by dynamically altering the input parameters of the simulation. To achieve it, the application is based on the support tools approach and more specifically on *glogin* applying a master/worker model for computation.

5.2 Bioinformatics

In the Bioinformatics field a typical example of interactive application is gPTM3D [11]. gPTM3D is a grid-enabled medical image analysis application based on PTM3D software. It implements interactive segmentation and analysis of

radiological data. The application targets the remote execution of time-consuming parts of the software suite volume reconstruction of large or complex organs. On the user side it exposes a complex interface featuring optimized graphics and medically oriented interactions. When in operation, gPTM3D generates a large number of short-lived jobs with an average lifetime of 2 min. gPTM3D follows the application agent approach to ensure real-time medical image analysis/rendering. It has also been used as a showcase for demonstrating SDJ-capable CEs.

5.3 High Energy Physics

In most cases, HEP applications require little interactivity or real-time responsiveness. Nevertheless, many of the large experiments in CERN have been the driving force behind many of the technologies that facilitate quick system response. The reason for this is that many of the HEP applications require the submission and execution of a huge number of jobs. For such cases, VO-level scheduling mechanisms are the best approach to achieve maximum job execution throughput. For example, ATLAS [2], LHCb [17], and ALICE [1] experiments are following the pilot jobs approach. ATLAS in particular exploits the DIANE/GANGA [15] framework applying a master/slave architecture for data analysis. Also in CMS [4] a typical use case is the batch submission of 1,000+ short jobs. Some of these jobs might fail but the primary concern of the system is to sustain a high number of submissions. Some of them will be failed jobs that are being re-submitted, others will be a new batch of jobs for data analysis. To improve the support from gLite on the above use cases, CMS has worked closely with the SDJ WG in defining requirements for the SDJ specification.

5.4 Astronomy

Astronomy is one of the scientific fields with strong interest in interactivity and remote instrumentation. Although no application from this discipline has reached production phase yet, it is interesting to examine a typical use case which pushes for the improvement of real-time interactivity support from the grid infrastructure.

One of the main activities astronomers carry out is the submission of observing runs to observing facilities. Observing facilities are almost always made of a ground-based or space telescope and a pool of instruments interfaced to it. Remote observing applications involve quite complex workflows that include tasks such as

- Checking the status of the telescope and of the related instrumentation (telemetry parameters retrieval).
- Changing the status of the telescope and of the related instrumentation by modifying some of these parameters.
- Driving an observing run that is made of subsequent pointing and tracking periods.

- Monitoring one-shot or continuous observations.
- Downloading a bunch of data at the time of their acquisition (optical images, spectra, etc.).

Short time for job scheduling and dispatching as well as high-throughput job execution is essential for the successful porting of such applications on the grid. Moreover, since these jobs have to be combined in complex workflows, workflow engines and respective development environments should also be able to exploit one of the described interactivity technologies.

The astronomy and astrophysics cluster of EGEE are currently experimenting with the various supported technologies and will be porting such applications during the third phase of the project.

6 Conclusions and Related Work

Interactive and real-time applications remain a challenge for modern grid infrastructures. Although significant progress has been made in the last years to support them, still there remain open issues and impediments that prohibit the full exploitation of the grid from such applications. In EGEE these impediments are being alleviated by improving the functionality offered by the middleware and by the site configurations. User level techniques and tools have also considerably contributed to improve the situation. The network upon which grid infrastructures are deployed also plays a significant role in enabling fast interactions by using advanced QoS. Nevertheless, the potentials of this still remain unexploited by most of the application communities who rely on best-effort services offered by the network infrastructure.

Despite the above restrictions, many interactive applications have successfully been demonstrated on the EGEE grid. Apart from those that require direct interactivity (e.g., Fusion, Biomedicine, and Astronomy) many other classes of applications drive the evolution of middleware and high-level services characteristics that are also useful for interactive applications (e.g., fast job scheduling, increased job throughput, and workflow engines).

Apart from EGEE, many other European projects have been contracted or still are underway that focus on grid interactivity, real-time applications, and remote instrumentation. We already mentioned int.eu.grid, which has put interactivity as a main goal of the project. In the area of remote instrumentation considerable work has been performed in the context of GridCC and RINGrid [19] projects. DORII [7] builds upon the findings of the previous ones, expanding their applicability to earthquake, experimental science, and environmental communities. Also, CYCLOPS [5] focuses on civil protection applications running on the grid, some of them exposing high interactivity requirements. All of the above projects exploit EGEE's basic middleware to build higher level services enabling domain-specific or generic interactivity features.

EGEE is already within in its third phase of execution ending in April 2010. During this time we will continue collaboration with research projects and scientific

communities with strong interest in interactivity, real-time applications, and remote instrumentation, making sure that EGEE's grid infrastructure provides an ideal platform for deploying and running this class of applications.

Acknowledgments The authors would like to thank Cécile Germain-Renaud, Paco Castejón. Rubén Vallés Péres, Francesco Giacomini, and Claudio Vuerli for providing invaluable input for this chapter. This work is co-funded by the European Commission through the EGEE-III project and contract number INFSO-RI-222667.

References

1. ALICE Experiment Portal. http://aliceinfo.cern.ch
2. ATLAS Experiment Home Page. http://atlas.web.cern.ch
3. I. Campos, F. Castejón, G. Losilla, J.M. Reynolds, F. Serrano, A. Tarancón, R. Vallés, and J.L. Velasco. IVISDEP: A Fusion Plasma Application Ported to the Interactive European Grid e-Infrastructure. In *Spanish Conference on E-Science Grid Computing*, Madrid, Spain, 2007.
4. CMS Experiment Home Page. http://cms.cern.ch
5. CYCLOPS Project Home Page. http://www.cyclops-project.eu/
6. DIRAC: Distributed Infrastructure with Remote Agent Control. Project Web Page, http://lhcb-comp.web.cern.ch/lhcb-comp/Dirac/default.htm
7. DORII: Deployment of Remote Instrumentation Infrastructures. http://www.dorii.eu
8. EGEE Project Web Site. http://www.eu-egee.org
9. G'EANT Web Site. http://www.geant.net
10. C. Germain-Renaud, C. Loomis, C. Blanchet, C. Grandi, D. Kranzlmüller, D. Liko, E. Laure, F. Prelz, F. Hemmer, I. Blanquer, J. Moscicki, J. Montagnat, M. Lamanna, F. Pacini, and R. Texier. Short Deadline Jobs, Working Group – First Report. Technical Report, EGEE, March 2006. http://egee-intranet.web.cern.ch/egee-intranet/NA1/TCG/wgs/sdj.htm
11. C. Germain-Renaud, R. Texier, A. Osorio, and C. Loomis. Grid scheduling for interactive analysis. In *Challenges and Opportunities of HealthGrids: Proceedings of Healthgrid*, pp. 22–33, Geneva, 2006.
12. gLite: Lightwight Middleware for Grid Computing. http://glite.org
13. GridCC Project Web Site. http://www.gridcc.org
14. D. Groep, O. Koeroo, and G. Venekamp. gLExec: gluing grid computing to the Unix world. In *Computing in High Energy and Nuclear Physics (CHEP)*, Victoria, British Columbia, Canada, Sep. 2007.
15. K. Harrison, C.L. Tan, D. Liko, A. Maier, J. Moscicki, U. Egede, R.W.L. Jones, A. Soroko, and G.N. Patrick. Ganga: A Grid user interface for distributed analysis. In *Proceedings of the Fifth UK e-Science All-Hands Meeting*, Nottingham, UK, Sep. 2006.
16. Interactive European Grid Project Web Site. http://www.interactive-grid.eu
17. LHCb Experiment Home Page. http://lhcb.web.cern.ch/lhcb/
18. J.T. Moscicki. DIANE – Distributed analysis environment for GRID-enabled simulation and analysis of physics data. In *NSS IEEE 2004*, Portland, Oregon, USA, Oct. 2003.
19. RINGrid Project Home Page. http://www.ringrid.eu
20. H. Rosmanith and D. Kranzlmüller. glogin – A multifunctional, interactive tunnel into the grid. In *Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pp. 266–272, Washington, DC, Sep. 2004.

Part IV
Supporting Services

Chapter 23
g-Eclipse – A Middleware-Independent Framework for Accessing Existing Grid Infrastructures

M. Stümpert, H. Kornmayer, and M. Knauer

Abstract Even as grid computing became more and more popular in the past few years, it is still not fully established in the scientific world as an alternative to high performance or cluster computing. One of the reasons is the inherent complexity of grids that makes their usage far too difficult for non-experts. Therefore new concepts are needed to lower the threshold of entering grid computing for new users in order to open the grid world to a broader community.

The g-Eclipse project tries to hide the complexity of grids on the client side. It supports the user with his daily work on the grid and avoids common pitfalls that users are usually confronted with. In that sense g-Eclipse may be seen as a grid browser in comparison to an Internet browser that hides the complexity of the Internet from the user.

Keywords Grid · Eclipse · g-Eclipse · Grid abstraction · Grid model · User interface · UI · GUI · Middleware independent

1 Introduction

Scientific computing on distributed and heterogeneous infrastructures – called grid computing – becomes more and more popular. A multitude of national (e.g. [5]) and international (e.g. [7]) organisations established large-scale computing infrastructures over the past few years. These infrastructures are now available to scientists all over the world in order to allow them to migrate their daily work from their personal computers or local computing clusters to the grid. Besides the submission and management of computational jobs and the access to distributed data, the tasks of today's grids extend to fields like experiment control and/or maintenance, data acquisition, and the distributed large-scale processing of the acquired raw data.

M. Stümpert (✉)
Institute for Scientific Computing, Forschungszentrum Karlsruhe, Germany
e-mail: mathias.stuempert@iwr.fzk.de

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_23,

In a modern grid infrastructure a user can play a diversity of roles. Additionally a variety of grid middlewares exist. As a result the burden of entering grid computing is often too high for potential grid users like, for instance PhD students. Therefore strategies are needed to greatly simplify the access to grids comprised of different infrastructures and middlewares and to unify both the access and the way a user interacts with the available resources in respect to the different roles. One way of doing so is to push standardisation efforts like done, for instance, by the OGF[1] [19]. First successful results of these efforts include the development of JSDL[2] [8] and SAGA[3] [12].

The g-Eclipse framework presents another way of dealing with these heterogeneities. Instead of trying to force middleware developers to follow standards and therefore depending on their good will, g-Eclipse builds up a middleware-independent API for accessing grids. This API is mainly comprised of an abstraction layer that aggregates the functionalities of modern middleware technologies and a graphical user interface based on this abstraction layer. Concrete implementations of the abstraction layer are brought to the platform in the form of so-called plug-ins that are provided by the Eclipse framework [6]. This approach ensures to a great extent a common graphical interface for all supported middlewares and the underlying infrastructures. Currently the g-Eclipse team is proving its middleware independence with the implementation of the second middleware (GRIA [11]) after the implementation of the first middleware (gLite [10]) has finished with the end of 2007.

2 Grid Roles and Contexts

By definition a grid is a compound of distributed computing and storage resources [14]. These resources can be comprised of an intermixture of different hardware and software components. The controlling operating system may also differ from resource to resource. Therefore common interfaces and protocols are needed to connect such heterogeneous systems. The key players in these interconnection efforts are the different middleware systems that are tailored to interface these resources. A middleware can therefore be seen as gateway of a resource to the outside world and especially to other resources equipped with the same middleware. With the help of dedicated user interfaces it is afterwards possible to access the combined power of all these resources. It is worth mentioning that currently work is ongoing to at least loosely connect resources from different middleware domains (e.g. [16] or [9]) in order to also allow interoperability between these resources.

Generally all currently available middlewares can be subdivided into two categories:

Batch-like systems: A popular representative of this group is gLite. The user submits a job with a dedicated user interface to a service provided by the middleware – in the case of gLite the WMS.[4] Afterwards the middleware handles this job as a batch job.

Service-based systems: The user interacts with services – most often web services – to access the underlying grid infrastructures. One example of such a service-based middleware is GRIA.

The general tendency here is towards the service-based systems.

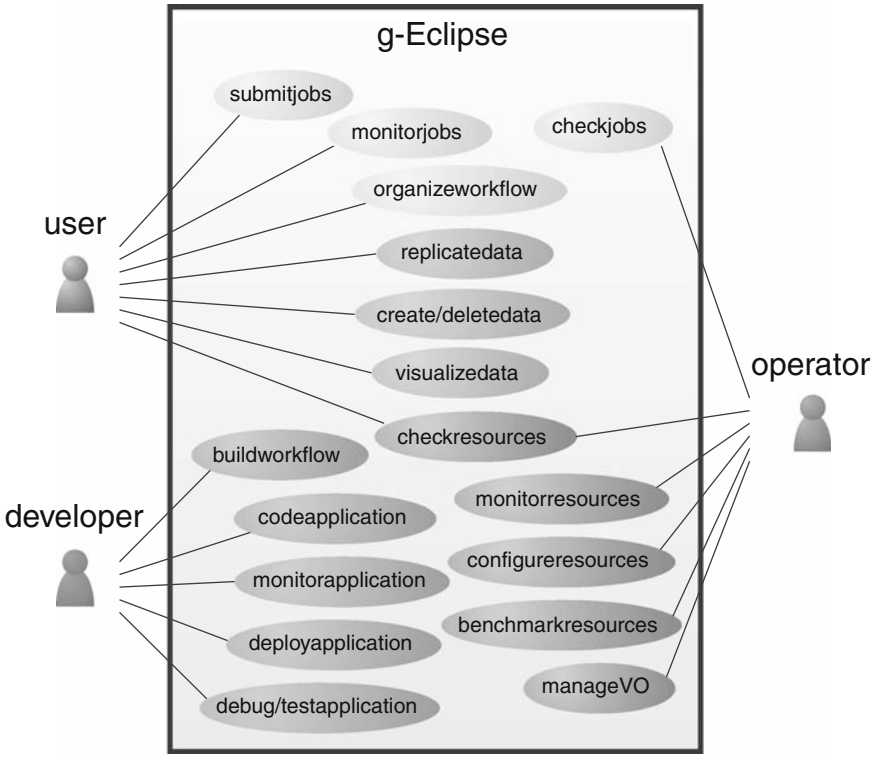


Fig. 23.1 The main use cases for grid users, operators, and application developers that are covered by the g-Eclipse framework

Additionally to different middleware systems, grid users are often assigned to different roles within a grid (see Fig. 23.1). That is, submitting a simple job is a completely different task than developing applications for a grid. Nowadays a variety of tools exist that cover either one or the other role. g-Eclipse aims for an integrated workbench and therefore supports the following three user roles:

Grid application users: Start their applications on the grid and need tools to support job submission and job status monitoring. Furthermore they must be able to manage their distributed data on the grid and to visualise them on remote or local resources.

Grid application developers: Need tools for the development of applications and for their deployment on the distributed grid infrastructure. The application

developers should therefore not be limited to a dedicated programming language or a dedicated programming environment, but may profit from possibilities to remotely compile or debug their applications.

Grid operators: Operate and maintain resources and middleware systems and components. They would like to access these resources in order to configure them or test them for availability and reliability.

Within g-Eclipse these different roles are supported by the perspective concept of the Eclipse platform that will be explained in more detail in the next section.

3 Benefitting from the Eclipse Framework

The g-Eclipse consortium decided to make use of the reliable Eclipse platform in order to build a middleware-independent framework for accessing the grid. Eclipse is an open and extensible platform for building integrated tools on top of it managed by the Eclipse Foundation. To sustain the collaboration with the Eclipse Foundation, g-Eclipse is an official Eclipse project since October 2006. The main features of the Eclipse framework from which g-Eclipse benefits are

Reliability: The open source eclipse framework evolves continuously driven by a variety of projects. The quality of the framework is assured by the use of the JUnit [18] testing framework. All Eclipse projects – including g-Eclipse – have to meet common quality criteria concerning issues like coding style and IPR.

Usability: Modern software comes with a bunch of standard functionalities like message and error logging or GUI-customisation. Eclipse already comes with all these standard features and a lot of features that go well beyond the standards.

Extensibility: In the Eclipse world, every plug-in amends the functionality of other plug-ins. This is achieved by the underlying OSGi [2] framework which manages the runtime dependencies between the plug-ins. In addition the Eclipse framework relies on the mechanism of extension points and extensions. An extension point is a definition of how to enhance existing functionality.

OS-Independency: Eclipse itself is known to run on several operating systems (Windows, Linux, Mac OS X, etc.). The g-Eclipse framework develops platform-independent extensions and therefore runs on any platform which is supported by Eclipse.

Contextualisation: Besides other components the graphical front end of g-Eclipse consists of menus, toolbars, and so-called views. The arrangement of these components and the composition of context sensitive actions within these elements is called a perspective. These perspectives are the base of the adaption of g-Eclipse to the different roles a user may play in the grid as described in Section 2.

By using the Eclipse framework, the g-Eclipse team gets these features for free and can focus on the development of a user-friendly client software for accessing grid infrastructures.

4 The g-Eclipse Architecture

The g-Eclipse architecture provides a middleware-independent model that aggregates the functionalities of today's middleware technologies. Upon this model common graphical user interfaces are built in order to access these grid functionalities in a user-friendly and middleware-independent way. In this section we will focus on the architecture while Section 5 will afterwards give an overview of the user interface. A detailed description of the g-Eclipse architecture can be found in [15].

4.1 The Grid Model

The grid model is subdivided into an abstraction layer and an implementation layer. The abstraction layer is a collection of Java interfaces that define the basic functionalities of the various model elements. The implementation layer itself is subdivided into two subgroups, public implementations that are visible to the outside world and internal implementations that are used by the core plug-in to set up the basic model structure. The public implementation layer contains all abstract classes that are available for developers in order to extend the model with middleware-specific functionality. Although it is possible to extend the functionality by implementing the interfaces of the abstraction layer directly, it is best practise to use the classes of the public implementation layer because basic mechanisms to manage the grid model are provided already. Nevertheless, the public definition of the private model parts within the abstraction layer is necessary in order to allow user interface interaction with these parts.

The abstraction of grid components is shown in Fig. 23.2 as a layered inheritance tree. As this layer only contains Java interfaces, multiple inheritance is allowed. The abstraction layer is used to map the structure of local and grid elements to the model and to also map the relations between the different elements. The base interface of this inheritance tree is the `IGridElement`. This class is the connector between resources in the Eclipse platform and grid resources.

The leaves of the model's inheritance tree are either internally implemented by the model's implementation layer (e.g. `IGridProject`) or thought to be implemented by concrete middleware implementations (e.g. `IGridComputing`).

Furthermore g-Eclipse provides implementations of emerging grid standards for a few of these interfaces. The `JSDLJobDescription`-class as part of the `eu.geclipse.jsdl`-plug-in, for instance, implements the `IGridJob Description`-interface. Therefore g-Eclipse comes with a fully featured JSDL implementation both on the model and on the UI side. Additionally the `eu.`

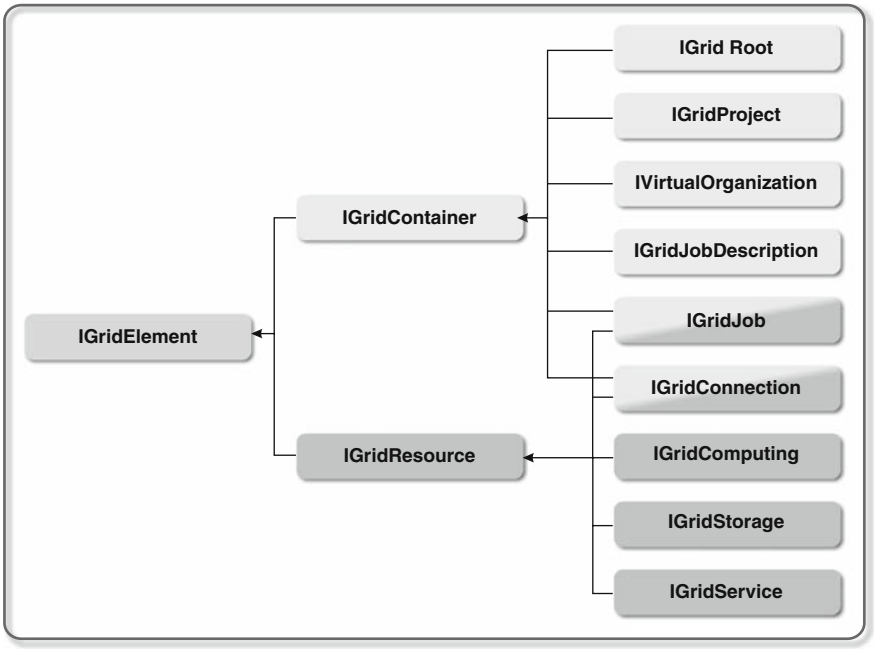


Fig. 23.2 Simplified outline of the grid model's abstraction layer showing the most important interfaces

`geclipse.info`-plug-in contains a fully featured implementation of the GLUE-schema [17]. This schema is used to represent infrastructure information within a grid. The info-plug-in furthermore provides classes for connecting services and computing and storage elements from the GLUE-schema with the grid model. Therefore the implementation of middlewares supporting the GLUE-schema is straightforward within g-Eclipse.

4.2 Job Management

One of the most important use cases for a grid client like g-Eclipse is the submission of jobs to a grid infrastructure and the status retrieval for these submitted jobs. The framework itself therefore comes with a job management infrastructure that is based on the `IGridJob`- and `IGridJobDescription`-interfaces. The following list depicts the corresponding workflow:

- The user creates a job description with the job description wizards.
- Optionally the user changes the job description with the corresponding editor.
- The job description is submitted to a grid infrastructure with the job submission wizard.

- As soon as the job description was successfully submitted the corresponding job appears in the g-Eclipse workspace, i.e. in the grid model and therefore also in the corresponding UI elements.
- The job status is updated at regular intervals giving the user continuous feedback about his submitted jobs.

The above-sketched functionalities are already integrated in the g-Eclipse abstraction layer. Middleware implementations are hooked in with the help of dedicated job submission and status services implementing `IGridService`. As long as the underlying middleware supports the submission of JSDL-based job descriptions no further actions are needed in order to open the g-Eclipse job infrastructure to a middleware. If JSDL is not supported a dedicated implementation of the `IGridJobDescription`-interface and a corresponding wizard and editor may be needed. Another option would be to provide a translator from JSDL to the middleware's native job description language. In this way the user may describe his jobs with the help of g-Eclipse's JSDL capabilities. The JSDL is then translated as integral part of the job submission process. An example of such an implementation is the gLite middleware adapter that provides an interface from JSDL to JDL – the native job description language of gLite.

4.3 Data Management

The access to remote file systems from within the g-Eclipse framework is managed with the `IGridConnection`-interface, which is implemented internally. This implementation uses EFS,[5] which is an abstraction for file systems within the Eclipse platform. With this implementation the g-Eclipse UI elements can directly interact with any implementation of EFS including file system actions like creating, copying, and deleting files or folders on remote sites. Therefore the framework does not distinguish between local and remote connections. Besides the inherent delay of remote connections the user should experience no differences between the different EFS implementations.

The g-Eclipse framework itself comes with four EFS implementations for remote grid file systems, i.e. GridFTP [1], SRM [21] LFC [4], and Amazon's S3 [3]. The first one is mainly supported by the globus middleware [13] while SRM and LFC are dedicated to the gLite middleware. Finally S3 is part of Amazon's Web services and represents the first step for introducing cloud computing to g-Eclipse.

4.4 Grid Project

The arrangement of available components (i.e. grid services, computing elements, storage elements, local files, and directories) in a project structure is another liability of the grid model. A user of g-Eclipse is normally working with so-called grid projects. Figure 23.3 shows an example of such a grid project. The underlying structure of such a project is built from the components of the grid model.

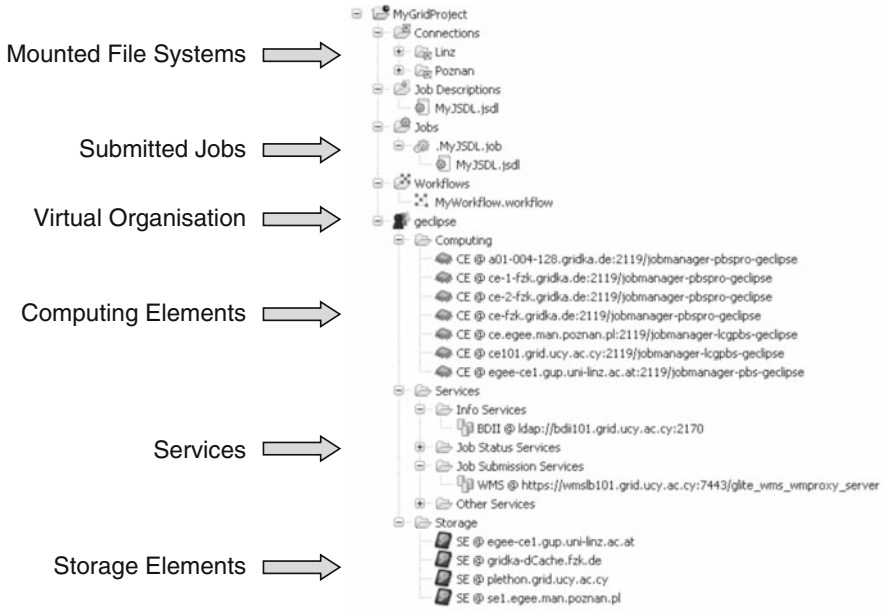


Fig. 23.3 Screenshot of the grid project view showing an exemplary gLite project for the geclipse-VO

The nature of a grid project is identified by its associated VO.[6] A VO in the sense of grid computing is a group a people – mostly distributed all over the world and joining an interest – that has managed access to resources on a dedicated grid infrastructure. A growing number of middlewares supports the VO concept in one or the other way. Nevertheless within the g-Eclipse framework the VO concept is a bit different from the aforementioned. It is the central access point for a user to an infrastructure and helps g-Eclipse to build up the user's personalised grid within a grid project. Therefore g-Eclipse uses the VO to query a provided information endpoint for available resources that may be accessed by the user. Figure 23.3 shows a subset of these available resources for the geclipse-VO on a gLite-based infrastructure under the VO item called "geclipse".

4.5 Authentication and Authorisation

Since grid computing is about sharing resources among a large community, authentication and authorisation are important topics. The g-Eclipse framework comes with a fully featured authentication and authorisation infrastructure (AAI) that is based on a separate abstraction layer not being part of the grid model. The base interfaces of this infrastructure are called `IAuthenticationTokenDescription` – used to describe, query, and create authentication tokens – and `IAuthentication`

`Token` representing the token itself. Such tokens can be of any type. Examples range from simple passwords to X.509-based certificates with implementation-specific extensions.

Whenever an operation within the g-Eclipse framework needs to authenticate the user to the underlying grid infrastructures the AAI is asked to provide an appropriate authentication token (see Fig. 23.4). Therefore the corresponding operation prepares a so-called token request specifying the type of token – and optionally its parameters – that is needed in order to successfully finish the operation. The AAI looks up the pool of already existing tokens and returns a token that matches the requirements. If no such token could be found the user is asked to create a new token. When creating such a new token the token parameters of the request are fed into the tokens creation process. Therefore a minimum of user interaction is needed in order to create authentication tokens required for dedicated operations on the grid.

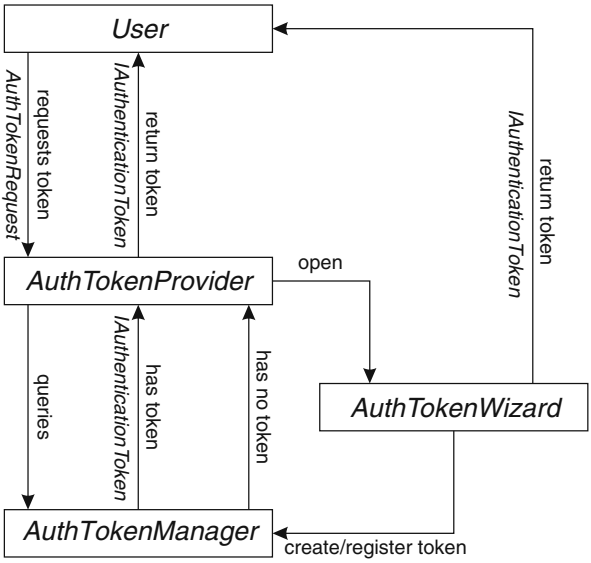


Fig. 23.4 The workflow for requesting an authentication token within the g-Eclipse authentication and authorisation infrastructure

4.6 Middleware Implementations

With the described architecture, the g-Eclipse team is able to provide a general middleware-independent framework and on top of it two middleware implementations – one for the gLite middleware and a second one for the GRIA middleware. Table 23.1 compares the current status of both implementations. In the following we will give a rough overview of each implementation.

gLite The g-Eclipse team developed the gLite middleware binding in parallel to the grid model in the first project year. This development helped enormously to improve the model in the context of a batch system-based middleware. The implementation of the gLite binding is complete in the sense that all daily tasks of a gLite user can be performed using g-Eclipse. These tasks contain data and job management, information retrieval, application development, site administration, and data visualisation.

GRIA In order to prove the concept of middleware independency the g-Eclipse team started with the development of a second middleware implementation in October 2007. As a service-based middleware GRIA is in some points completely different from gLite and therefore is an excellent candidate for testing the middleware independency of the grid model. Up to now the model had not to be changed in order to integrate GRIA as second middleware. This can be seen as a successful proof of concept for the grid model.

Table 23.1 Comparison of the main g-Eclipse features for the gLite and the GRIA middleware implementations

	gLite	GRIA
Virtual organisation	VOMS-VOs	GRIA
Authentication/authorisation	VOMS-Proxies	Key stores
Information system	BDII	Information registry
Job submission	WMS-based	Job service-based
Job status	L& B-based	Job service-based
Data access	GridFTP, SRM, LFC	Data stager-based

5 A Common User Interface for Different Grid Middlewares

The user interface plays a major role within the g-Eclipse framework. Its main task is to provide a set of common control elements for all types of middlewares. Therefore, the user interface may only depend on the abstraction layer of the grid model but not on any concrete implementation. In this way, it is guaranteed that if a middleware implements the abstraction layer in a proper way, it will automatically be integrated in the g-Eclipse user interface.

5.1 Grid Model Views

Grid model views are the central UI elements for accessing the g-Eclipse core functionalities. A view itself is a rectangular area within the Eclipse workbench that provides viewing elements to present data and information to the user and control elements such as buttons and context menus to allow the user to manipulate that data (like copying and deleting). All grid model views represent certain portions of the grid model and provide functionality to manipulate the model. This implies that these views share a lot of functionalities. Therefore the grid model views are all based on a common subclass.

The most important model view is the grid project view (see Fig. 23.3). It gives the user access to his projects and provides actions to interact with the grid. Submission of jobs, data management, and site administration are such actions that are available within the project view. Other views are the connection view for managing all connections that are defined within the user's workspace or the job view that contains all submitted jobs and gives the user feedback about their current status. Both the project view and the job view can be seen in Fig. 23.5 on the top left and on the bottom, respectively.

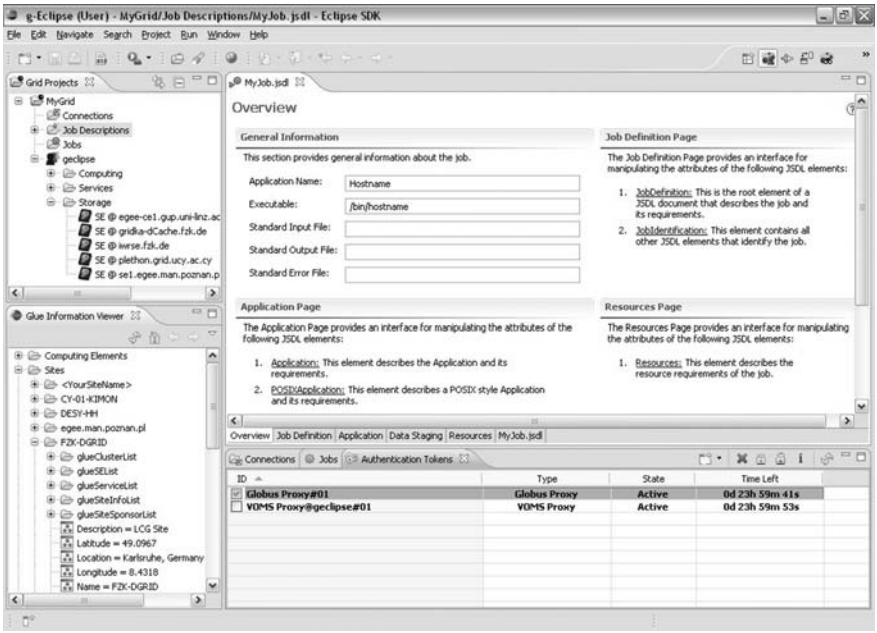


Fig. 23.5 Screenshot of the g-Eclipse user perspective showing an example for a grid project. In the center of the perspective one can see the JSDL multipage editor

5.2 Perspectives

A perspective in the Eclipse sense is a collection of views and other UI elements that serve a dedicated purpose. The g-Eclipse framework defines four perspectives for accessing the grid, i.e. the user perspective, the operator perspective, the developer perspective, and the grid exploring perspective.

Figure 23.5 shows a screenshot of the user perspective. This perspective collects all views for accessing grids in order to submit jobs or to manage data. It is mainly comprised of the grid project view and the GLUE Info View on the left, the job view, the connection view, and the authentication token view on the bottom and the editor area centered. The picture shows the JSDL multipage editor in the editor area

that provides an easy way to edit JSDL files. Nevertheless this area is not limited to the JSDL editor, but may contain any type of editor depending on the type of the file being edited. Other editors that are included in g-Eclipse are a workflow editor or various site configuration editors.

The operator perspective is designed to support site administrators (see Fig. 23.6). It collects all views for accessing grid infrastructures in order to manage job queues or to configure sites. For experienced users it furthermore contains a SSH and a gLogin [20] console where the operator is able to directly login to a remote machine in order to administrate this machine.

The developer perspective is dedicated to grid application developers. It assists developers in bringing their applications to the grid. It offers, for instance, possibilities to compile and debug applications remotely. Therefore a developer writes his source code on his local machine using Eclipse with one of the available development environments (e.g. JDT for Java development or CDT for C/C++ development). Afterwards he uses the developer perspective's functionality to transfer the source code to a grid node and to compile or to debug it there. From the developer's point of view there is no difference in compiling/debugging locally or remotely

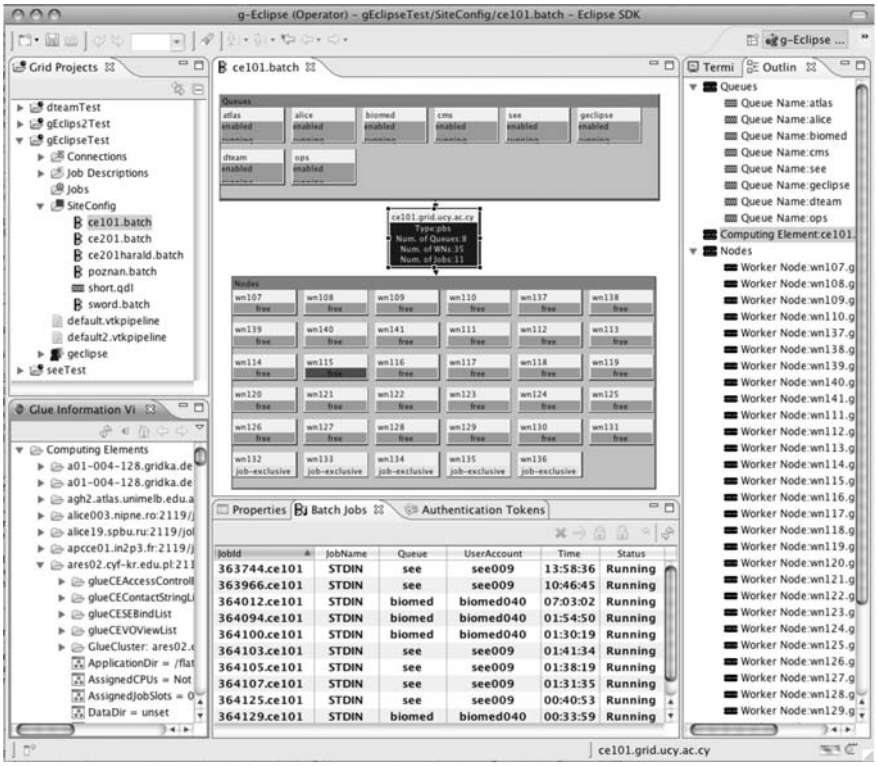


Fig. 23.6 Screenshot of the g-Eclipse operator perspective showing a computing element with its associated queues and worker nodes

since g-Eclipse actually hides the complexity of the remote parts. Another purpose of the developer perspective is to assist the developer with deploying his application on a grid infrastructure, i.e. installing the application in a way that seems to be appropriate for the underlying middleware and the corresponding infrastructure.

Finally, the grid exploring perspective presents two connection views side by side in order to give the user the possibility to easily transfer data in the grid, even among different middlewares. Within this perspective it is also very easy to open files remotely, edit them in the editor area with an appropriate editor, and directly save them back to the remote location without creating a local copy of the file.

5.3 Wizards and Dialogs

The g-Eclipse framework makes extensive use of the Eclipse concept of wizards to support users and to hide the complexity of initialising and managing grid resources. A wizard is a context sensitive set of UI dialogs that guides the user through a dedicated process. The g-Eclipse user interface makes use of wizards wherever possible in order to enable inexperienced users to start up their daily work.

The JSDL wizard (Fig. 23.7 on the left) creates an initial version of a JSDL file – which can be very complex – from a minimal set of input parameters like the input or output files. This file can afterwards be edited with the JSDL editor that provides much more and enhanced possibilities to edit JSDL files. Other wizards of the g-Eclipse framework make the access to grid infrastructures as simple as possible for new users. There are, for instance, wizards to create new VOs or to import existing VOs. The CA certificate wizard supports the import of CA certificates, the workflow wizard enables the creation of new workflows, the connection wizard helps to create a new file system mount, while the authentication token wizard creates authentication tokens for accessing an infrastructure.

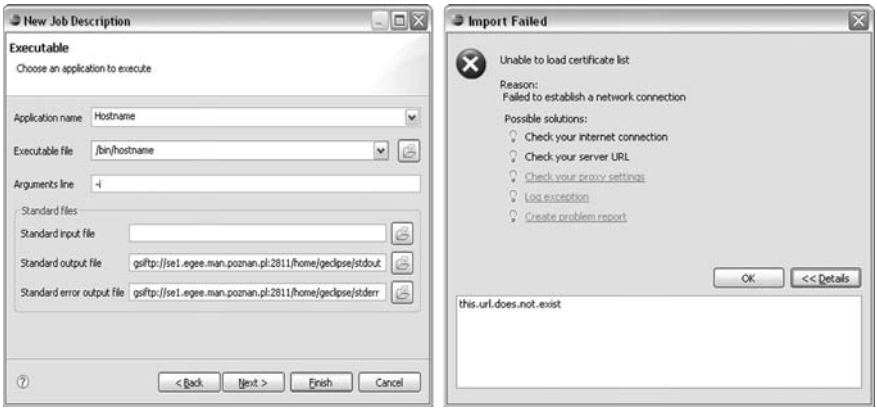


Fig. 23.7 Screenshots of the JSDL Wizard on the left-hand side and the problem dialog on the right-hand side

The g-Eclipse framework supports the troubleshooting of errors and problems, which occur when accessing grid infrastructures. g-Eclipse provides a dedicated problem reporting mechanism, which does not only report about problems that may have occurred during an operation on the grid but also provides possible solutions to a problem. These solutions may either be passive and therefore only be represented to the user by a descriptive text or they are active and may trigger an action that assists the user in solving the problem. The graphical front end of this mechanism is the problem dialog (Fig. 23.7 on the right) that lists the problem and its possible reasons and solutions. Active solutions are presented as hyperlinks that – when activated – trigger the action that may solve or may at least assist the user to solve the problem.

6 Conclusions

Today's grid infrastructures suffer from the complexity of the underlying middlewares. Accessing these infrastructures is therefore a difficult task for non-experts. The g-Eclipse project tries to circumnavigate these difficulties on the client side by providing a simple GUI-driven framework that supports grid users, operators, and developers with their daily work on the grid. Therefore g-Eclipse is based on the Eclipse platform that provides not only a complete workbench environment but also a lot of advanced programming concepts that fit well with the needs of a middleware independent and extensible framework like g-Eclipse.

The heart of the g-Eclipse framework is its middleware-independent grid model. It mainly is comprised of an abstraction layer that defines access points to functionalities of an idealised grid middleware. Middleware-specific implementations of at least parts of this model bring the real-world grid computing functionalities into g-Eclipse. The g-Eclipse consortium itself currently provides implementations for two different middlewares, gLite and GRIA. Both middlewares are based on totally different concepts but the g-Eclipse client parts are both based on the grid model, which may be seen as the proof of concept for the middleware-independent model.

The user interface of g-Eclipse is completely based on the grid model. This means that all available and upcoming middleware implementations will have the same face within g-Eclipse. Therefore it makes only little difference for a user if he works with the one or the other middleware. This helps enormously to lower the threshold for new users. Furthermore the user interface makes heavy use of well-known user-friendly Eclipse UI concepts like views, perspectives, and wizards in order to further simplify the access to grids. Especially for g-Eclipse a new problem reporting mechanism was developed that allows developers to give the user solutions by the hand for problems that occurred during runtime.

The g-Eclipse platform is ready to be used for gLite-based infrastructures. Daily tasks like job submission, data management, site configuration, and grid application development are defined by the middleware-independent model and are implemented either as general functionality on the core side or as gLite-specific

functionality on the middleware side of g-Eclipse. Some first parts of these functionalities are furthermore already available for the second middleware, i.e. GRIA.

In the near future the core functionalities will be extended by higher level features like support for workflows and parametric jobs. Attempts will be made to integrate these features seamlessly into already existing functionalities. Parametric jobs will, for instance, be handled by an extended JSDL specification. The middleware-dependent part will then reduce to the translation of these extended JSDLs to the middleware's own job description language. Further improvements of the core functionalities contain support for VO administrators in the operator perspective and application deployment in the developer perspective.

All these functionalities will be made available for the already supported middlewares, i.e. gLite and GRIA. Furthermore the still missing functionalities for the GRIA middleware will also be provided in the near future. Other plans contain support for further authentication token types – e.g. Shibboleth or MyProxy – or further visualisation systems besides VTK.

At the current phase g-Eclipse has proven its middleware independence. The next step is to test its architecture against a non-middleware concept. Therefore an adapter for Amazon's Web services will be provided in the next few months. As a first step in this direction the project already provides an implementation for the Amazon Simple Storage Service (S3) as of Milestone Release 1.0.0 M3. With the further integration of Amazon's Elastic Compute Cloud (EC2) g-Eclipse will not only be independent of the underlying middleware but also of the underlying grid concept.

Acknowledgments The g-Eclipse project is funded by the European Commission's 6th Framework Programme (#FP6-2005-IST-034327). The authors would like to thank the member institutions of the g-Eclipse consortium and all the project members. Furthermore g-Eclipse is supported by the Eclipse Foundation as an Eclipse Technology project.

Notes

1. *O*pen *G*rid *F*orum
2. *J*ob *S*ubmission *D*escription *L*anguage
3. *S*imple *API* for *G*rid *A*pplications
4. *W*orkload *M*anagement *S*ystem
5. *E*clipse *F*ile *S*ystem
6. *V*irtual *O*rganisation

References

1. W. Allcock. *GridFTP Protocol Specification*. http://www.globus.org/alliance/publications/papers/GFD-R.0201.pdf
2. Osgi Alliance. *Osgi Service PlatformRelease 3*. IOS Press Inc., 2003. ISBN1586033115.
3. Amazon S3 – Amazon Simple Storage Service. http://aws.amazon.com/s3

4. J.-P. Baud and J. Casey. *Evolution of LCG-2 Data Management*. http://indico.cern.ch/contributionDisplay.py?contribId=278\&sessionId=7\&confId=0
5. D-Grid – Die Deutsche Grid-Initiative. http://www.d-grid.de
6. Eclipse – An Open Development Platform. http://www.eclipse.org
7. EGEE – Enabling Grids for E-sciencE. http://www.eu-egee.org
8. A. Anjomshoaa et al. *Job Submission Description Language (JSDL) Specification Version 1.0*. http://www.gridforum.org/documents/GFD.56.pdf
9. B. Tierney et al. *A Grid Monitoring Architecture*. http://www.ggf.org/documents/GFD.7.pdf
10. E. Laure et al. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45, 2006.
11. M. Surridge et al. Experiences with GRIA – Industrial applications on a Web Services Grid. In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pp. 98–105, 2005.
12. T. Goodale et al. *A Simple API for Grid Applications (SAGA) Version 1.0*. http://www.ogf.org/documents/GFD.90.pdf
13. I. Foster. *Globus Toolkit Version 4: Software for Service-Oriented Systems*, vol. LNCS 3779, pp. 2–13, Springer-Verlag, 2006.
14. I. Foster and C. Kesselman. *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufmann Technical Report, 2 edition, 2003. ISBN 978-1-55860-933-4.
15. The g-Eclipse consortium. *g-Eclipse Final Architecture*. Project deliverable D1.8 http://www.geclipse.eu/fileadmin/Documents/Deliverables/D1.8.pdf
16. GIN – Grid Interoperation/Interoperability Now. http://wiki.nesc.ac.uk/read/gin-jobs?HomePage
17. GLUE – The Grid Laboratory Uniform Environment Schema. http://forge.ogf.org/sf/projects/glue-wg
18. JUnit – Resources for Test Driven Development. http://www.junit.org/
19. OGF – Open Grid Forum. http://www.gridforum.org
20. H. Rosmanith and J. Volkert. glogin – Interactive Connectivity for the Grid. In Z. Juhasz, P. Kacsuk, and D. Kranzlmüller, editors, *Distributed and Parallel Systems – Cluster and Grid Computing, Proceedings of DAPSYS 2004*, 5th *Austrian-Hungarian Workshop on Distributed and Parallel Systems*, pp. 3–11, Budapest, Hungary. Kluwer Academic Publishers, 2004.
21. A. Shoshani, A. Sim, and J. Gu. *Storage Resource Managers: Essential Components for the Grid*. http://sdm.lbl.gov/srm-wg/papers/SRM.book.chapter

Chapter 24
Semantics-Based Context-Aware Dynamic Service Composition

K. Fujii and T. Suda

Abstract This chapter presents a semantics-based context-aware dynamic service composition framework that composes an application through combining distributed components based on the semantics and contexts of the components. The proposed framework consists of component service model with semantics (CoSMoS), component runtime environment (CoRE), and semantic graph-based service composition (SeGSeC). CoSMoS represents the semantics and contexts of components. Since users are also modeled as components in the proposed framework, the contexts of users are also represented in CoSMoS. CoRE is a middleware to support CoSMoS on various distributed computing technologies. SeGSeC is a mechanism to compose an application by synthesizing its workflow based on the semantics and contexts of components. By modeling and leveraging the semantics of components, the proposed framework is capable of composing applications requested in a natural language. The proposed framework also utilizes the contexts of components to compose context-aware applications. The proposed framework allows users to explicitly specify rules on which components they prefer in a specific context and also acquires user preferences in different contexts through learning. This chapter describes the design and mechanism of the proposed framework and also presents a case study and simulation experiments for the proposed framework.

Keywords Dynamic service composition · Semantics · Context-aware · Service-oriented framework

1 Introduction

Many software programs, devices, and resources are deployed today as distributed components. As a network becomes more ubiquitous, the number of such distributed components available on the network also grows rapidly. Once a large number of distributed components become available, it is possible to autonomously compose

K. Fujii (✉)
School of Information and Computer Science, University of California, Irvine, CA, USA
e-mail: kfujii@ics.uci.edu

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_24,

various applications through combining different sets of distributed components on demand (i.e., upon request by users). For example, by combining a microphone component, a voice recognition service component, and an instant messaging service component, it is possible to compose an application which first translates a user's voice to a text message and then sends it to another user who is using the instant messaging service. This concept of dynamically composing applications from distributed components on demand is called dynamic service composition [1].

Dynamic service composition provides flexible and adaptive applications by composing the applications according to user requests, user contexts (e.g., location, time, and profile), and user preferences. Dynamic service composition also reduces application development costs because a variety of new applications may be created by simply deploying a small number of new components.

Although several dynamic service composition systems have already been proposed and implemented (e.g., [4, 16, 12]), there are two major issues which remain unaddressed in the existing systems. First, the existing systems often require a user to request an application in a manner that may not be intuitive to the user. For instance, several systems (e.g., [4]) require a user to choose or create a template of the application, while others (e.g., [12]) require a user to specify the pre/post conditions of the application as logic formulas. Because it is not trivial to understand the syntax of application templates or logic formulas, end users may experience difficulties in requesting applications using existing systems. In order to provide higher usability to end users, a dynamic service composition system should allow users to request applications in an intuitive manner, for example, using a natural language. Second, as the number of components available in the network increases, it becomes more likely that different combinations of components may satisfy the same composition request. For instance, when multiple microphone components and voice recognition service components are available, any combination of the two provides an application to translate a user's voice into a text message, although the user may prefer the nearest microphone and the most popular voice recognition service. When different combinations of components satisfy the given composition request, a dynamic service composition system should rank the possible combinations for the user, similarly to modern web search engines, which rank search results based on certain criteria (e.g., the hyperlink structure).

In order to allow users to request applications in an intuitive manner and to rank possible combinations of components for users, the authors of this chapter propose a semantics-based context-aware dynamic service composition framework. In order to allow users to request applications in a natural language, the proposed framework models the semantics of components and composes applications based on the semantics of the components. For example, a user James may request an application to talk to his friend Tom by simply typing or verbally stating a sentence "I want to talk to Tom," and the proposed framework composes the voice-to-instant-message application described earlier to allow James to talk to Tom. In addition, in order to rank possible combinations of components for users, the proposed framework composes applications based on the context information of users (e.g., locations and profiles) and components (e.g., device capabilities). For instance, when James

requests an application to talk to Tom, the proposed framework may utilize location information to select the nearest audio device components around James to compose the voice-to-instant-message application described earlier. The proposed framework allows users to explicitly specify rules on which components they prefer in a specific context and also acquires user preferences in different contexts through learning.

The proposed semantics-based context-aware dynamic service composition framework is an extension of the framework which has been proposed by the authors of this paper [6, 5, 7]. The authors of this paper first introduced the concept of composing applications based on the semantics of components in [6, 5] and demonstrated through preliminary implementation that it is possible to compose applications requested in a natural language by utilizing the semantics of components. In [7], the authors of this paper have also shown that the framework is applicable in the Web Service domain.

This chapter describes the design and mechanism of the proposed framework, and also presents a case study and simulation experiments for the proposed framework. Section 2 describes the detailed design and mechanism of the proposed framework. Section 3 describes a case study of context-aware service composition based on the proposed framework. Section 4 presents simulation experiments that evaluate the performance of the proposed framework. Section 5 concludes this chapter.

2 Semantics-Based Context-Aware Dynamic Service Composition Framework

The proposed semantics-based context-aware dynamic service composition framework consists of component service model with semantics (CoSMoS), component runtime environment (CoRE), and semantic graph-based service composition (SeGSeC). CoSMoS is a component model to represent the functions, semantics, and contexts of a component as a semantic graph. Since users are also modeled as components in the proposed framework, the contexts of users are also represented in CoSMoS. CoRE is a middleware to support CoSMoS on various distributed computing technologies, such as Web Service technologies. SeGSeC is a semantics-based service composition mechanism that composes applications based on the semantics and contexts of components. SeGSeC implements both rule-based context-aware service composition where it composes context-aware applications based on user-specified rules, and also learning-based context-aware service composition where it composes context-aware applications based on the user preferences it acquired through learning.

2.1 Component Service Model with Semantics (CoSMoS)

Component service model with semantics (CoSMoS) is an abstract component model to represent the functions, semantics, and contexts of a component. CoSMoS also represents rules, which consist of conditions and consequences and are used for

the rule-based context-aware service composition in SeGSeC. CoSMoS represents the functions, semantics, contexts, and rules of a component as a semantic graph, which is a directed graph that consists of nodes and labeled links. Figure 24.1 shows the formal specification of CoSMoS as a UML class diagram, and Fig. 24.2 shows the example CoSMoS representations of a microphone component, a voice recognition service component, and a user component. In CoSMoS, nodes (e.g., "component microphone" in Fig. 24.2a) are instances of the classes (e.g., the "component" class) in Fig. 24.1. The labels on the associations in Fig. 24.1 (e.g., "implements") are the predefined concepts used to label a link to connect two nodes (e.g., the "implements" link between "component microphone" and "operation recordSound" in Fig. 24.2a). Since CoSMoS is an abstract model, it is possible to describe CoSMoS in different ways, for example, in RDF [7].

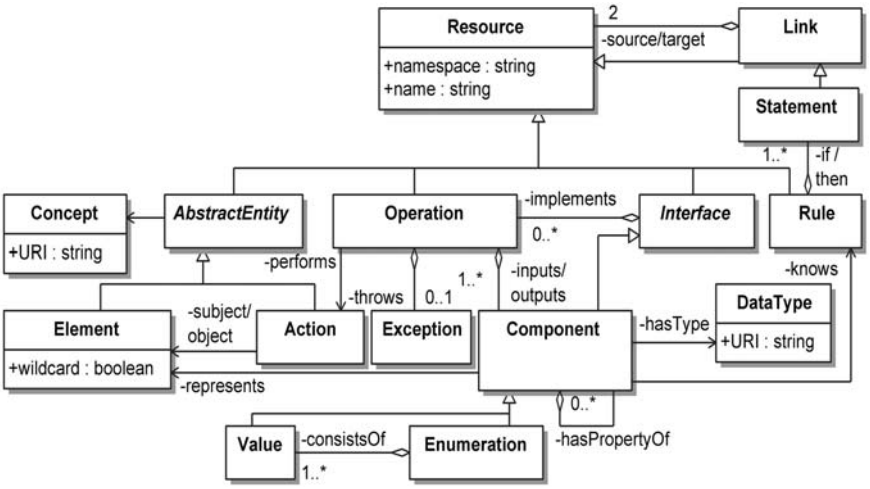


Fig. 24.1 CoSMoS class diagram

Representing functional information in CoSMoS. CoSMoS represents the functions of a component as a set of operations and properties. An operation is modeled as a set of inputs, outputs, and exceptions. Each input (and output) of an operation is modeled as a component, representing that the operation accepts (or generates) another component as its input (or output). For instance, Fig. 24.2a shows that "component microphone" implements "operation recordSound" which generates "component outSound". A property of a component is also modeled as a component, representing that the property can be retrieved as another component. When an input, output, or property of a component is binary data (e.g., integer value, string, byte array, or structured data), its data type is specified. For instance, Fig. 24.2a shows that the data type of "Component outSound" is "Binary audio/wav", i.e., audio data. If a component (e.g., a printer) accepts or generates a physical object (e.g., paper) instead of binary data, the input or the output of the component is represented as a component without any data type.

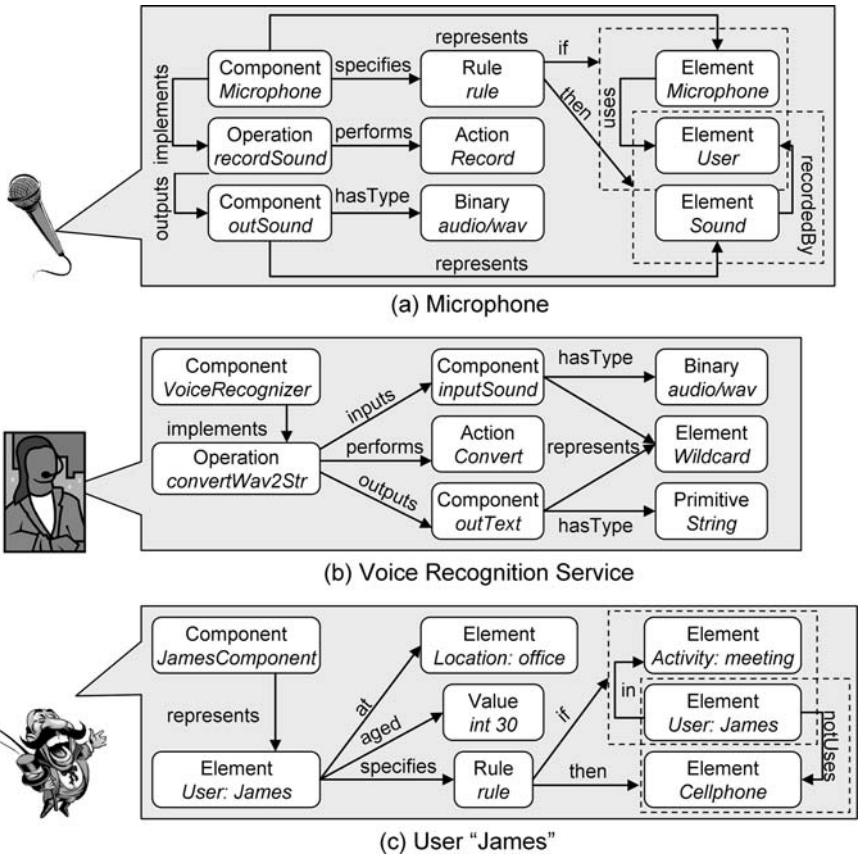


Fig. 24.2 Example CoSMoS representations

Representing semantic information in CoSMoS. CoSMoS represents the semantics of a component, i.e., what each operation, input, output, and property of the component semantically represents, based on the notation of Conceptual Graph (CG) [15]. In CG, each node and link in a graph is an instance of a concept, and each concept is defined by an ontology. For example, a node which represents a human is modeled as an instance of the concept "Human." A node may also have a name to distinguish it from other nodes. For example, two "Human" nodes can be named "Alice" and "Bob" to represent Alice and Bob. Two nodes (e.g., "Human: Alice" and "Human: Bob") may be connected by a link (e.g., "isSiblingOf") to represent their semantic relationship. An ontology defines a set of concepts (e.g., "Human", "Mammal" and "Animal") as well as their relationships (e.g., "Human is a kind of mammal, which is a kind of animal"). Domain experts or component designers design and deploy ontologies using existing ontology definition frameworks, such as RDF Schema and OWL. Figure 24.3 illustrates an example of an ontology which defines concepts about animals, as well as a CG which represents that "Human: Alice" is a sibling of "Human: Bob".

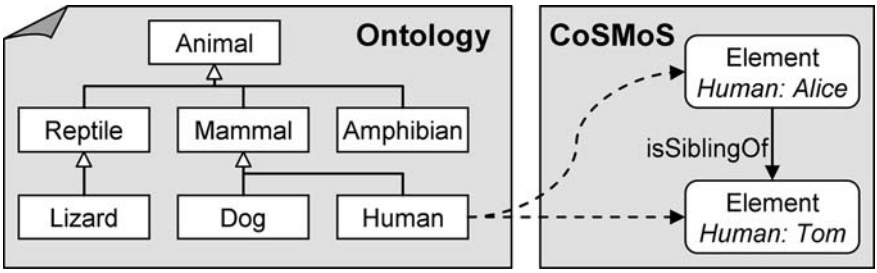


Fig. 24.3 Ontology and CoSMoS semantic representation

In CoSMoS, the nodes representing nominal concepts (e.g., "Microphone" and "Sound") are called element nodes, and the nodes representing verbal concepts (e.g., "Record" and "Play") are called action nodes. Element nodes are used to represent the semantics of components. For example, Fig. 24.2a shows that "element sound" represents the semantics of "component outSound". Similarly, action nodes (e.g., "Action Record" in Fig. 24.2a) are used to represent the semantics of operations (e.g., "Operation recordSound" in Fig. 24.2a). Element nodes may be connected via links to represent semantic relationships between them. An element node may also be defined as a wildcard to model an arbitrary element. For instance, Fig. 24.2b uses a wildcard element to model that the input sound of a voice recognition service component represents the same arbitrary element as the output text.

Representing contextual information in CoSMoS. CoSMoS represents the contexts of a component in the same way as representing the semantics of a component, i.e., by forming a graph of nodes and links using concepts defined in ontologies. Figure 24.2(c) shows an example CoSMoS representation of a user James who is 30 years old and is currently at his office. CoSMoS supports any kind of context information as long as its ontology is formally defined. Several ontologies have been already designed specifically for modeling context information (e.g., SOUPA [2]), and they can be used to model context information in CoSMoS. Some context information (e.g., user's age) may be modeled as a constant value (e.g., an integer value "30") instead of an element. Such a constant value is represented as a value node (e.g., "Value int 30" in Fig. 24.2c).

Representing rules in CoSMoS. In CoSMoS, a rule is modeled as a set of conditions and consequences – when a rule's conditions are met (e.g., "if a user is in a meeting"), its consequences become valid (e.g., "then, do not use a cell phone"). Each condition and consequence is modeled as a statement (denoted as a dotted box in Fig. 24.2), which groups a pair of nodes and a link between them. For example, Fig. 24.2c shows the rule "if I am in a meeting, do not use a cell phone" specified by a user James. Rules in CoSMoS are used by CoRE to infer new contexts (e.g., "the user is in a meeting") from other contexts (e.g., "on Monday between 9 and 10 AM") and by SeGSeC to compose applications based on a user's contexts and rules. Details of how rules are used in CoRE and SeGSeC will be explained later.

2.2 Component Runtime Environment (CoRE)

Component runtime environment (CoRE) is a middleware to support CoSMoS on various distributed computing technologies. CoRE consists of discovery manager, execution manager, and user manager (see Fig. 24.4). The discovery manager discovers components from the network and parses the metadata of the discovered components into CoSMoS semantic graph representations. The execution manager invokes an operation of a component and retrieves a property of a component. The user manager creates and manages user components.

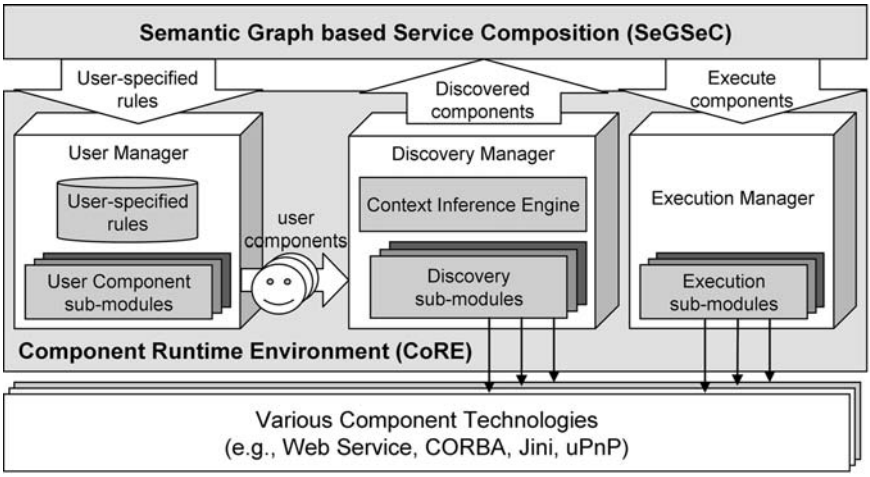


Fig. 24.4 Architecture of component runtime environment

CoRE employs a pluggable architecture where several sub-modules can be installed at start-up time. Upon receiving a query from SeGSeC, the discovery and execution managers in CoRE delegate the query to an appropriate sub-module (or sub-modules), which actually performs discovery and execution of components based on the query. By implementing multiple sub-modules based on different distributed computing technologies, CoRE is capable of supporting various distributed computing technologies. For example, the current CoRE implementation includes sub-modules based on SOAP, WSDP, RDF, and UDDI to support Web Services. Sub-modules to support other technologies, such as Jini and uPnP, are under development.

CoRE assumes that the contexts of components are acquired through existing context acquisition technologies (e.g., Context toolkit [3] and Context Fabric [8]) and have already been embedded in their metadata when they are discovered. To support a specific context acquisition technology, one may develop a custom sub-module for the discovery manager (e.g., a sub-module which acquires the coordinate of a device from a GPS sensor and embeds it in the metadata of the device). Although CoRE is not responsible for acquiring the contexts of components, the

context inference engine in the discovery manager may derive high-level contexts from the contexts of the discovered components based on user-specified rules. For example, if a user has specified a rule "I am in a meeting on Monday between 9 and 10 AM," the context inference engine in the discovery manager may derive the new context "the user is in a meeting" from the current date and time obtained from a calendar component and embed the newly derived context into the user component's CoSMoS representation.

In the proposed framework, users are modeled as components. User components are used to represent user contexts and user-specified rules in CoSMoS and are managed by the user manager in CoRE. The user manager may create a user component from user account/profile information, login session information, and/or from sensor outputs (e.g., RFID sensors). When no information about the current user is available, the user manager may temporarily create a user component with no identity information. The user manager supplies the user components to the discovery manager so that those user components can be discovered from SeGSeC.

2.3 Semantic Graph-Based Service Composition (SeGSeC)

Semantic graph-based service composition (SeGSeC) is a service composition mechanism that composes an application by synthesizing its workflow composed of multiple components based on the semantics and contexts of the components. SeGSeC assumes that all components are modeled by CoSMoS, and they can be discovered and accessed through CoRE. SeGSeC supports both rule-based context-aware service composition where it composes context-aware applications based on user-specified rules and also learning-based context-aware service composition where it composes context-aware applications based on the user preferences it acquired through learning. SeGSeC also utilizes semantic similarities among components to rank not only the components which have been used before but also newly available components based on the user preferences it acquired through learning. This improves the adaptability of SeGSeC in a dynamic environment where the availability of the components dynamically changes (e.g., due to user's movement or because components are used by other users).

SeGSeC consists of six modules, namely, user interaction module, workflow synthesis module, component selection module, learning module, semantics analysis module, and workflow execution module (Fig. 24.5).

User interaction module. The user interaction module accepts a natural language input from a user and parses it into the CoSMoS semantic graph representation through existing natural language analysis technologies (e.g., BEELINE [11]). The user interaction module then identifies if the input is a service composition request or a user-specified rule by checking whether the root node of the input (parsed as a semantic graph) is an action node or a rule node. If the input is a user-specified rule, the user interaction module passes the input to the user manager in CoRE, which stores the rule in the user component. If the input is a service composition request,

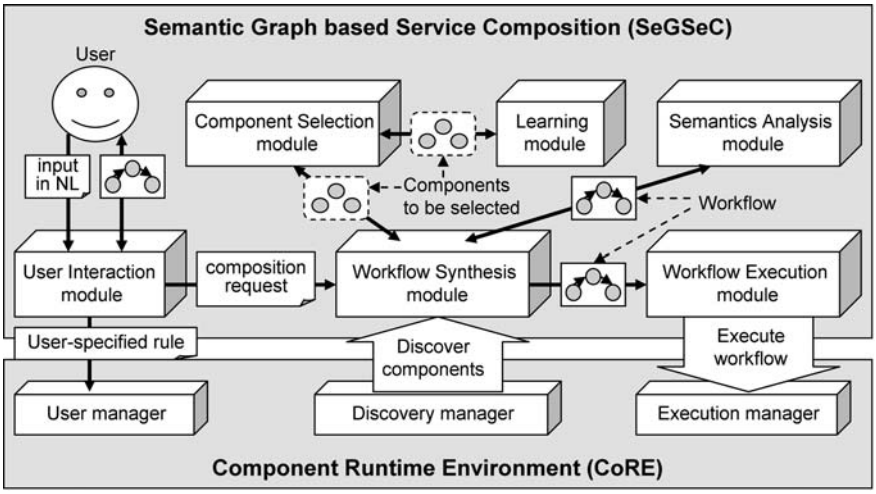


Fig. 24.5 Architecture of semantic graph-based service composition

the user interaction module passes the input to the workflow synthesis module and the semantics analysis module to compose the requested application. When parsing a natural language input, the user interaction module may discover some components from the network in order to obtain necessary semantics and rules.

Workflow synthesis module. The workflow synthesis module accepts a service composition request (represented as a CoSMoS semantic graph) from the user interaction module and synthesizes a series of workflows which satisfy the request. To synthesize a series of workflows from a given request, the workflow synthesis module first accesses the discovery module in CoRE and discovers an initial component which implements an operation to perform the action node in the request. Next, the workflow synthesis module discovers other components which can provide necessary data to execute the operation of the initial component and synthesizes a partial workflow by interconnecting the outputs or properties of the discovered components with the inputs of the initial component. The workflow synthesis module repeatedly discovers necessary components and expands a workflow until the workflow becomes executable, i.e., until all the inputs of the components in the workflow are provided by the outputs of the other components in the workflow. Once it synthesized an executable workflow, the workflow synthesis module asks the semantics analysis module to check if the synthesized workflow semantically satisfies the user request. If the semantics analysis module concludes that the synthesized workflow does not satisfy the request, the workflow synthesis module synthesizes another workflow using different components until it synthesizes the one satisfying the request.

While synthesizing workflows, the workflow synthesis module may discover two or more components which can provide the same data. In that case, the workflow synthesis module asks the component selection module to rank those components

based on their contexts of the components. The workflow synthesis module then selects the highest ranked component to synthesize a workflow.

Component selection module. The component selection module accepts a list of components and a service composition request and ranks the components based on the contexts of the components. To rank a list of components, the component selection module first gathers the context information of the components in the list and of the components referred in the composition request, which typically includes the user component corresponding to the current user. Next, the component selection module ranks the components by applying the rules specified by the current user onto the contexts of the components through an inference engine. For example, suppose that a user has specified the rule "if I am in a meeting, then, do not use a cell phone." When the user is actually in a meeting, applying the above rule onto the user's context derives the consequence "do not use a cell phone." Based on the consequence, the component selection module ranks a component corresponding to a cell phone lower than the other components in the list.

Since it is unlikely that every user has specified rules for every possible context, when the composition selection module cannot rank a list of components based on user-specified rules, it asks the learning module to rank the components based on the user preferences it acquired through learning.

Learning module. The learning module accumulates a history of user contexts and the workflows executed by the users and performs learning to derive context matching conditions from the accumulated history. A context matching condition is derived for each component in the accumulated workflows and describes under what context the component has been selected. Then, when asked by the component selection module, the learning module ranks a list of components based on the user's context, the context-matching conditions, and semantic similarities among components. By combining context matching conditions and semantic similarities, the learning module ranks a component higher even if the component has never been used before, but if it is similar to another component which has been used in the same or similar context. This allows the learning module to apply context matching conditions not only to the components which have been used before but also to newly available components, thus improving its adaptability in a dynamic environment where the availability of the components dynamically changes (e.g., due to user's movement or because components are used by other users).

The learning module derives context-matching conditions from the accumulated history of user contexts and the workflows executed by the users in the following way. (See Fig. 24.6 for a simple example of deriving context-matching conditions.) First, the learning module converts each user context represented as a CoSMoS semantic graph into a set of key-value pairs. Each key-value pair corresponds to a link in the semantic graph, where the key is a concatenation of the name of the source node and the name of the link, and the value is the name of the target node (if it is an element node) or the value of the target node (if it is a value node). Next, the learning module builds a decision tree through an existing decision tree building algorithm (e.g., C4.5 [14]) such that the decision tree represents the conditions under which the components have been chosen. The generated decision tree consists of

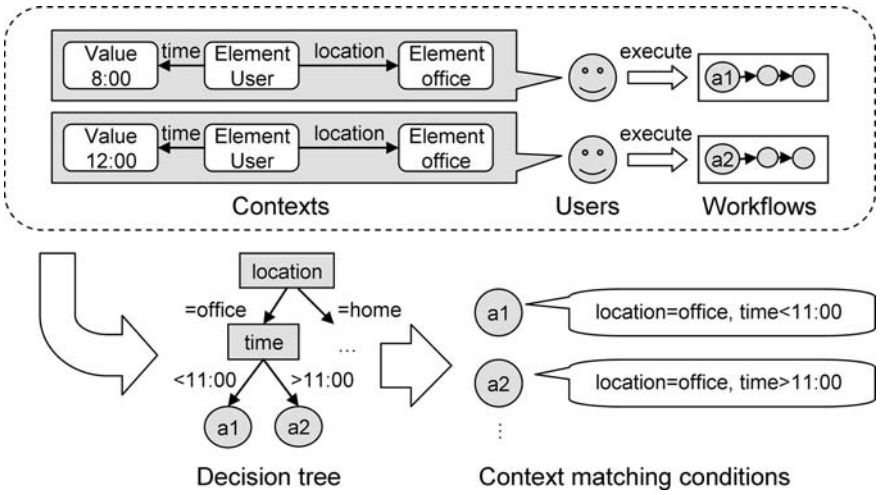


Fig. 24.6 Deriving context matching conditions

leaf nodes that correspond to the components and other non-leaf nodes that specify the conditions in which the underneath leaf nodes have been chosen. In the generated decision tree, each path from a root node to a leaf node represents a condition under which the component corresponding to the leaf node has been chosen. Therefore, the learning module traverses the tree from its root node to each leaf node to derive the context matching condition of the component at the leaf node. The learning module derives and stores context-matching conditions for all the components in the accumulated workflows.

When asked by the component selection module, the learning module ranks a list of components based on the current user's context, the context matching conditions, and semantic similarities among components. To rank a list of components, the learning module first calculates context-matching degrees of the components using their context-matching conditions and the current user's context. Since a context-matching condition consists of multiple sub-conditions, a context-matching degree is calculated as the ratio of the sub-conditions in the context-matching condition that are satisfied in the current user's context. For example, if a context-matching condition consist of two sub-conditions, "*location* = *office*" and "*time* > 12:00", and the current user's context contains "*location* = *office*" and "*time* = 11:00", the context-matching degree is $1/2 = 0.5$. Next, the learning module calculates semantic similarities between the components through the existing graph matching scheme (e.g., [10]). Then, using the semantic similarities and context-matching degrees, the learning module computes preference values of the components in the list. A preference value P_i of a component i is calculated as

$$P_i = \max\left(S_{i,j} \times (C_j + \alpha)\right) 1 \leq j \leq n$$

where $S_{i,j}$ is a semantic similarity between component i and j, C_j is a context-matching degree of component j, n is the total number of components, and α is an arbitrary constant value (e.g., 0.001) to preserve the semantic similarity even if the context matching degree is zero. The learning module ranks the components based on their preference values, i.e., it ranks those with higher preference values higher. In this manner, even when some components which have been used before become unavailable, the learning module ranks the other components which are similar to the unavailable ones.

Semantics analysis module. The semantics analysis module accepts a service composition request and an executable workflow synthesized by the workflow synthesis module and examines if the semantics of the workflow satisfies the request or not. In order to examine a workflow against a composition request, the semantics analysis module first extracts the semantics of the workflow by applying semantic retrieval rules [5, 7] onto the workflow through an inference engine. The semantics retrieval rules are defined to derive the semantics of the workflow from the semantics of the components in the workflow and from the data flows (i.e., which inputs should be provided by which outputs or properties) defined in the workflow. Then, the semantics analyzer module compares the extracted semantics of the workflow against the composition request. Since both the semantics of the workflow and the service composition request are represented as CoSMoS semantic graphs, the semantics analyzer module concludes that the semantics of the workflow satisfies the request if the composition request is a sub-graph of the semantics of the workflow.

If the semantics analysis module concludes that the semantics of the workflow satisfies the composition request, the user interaction module presents the workflow to the user and asks whether to execute the workflow or not. If the user agrees to execute the workflow, the workflow execution module executes the workflow, and the learning module records the workflow along with the user's context. If the semantics of the workflow does not satisfy the request, or if the user does not agree to execute the workflow, the workflow synthesis module synthesizes another workflow using different components.

Workflow execution module. The workflow execution module accepts an executable workflow and executes it through the CoRE's execution manager by invoking operations of the components and retrieving properties of the components as specified in the workflow.

3 Case Study

The proposed semantics-based context-aware dynamic service composition framework has been implemented in Java. The Java implementation of the proposed framework is built upon Java SE 1.5, Jena 2 [9] (for inference) and Weka 3 [17] (for building decision trees). It supports components implemented as Java class files (with CoSMoS annotations [7]) and as Web Services (described in WSDL and RDF [7]).

In order to test and evaluate its feasibility and correctness, a virtual environment is emulated using the Java implementation of the proposed framework, and case studies have been conducted by deploying various components and composing various context-aware applications in the virtual environment. Figure 24.7 shows the screenshot of the virtual environment emulated on the Java implementation of the proposed framework. The following describes one of the context-aware applications composed in the virtual environment.

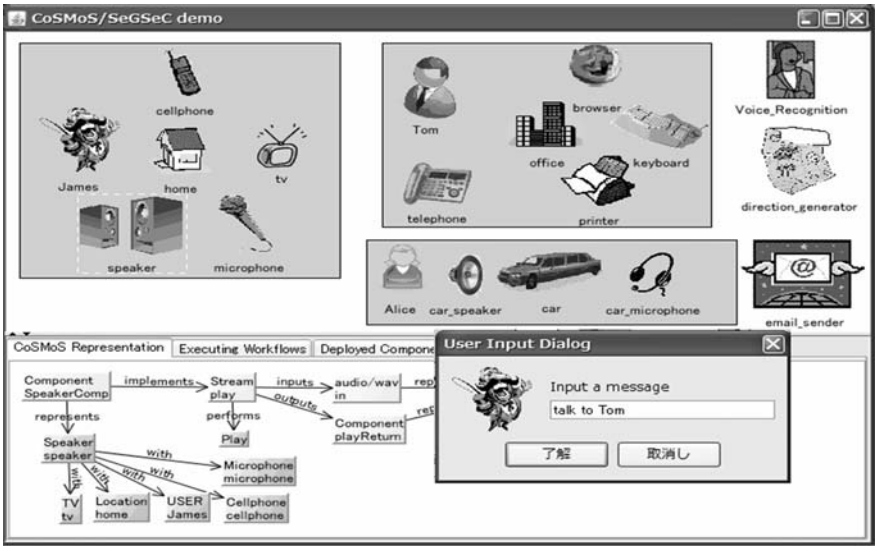


Fig. 24.7 Screenshot of the proposed framework

In a virtual environment, James requests an application to talk to Tom. James is at home where a microphone, a speaker, and a cell phone are available, and Tom is at an office where a telephone is available. When talking to Tom, James prefers to use a microphone and a speaker when nobody else is at home so that he can freely move around, but prefers to use a cell phone if somebody else is also at home. The locations of James, Tom, and the device components are monitored by (emulated) location sensors deployed at home and the office. If James has explicitly specified his preferences by inputting the sentences "I use a speaker and a microphone if I am alone" and "I use a cell phone if I am not alone", SeGSeC applies the rules when composing the requested application and selects device components (i.e., a pair of speaker and microphone or a cell phone) based on his context (i.e., whether he is alone at home or not) accordingly. Or, even though James has not explicitly specified his preferences, if he always chooses a speaker and a microphone (or a cell phone) every time he talks to Tom when he is alone (or not alone), SeGSeC autonomously learns his preferences and chooses the device components accordingly.

4 Simulation Experiments

In order to evaluate the performance of the learning-based context-aware service composition of the proposed framework, a series of simulation experiments have been conducted. In the simulation experiments, a simulator based on the Java implementation of the proposed framework is used, and the performance of the learning-based context-aware service composition of the proposed framework is evaluated against the heterogeneity and dynamics in user contexts, against the number of possible workflows (i.e., workflows which can be synthesized from the deployed components), and in dynamic environments where the availability of components dynamically changes.

4.1 Simulation Configuration and Evaluation Metrics

Figure 24.8 illustrates the simulation configuration. In the simulation experiments, 25 components are deployed in a simulated environment. Those components are categorized into five groups, A, B, C, D, and E, each of which consists of exactly five components. The components in group A (named A1 ... A5) implement a service to generate an alphabet "A". The components in group B (named B1 ... B5) implement a service to convert an alphabet "A" to "B". Similarly, the components in group C, D, and E implement services to convert "B" to "C", "C" to "D", and "D" to "E", respectively. Using those components, a pseudo-application which generates an alphabet "E" is composed. The pseudo-application can be composed by synthesizing any of the $5^5 = 3{,}125$ possible workflows, from A1 → B1 → C1 → D1 → E1 to A5 → B5 → C5 → D5 → E5. Each component has a probability of becoming unavailable to simulate the dynamic availabilities of components. Since the proposed framework utilizes semantic similarities among components, a semantic similarity is randomly assigned for every pair of components.

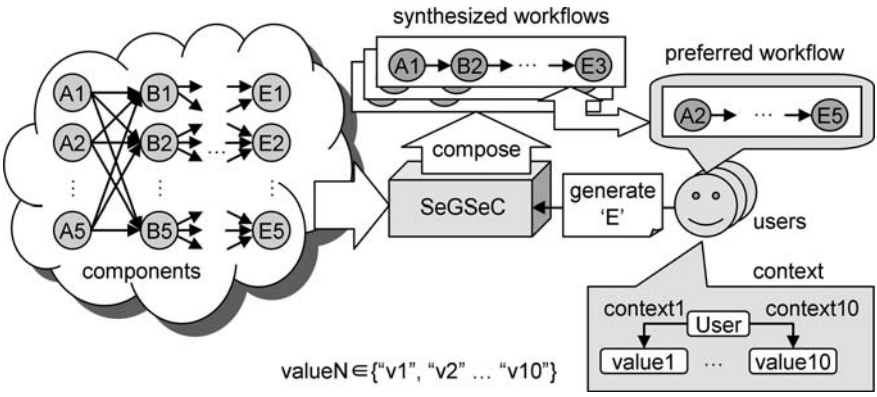


Fig. 24.8 Simulation configuration

In each experiment, the simulator first generates a certain number of users. Each user has a set of 10 randomly generated values to represent his context. In the simulation experiments, a single human (e.g., James) in different contexts (e.g., at home and at office) is simulated as different users (e.g., James at home and James at office). Therefore, in the simulation experiments, the heterogeneity of user contexts and the dynamics of user contexts are both measured by the number of users simulated. Each user also has a specific workflow (called "preferred workflow") that he wants to execute in his context. A user agrees to execute a workflow if it is similar to his preferred one. The similarity between two workflows is calculated as the average of the semantic similarities of the components in those workflows.

After generating a certain number of users, the simulator randomly chooses a user to request the pseudo-application. Then, the simulator counts the number of workflows synthesized by the proposed framework until it synthesizes the one which the user agrees to execute. This number of synthesized workflow indicates the efficiency of the proposed framework because it requires more computational and network resources to synthesize a larger number of workflows. The simulator repeats the step of randomly choosing a user from the generated ones and requesting to compose the pseudo-application 100 times and calculates the average of the number of the synthesized workflows. The simulator also measures the success rate that the current user agrees to execute the first workflow synthesized by the proposed framework. The success rate is calculated as the probability that the number of the synthesized workflows becomes one in requesting the pseudo-application.

For comparison, the simulation experiments compare the performance of four schemes: a scheme which randomly selects components (called "Random"), a scheme which counts how many times each component has been used and selects the ones that are most frequently used (called "Popular"), a scheme which implements SeGSeC but does not utilize semantic similarity in the learning module (called "SeGSeC w/o similarity"), and a scheme which implements SeGSeC and utilizes semantic similarity in the learning module (called "SeGSeC with similarity").

4.2 Performance Evaluation Against Heterogeneity and Dynamics of User Contexts

In the first set of simulation experiments, the performance of the proposed framework is measured while varying the number of users simulated from 1 to 100. Figure 24.9a shows that the number of synthesized workflow increases as the number of users increases, but SeGSeC (both with similarity and without similarity) scales better than the random scheme and the popularity-based scheme. Figure 24.9b shows that as the number of users increases the success rate of SeGSeC decreases, but still remains much higher than those of the random scheme and the popularity-based scheme. As described earlier, in the simulation experiments, the number of users indicates both the heterogeneity and the dynamics of user contexts. Therefore,

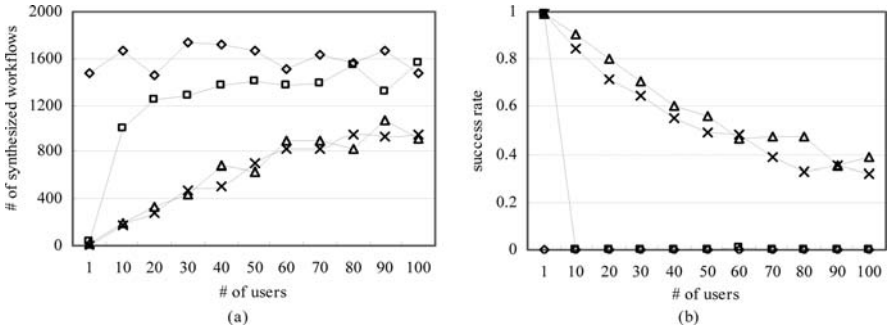


Fig. 24.9 Performance over different number of users

the results of the first set of experiments indicate that SeGSeC is capable of composing the workflows preferred by users efficiently and with a high success rate even when the user contexts become heterogeneous or dynamic.

4.3 Performance Evaluation Against the Number of Possible Workflows

In the second set of simulation experiments, different numbers of components are deployed, and the performance of the proposed framework against the number of possible workflows (i.e., workflows which can be composed from the deployed components) is measured. In this set of experiments, 10 users are simulated. Figure 24.10a shows that the number of synthesized workflow increases as the number of possible workflows increases, but SeGSeC (both with similarity and without similarity) scales better than the random scheme and the popularity-based scheme. Figure 24.10b shows that SeGSeC maintains high success rate (above 0.8) even when the number of possible workflows increases, while the success rates of the random and popularity-based schemes quickly drop down close to 0. Those results

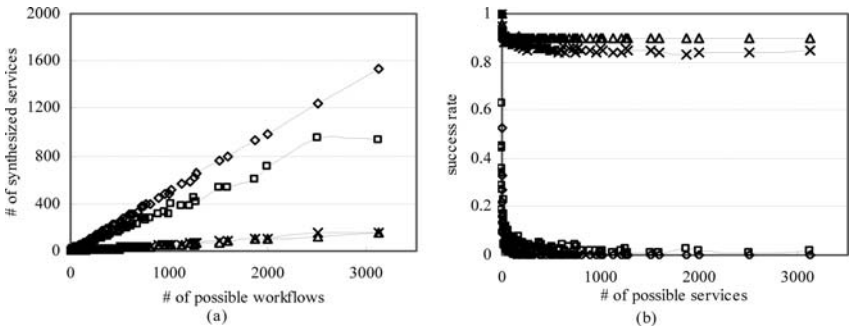


Fig. 24.10 Performance over different number of possible workflows

indicate that SeGSeC is capable of composing the workflows preferred by users efficiently and with a high success rate even when more components are deployed and the number of possible workflows increases.

4.4 Performance Evaluation Against Dynamic Environments

In the last set of simulation experiments, the probability of components to become unavailable is varied from 0.0 to 1.0, and the performance of the proposed framework against the dynamics in the availability of components is measured. In this set of experiments, only a single user is deployed, meaning that the user's context and preferred workflow remain static. Figure 24.11a shows that, as more components become unavailable, the number of synthesized workflows increases first, but then starts decreasing. This is because when too many components become unavailable the number of possible workflows also decreases. Figure 24.11a also shows that SeGSeC with similarity synthesizes the smaller number of workflows compared to the other schemes. Figure 24.11b shows that as more components become unavailable the success rate decreases, yet SeGSeC with similarity achieves a higher success rate compared to the other schemes. As described in Section 2, even when some components which have been used before become unavailable, SeGSeC with similarity is still capable of selecting the other components which are similar to the unavailable ones by using the semantic similarity among the components. Therefore, SeGSeC with similarity is more robust against the dynamics in the availability of components.

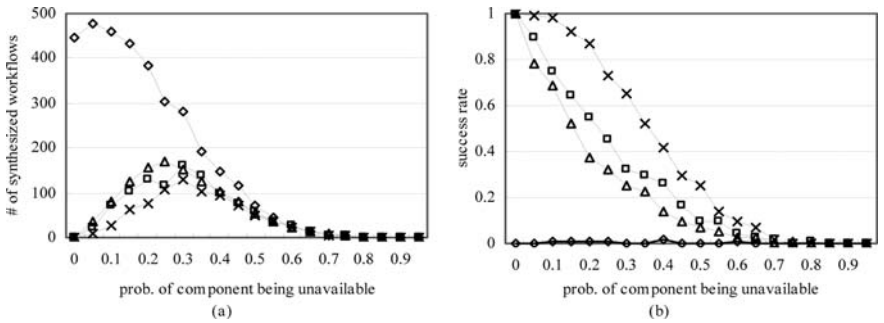


Fig. 24.11 Performance over different prob. of component being unavailable

5 Conclusions

This chapter proposed the semantics-based context-aware dynamic service composition framework, which allows users to request applications in a natural language and supports both rule-based context-aware service composition and learning-based context-aware service composition. Case studies and simulation experiments proved

that the proposed framework is capable of composing various context-aware applications in a large-scale and dynamic environment. Compared to other existing context-aware dynamic service composition systems (e.g., [13, 18]), the proposed framework is novel in that (1) it allows not only system designers but also end users to specify rules on how to compose context-aware applications, (2) it supports both rule-based context-aware service composition and learning-based context-aware service composition, and (3) it utilizes semantic similarities among components to improve its adaptability in a dynamic environment. It is still necessary to conduct further empirical evaluations to deploy the proposed framework in real environments and investigate its adaptability. This awaits further research.

Acknowledgements This research is supported by the NSF through grants ANI-0083074, ANI-9903427, and ANI-0508506, by DARPA through grant MDA972-99-1-0007, by AFOSR through grant MURI F49620-00-1-0330, and by grants from the California MICRO and CoRe programs, Hitachi, Hitachi America, Hitachi CRL, Hitachi SDL, DENSO IT Laboratory, DENSO International America LA Laboratories, NICT (National Institute of Communication Technology, Japan), NTT Docomo and Novell.

References

1. D. Chakraborty and A. Joshi. Dynamic service composition: State-of-the-art and research directions. Technical Report Technical Report TR-CS-01-19, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, USA, 2001.
2. H. Chen, T. Finin, and A. Joshi. The SOUPA ontology for pervasive computing. *Ontologies for Agents: Theory and Experiences*, Birkhauser, Boston, 2005.
3. A.K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, Dec. 2000.
4. P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition using Markov decision processes. In *Proceedings of the Second International Conference on Web Services (ICWS)*, pp. 576–582, San Diego, CA, Jul. 6–9, 2004.
5. K. Fujii and T. Suda. Semantics-based Dynamic Service Composition. *The IEEE Journal on Selected Areas in Communications (JSAC)*, 23(12):2361–2372, December, 2005. Special issue on Autonomic Communication Systems.
6. K. Fujii and T. Suda. Dynamic service composition using semantic information. In *Proceedings of the Second International Conference on Service Oriented Computing (ICSOC 2004)*, New York, USA, Nov. 2004.
7. K. Fujii and T. Suda. Semantics-based dynamic Web service composition. *The International Journal of Cooperative Information Systems (IJCIS)*, 15(3):293–324, Sep. 2006. Special issue on Service-Oriented Computing.
8. J.I. Hong and J.A. Landay. An infrastructure approach to context-aware computing. *HCI Journal*, 16:287–303, 2001.
9. Jena Semantic Web Framework. http://jena.sourceforge.net
10. J. Kwon, O. Choi, C. Moon, S. Park, and D. Baik. Deriving similarity for semantic Web using similarity graph. *Journal of Intelligent Information Systems*, 26(2):149–166, Mar. 2006.
11. G.A. Mann. BEELINE – A situated, bounded conceptual knowledge system. *International Journal of Systems Research and Information Science*, 7:37–53, 1995.
12. S. McIlraith and T. Son. Adapting Golog for composition of semantic Web services. In *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)*, Toulouse, France, pp. 482–493, Apr. 2002.

13. S.B. Mokhtar, D. Fournier, N. Georgantas, and V. Issarny. Context–aware Service Composition in Pervasive Computing Environments. In *Proceedings of the 2nd International Workshop on Rapid Integration of Software Engineering Techniques (RISE'2005)*, Heraklion, Crete, Greece, Sep. 2005.
14. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
15. J.F. Sowa. Conceptual graphs summary. In P. Eklund, T. Nagle, J. Nagle, and L. Gerholz, editors, *Conceptual Structures: Current Research and Practice*, pp. 3–52, Ellis Horwood, 1992.
16. P. Traverso and M. Pistore. Automated composition of semantic Web services into executable processes. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, Hiroshima, Japan, 7–11 Nov. 2004.
17. Weka 3 – Data Mining with Open Source Machine Learning Software in Java. http://www.cs.waikato.ac.nz/ml/weka/
18. K. Yang, S. Ou, A. Liotta, and I. Henning. Composition of context-aware services using policies and models. In *Proceedings of the IEEE Globecom 05*, St. Louis, USA, 28 Nov.–2 Dec. 2005.

Chapter 25
Distributed e-Science Application for Computational Speech Science

K. Nozaki, K. Baba, M. Noro, M. Nakagawa, S. Date, and S. Shimojo

Abstract The computational speech science portal is a highly intensive data sharing and remote visualization environment that uses e-Science technologies to support scientists in accessing distributed databases and visualization resources for solutions of the physics of human speech problems. These recourses include input files specifying the human oral cavity geometry of speech, and proprietary software and hardware for visualization and data sharing of computational fluid dynamics (CFD) results. Through a Web-based grid portal, a user can access these resources and collaboratively visualize the results of the CFD simulation and sharing the result of it. Thus, the portal allows users to use the e-science infrastructure to share resources among geographically distributed partners in order to execute large-scale CFD simulations and to collaboratively analyze and visualize the results. However, among geographically distributed partners, large-scale data normally cannot be transferred in a short period of time because of the long round-trip time of the TCP protocol. This chapter describes the web-based portal, built on the Gridsphere framework. The portal is composed of a number of JSR-168 compliant portlets that deploy services at the back end for the management of distributed resources and for the invocation of services for visualization. The chapter also describes how high-speed data transfer is achieved through the e-science infrastructure.

Keywords Remote visualization · e-Science web portal · Distributed Environments · Speech science · Computational fluid dynamics · Aeroacoustics · Sibilant · Fricative voice

1 Introduction

High-performance computing (HPC) resources are not currently easy for speech scientists to use [12]. The physics of voice production of speech is based on fluid dynamics [15]. Such voiced sound has been analyzed by signal processing

K. Nozaki (✉)
Cybermedia Center, Osaka University, Japan
e-mail: nozaki@cmc.osaka-u.ac.jp

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_25,

methods, which either summarize or extract characteristics of a targeted sound signal. In terms of the relationship between the morphological features of oral cavities and voiced sound, transfer functions have mainly been applied for oral models [13]. The transfer function analysis method cannot represent the actual oral cavity in principle [16]. Nowadays, oral morphologies can be taken in three dimensions.

Improving the fundamental analytical method for voiced sound requires enough comparisons between well-simplified transfer function models and finite element method (FEM) models [11]. The FEM is mainly used for computational fluid dynamics (CFD) and acoustics [18] and produces data volumes exceeding gigabytes. However, few researchers are focusing on this aspect for the comparison process. Speech scientists are not specialists in computers and networks. Moreover, they are very often geographically separated. In brief, we have two significant issues: easy access to resources and sharing large size volumes of data.

Web-based portal environments make it easy for computational scientists to use resources and share their results [3, 7]. In some instances, FEM calculation is much too heavy for a single node to perform. Parallel calculation is therefore required. The resultant data volume is normally too large to store in single-volume storage. Distributed storage is therefore required. In order to free a user to worry about those data managements, an integrated user interface should be developed, such as a web interface. Web-based applications are often developed by using Java servlet and JavaServer Pages (JSP). However, taking into consideration the need for reusability and easy-to-use interfaces, we need a framework that makes a separation between service providers and user interfaces. The portlet concept can provide such a framework. Portlets describe visual user interfaces to a content or service provider. The portlet concept frees portlet developers from worrying about the presentation layout of portlets and Web page composition, so that they can focus solely on providing functionality and usable interfaces [1]. Currently, the portlet API is included in IBM's WebSphere portal, Oracle's Portal Server, and Sun's Java System Portal Server, and the portlet specifications are standardized as Java Specification Request (JSR) 168. The Jakarta Jetspeed project provides an implementation of the evolving portlet API. Gridsphere [9] is also based on JSR168 and includes many basic services, such as user manager service, access control service, job manager service, job monitoring service, file transfer service, and data manager service, all of which are appropriate for computational speech scientists' requirements.

Grid middleware enables the sharing of geographically dispersed scientific data resources [4]. Computational speech scientists can access data through the Web portal. Mainly, there are three grid storage middlewares: storage resource broker (SRB), grid datafarm, and globus RLS. Among them, SRB has a lot of scientific users [2]. Grid datafarm provides virtual directories, such as "/gfarm", and it can be mounted on each node. SRB allows the access of using either Dublin Core metadata or a science-specific metadata, but grid datafarm does not aim at such kinds of access [17]. However, for world-wide geographically distributed data sharing, there is still room for improving the performance and function of data transfer, for which the

Reno TCP protocol is usually used SRB [14]. The round-trip time (RTT) increases considerably for data transfer between long-distance parties or when there are many hops in networks.

This chapter presents a framework for geographically dispersed collaborative study for computational speech science and describes our design for an e-science system architecture. It also reports an assessment of our system at SC2007.

2 Computational Speech Science

Improvements in computational speech science are expected to provide the medical and dental clinical basics needed to diagnose and treat speech disorders and prevent voice changes after dental treatments. An integrated simulation of CFD and computational aeroacoustic simulation (CAA) aims at providing a clinical index considering the prognostic of a disease. Currently, it is only possible to simulate a phoneme of a sibilant /s/. The sibilant /s/ most frequently occurs in speech problems related to dental treatments. Articulated phrases, such as /u-su-i/, cannot be simulated by using only conventional methods. The integrated simulation can help in predicting speech changes and thereby provide strategies for preventing them (Fig. 25.1) [5]. However, a university or hospital cannot prepare the vast resources needed for such integrated simulation by [10].

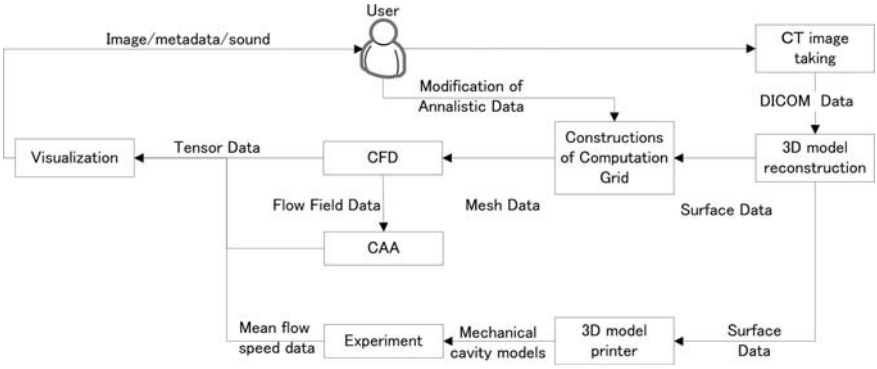


Fig. 25.1 Workflow of an integrated simulation system

2.1 Computed Tomography (CT) Imaging, Three-Dimensional Model Reconstruction, and Construction of Computational Grids

CT images can be used for clinical diagnoses and for the construction of three-dimensional human structures for medical simulations. Cone-beam CT keeps X-ray exposure doses quite low, and the CT image resolution is more precise than that of

general CT. The geometry of the oral tract pronouncing the sibilant /s/ was acquired by cone-beam CT. It took 9.6 s to acquire the whole volume ($512 \times 512 \times 512$) of the human oral area. In this study, we took a 0.1-mm-mode voxel for one patient who is one of the authors, and DICOM image processing software (RealINTAGE, KGT Corp, Japan) was used to read DICOM data and to visualize the data. Polygonal data was constructed as a STL file by using the Martin Cubes method. A hexahedral mesh was generated with Gridgen v15 (Pointwise Co. Ltd., USA) by tracing STL data.

2.2 CFD/CAA Simulations of Oral Airflow of Sibilant /s/

The integrated simulation needs a large number of meshes because of the representation of the complex morphological shapes and the consideration of turbulence related to sound generation. Large eddy simulation (LES) is the major method for modeling turbulence in aeroacoustics. LES does not average flow fields in a time course, but instead averages them in space. The grid scale is recognized as a filter size of space in LES using the dynamic Smagorinsky model. Hence, the accuracy of the simulation somehow depends on the quality of the mesh. In this study, we used FrontFlow Blue v5 (Tokyo Univ. Japan), which is a LES solver including a CAA tool [6]. This CFD software can be performed in parallel by using MPI. The overall workflow of this application is outlined in Fig. 25.1.

2.3 Current Issues in Computational Speech Science

For geographically dispersed collaborative research, data sharing and collaborative visualization are the primary objectives. In terms of application for clinical treatments, some kind of an interface is required. Hospitals themselves are not expected to be able to prepare the several kinds of HPC tools needed. However, in clinical cases of speech problems, the scientist needs an understanding of the realistic speech pathology. A mediation layer between the scientific problems and clinical problems is therefore needed. Then, the e-science infrastructure should cover the layer between them.

3 Applying e-Science for Computational Speech Science

Visualizing the integrated simulation results requires at least more than 2 GB of memory. The e-science infrastructure is expected to free speech scientists, doctors, and dentists from worrying about being lacking sufficient memory and visualization resources, about recalling what they have done, and about security. This chapter proposes the wide-area distributed computing environment. The distributed servers are aggregated by deploying a grid, network, and storage middleware through Web services (Fig. 25.2).

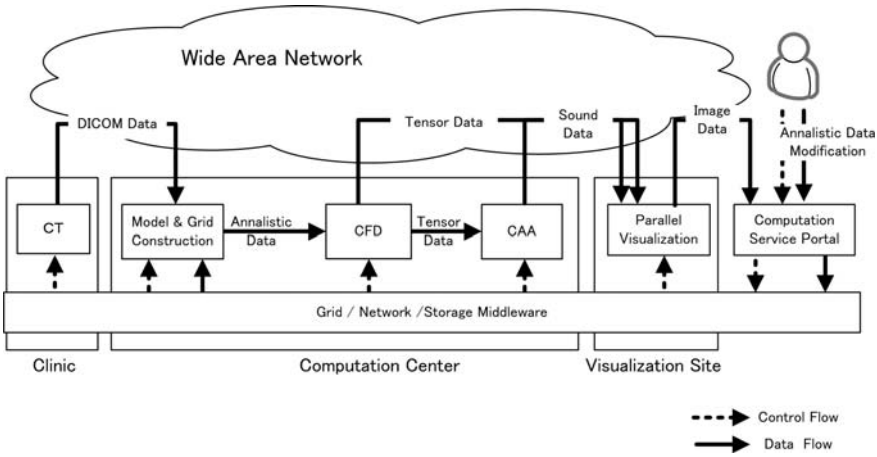


Fig. 25.2 Overall proposed e-science architecture for computational speech science

This architecture aims at providing doctors and speech scientists with visualization and database service as a part of computational services. A Web portal can work as the interface between users and this service. However, in the geographically dispersed environment, the RTT causes some problems for our proposed Web portal system. Especially, remote visualization tends to lack interactivity through the Web.

3.1 Computational Speech Science Portal

In our design of the computational speech science portal, we attempted to satisfy the following requirements:

1. *User registration*. Users should be registered at their first visit. The portal administrator should approve registration applications.
2. *User access*. The portal should provide the users with a single sign-on without multiple log-ins for each server.
3. Visualization. Users should be able to retrieve their necessary simulation or CFD data or both by inputting key phrases in the portal. Visualized images should be controllable in the portal, such as rolls, zoom-in/out, and shifts. Metadata should be added to the volume data referred with the geometry clicked on the image display window in the portal.

The architecture of computational speech science portal uses Gridsphere2.2.8 composed of portlets. Gridsphere is a portlet container. The portlet container provides a runtime environment where a portlet can be installed, used, and finally destroyed. In the system architecture, the portal framework is an additional layer, with a well-defined API, that provides a standard interface for content presentation. In traditional multitier architectures, we usually define three layers: persistence,

application logic, and presentation. The portal framework is a fourth layer that exists between the application logic and presentation layers and provides a consistent method of generating presentation content from application logic. A GAMA portlet was installed to the portlet container of Gridsphere for user registration and user access. The GAMA server authenticates users and retrieves their proxy certificate using the Web-services APIs. GAMA portlets include the stub of the GAMA server's wsdl. GAMA portlets provide the user registration and authentication function with the Portal Framework. The X.509 certification is generated through the Web portal and X.509 proxy certification is given through the GAMA portlet. The visualization portlet was the modification of the proprietary remote visualization system. The inherit system was composed of servlets in the tomcat 5.5.25 container, JSP, and AJAX for the user interface.

3.2 Visualization Portlet

The portal framework allows proprietary software to be plugged into it. Our proprietary remote distributed visualization system (RDVS) could be installed into the Gridsphere portal framework (Fig. 25.3). The RDVS consists of servlet in tomcat container. The main process of the RDVS is controlled by ViewControl class that extends HttpServlet class. In order to install the RDVS into the framework, some modifications were made as follows:

i. The master class of ViewControl was changed from HttpSession to GenericPortlet class.
ii. The HttpSession class was changed to PortletSession class.
iii. Request-processing methods were changed. The doPost() method in the HttpServlet class was included in the portlet container in Gridsphere. Hence, four methods were prepared for the portlet program (processAction(), doView(), doEdit(), and doHelp), so that it could receive requests from the portlet container.

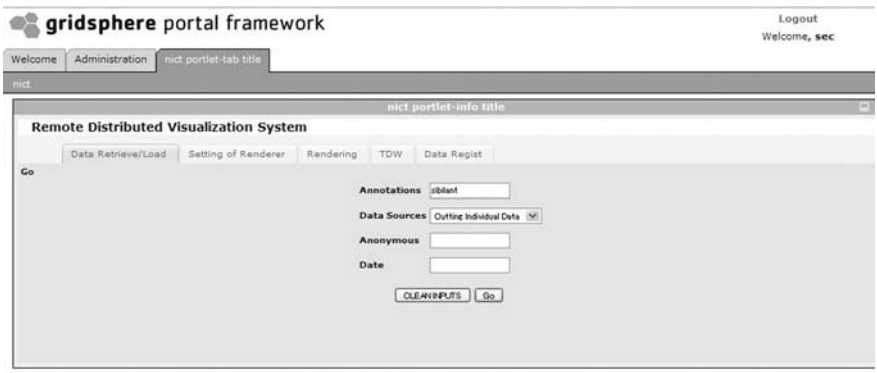


Fig. 25.3 Retrieval page of the visualization portlet

The contents of the doPost() method was almost involved in processAction() method. The doView() method was changed to show an adequate JSP image by getting the image number from the session information when this method was called.

iv. The URL described in JSPs was changed. Portlets must be independent of the URL (see JSR168 Portlet Specification PLT.3).

The portlet of the RDVS provides the search interface to retrieve the volume database. Users can select the data source type, such as CT, magnetic resonance imaging (MRI), and numerical simulation results. The annotation form can accept any text and reply with the matched annotation of metadata added to the three-dimensional geometry in the volume data. The anonymous form can accept anonymous IDs so that users can get their own data, knowing what their data was (Fig. 25.3).

The visualization setting phase is displayed after the requested data set is obtained to the visualization node from geographically dispersed storage. The visualization settings can be modified in this phase. When no changes are made on the settings, the proprietary settings are reflected (Fig. 25.4).

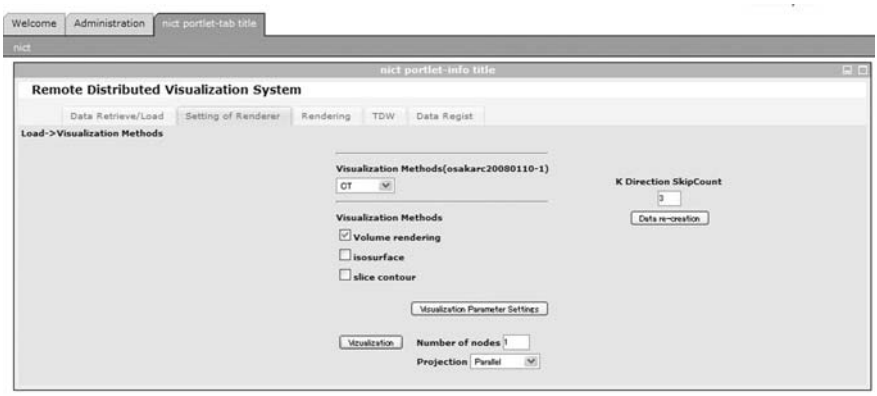


Fig. 25.4 Setting page of the visualization portlet

The RDVS portlet is the user interface of the visualization software (AVS, KGT Corp.). In order to keep the interface independent of the visualization software, a standardized formatted message was implemented for communication between the visualization portlet and visualization software. The message format was written in XML and is called Viz.xml. Viz.xml contains all of the setting parameters that indicate how to visualize the volume data and knows how to visualize the volume data with respect to angle, scale, color, and so on (Example 25.1). However, to communicate through Viz.xml, each side needs to prepare a communication module that can understand it. Hence, we made a Viz.xml module for AVS.

For collaborative visualization among researchers or doctors, metadata is necessary in order for them to be able to recognize what the visualization means. The annotation function provides as many notes on the visualized image as the user needs. The notes are related to the three-dimensional geometry in the volume data (Fig. 25.5). Characteristically, users can retrieve the notes in the search phase and the automatically visualized image is displayed with the previously saved condition stored as Viz.xml in the resource description framework (RDF) database.

Example 25.1 Example of Viz.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <viz>
  - <VizParam>
      <renderer number="1" />
      <note instance="1">CT volume and CFD result</note>
      <timestep>5</timestep>
  + <ParamItem id="param1" instance="1" iso="0" slice="0" volume="1">
      </ParamItem>
  - <ParamItem id="param2" instance="1" iso="0" slice="0" volume="0">
        <Data coord_max_x="52.8088" coord_max_y="36.0918" coord_max_z="
            40.1538" coord_min_x="6.45395" coord_min_y="-2.95353"
            coord_min_z="15.6307" filepath="/To7.avs.txt.inp" max="
            140.3928680419922" min="0.0" stepmax="5" type="UCD" typestr="
            Pressure" />
      + <iso_surface id="iso_surface_0" instance="0"></iso_surface>
      + <volume_render id="volume_render" instance="0"></volume_render>
    </ParamItem>
  + <ParamItem id="param3" instance="1" iso="1" slice="1" volume="0">
    </VizParam>
  - <ViewParam>
      <scale>1.0</scale>
      <Xform>ocenter = {255.5, 248.418, 255.5}; dcenter = {3.89211, 3.78422,
          3.89211}; mat = { 0.00999395, 0., -0.0114967, 0., -0.011105,
          0.00394267, -0.00965341, 0., 0.00297557, 0.0147142, 0.00258662,
          0., 0., 0., 0., 1. }; xlate = {-3.89211, -3.78422, -3.89211};
          center = {255.5, 248.418, 255.5};</Xform>
      + <annotation edit="0" id="annotation0" instance="1"></annotation>
      + <annotation edit="0" id="annotation1" instance="1"></annotation>
      + <annotation edit="0" id="annotation2" instance="1"></annotation>
      - <annotation edit="0" id="annotation3" instance="1">
          <timestep>5</timestep>
          <sentences>"Sibilant groove"</sentences>
          - <coord>
              <x>230.5</x>
              <y>194.46887</y>
              <z>240.40717</z>
          </coord>
        </annotation>
      + <annotation edit="0" id="annotation4" instance="1"></annotation>
      + <annotation edit="0" id="annotation5" instance="1"></annotation>
      + <annotation edit="0" id="annotation6" instance="1"></annotation>
      + <annotation edit="0" id="annotation7" instance="0"></annotation>
        <perspective switch="0" />
    </ViewParam>
  + <OutputImage></OutputImage>
</viz>
```

Fig. 25.5 Rendering page of the visualization portlet

3.3 Geographically Dispersed Storage

Distributed storage managing software should have the following functions:

i. Single sign-on (SSO) for each storage server.
ii. Virtualization of storages as a single volume.
iii. Metadata management.
iv. High-speed data access.

The grid security infrastructure (GSI) is appropriate for the SSO solution. In our e-science infrastructure, Globus4.1.2 was installed into each node to achieve SSO. SRBv3.2.4 could represent a logical single volume with the Meta Catalog Server (MCAT) for geographically distributed physical storage. SRB nodes can communicate by GSI. MCAT can manage metadata through Scommands or jargon APIs. However, currently, XML-formatted metadata, such as RDF, is recognized as a widespread standardized metadata format. Hence, our system was designed for hybrid metadata management, using both the MCAT server and metadata server, which can deal with the RDF database and communicate by Web service. SRB frees users from worrying about the physical location of storage and the amount of storages, but they still must pay attention to the data acquisition time, especially for large-scale data (Fig. 25.6).

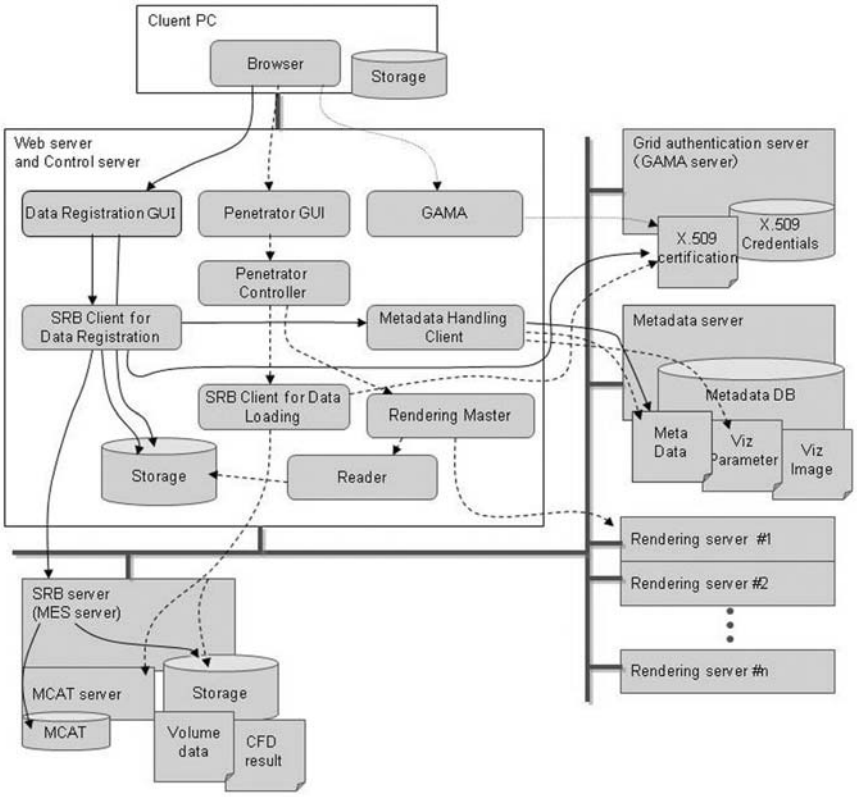


Fig. 25.6 RDVS components' architecture

3.4 Large-Scale Data Transfer Over Wide-Area Networks

The data size of the output from CFD is very large. In our infrastructure, this data transfer uses a wide-area IP network and broadband access links. It is not easy to transfer large CFD data for visualization over a wide-area network. We are developing a data transfer protocol (gUDT, gentle UDT) for large-scale data transfer over a wide-area IP network. To obtain our protocol, we modified UDT for enhancement of TCP friendliness over long distances and high-speed networks. The gUDT can transfer large amounts of data quickly using available network bandwidth on the Internet.

4 Current Implementations and Assessment at SC07 and JGN2 Symposium 2008

We assessed the usability of the RDVS portal in a real geographically distributed environment. There were two issues to be checked in the realistic environment: (1) remote visualization of a large size volumetric data sent through a wide-area

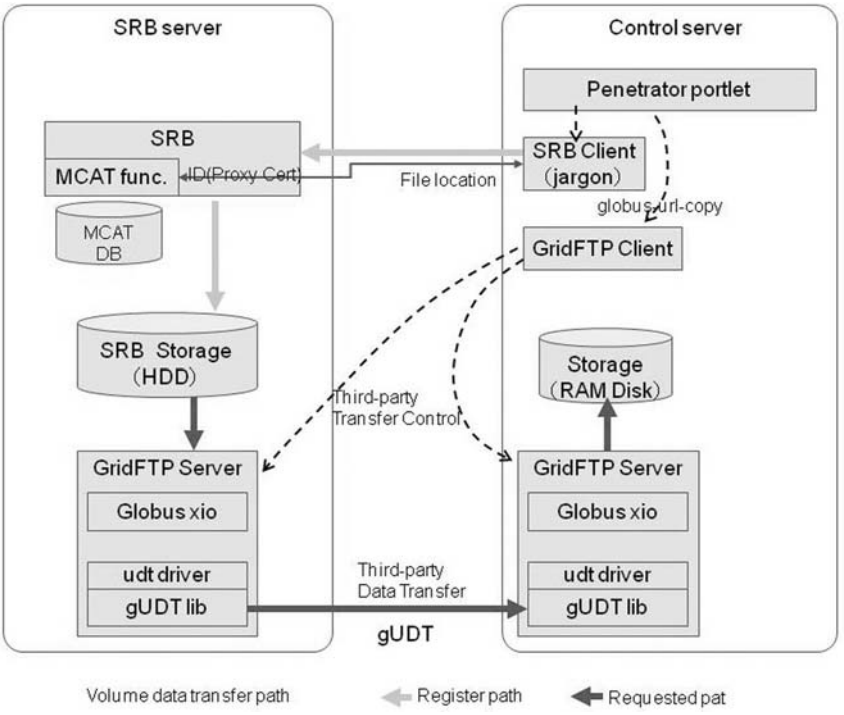


Fig. 25.7 SC07 demonstration system

network and (2) high-definition image display at remote places. The system architecture is outlined in Fig. 25.7.

The issues of its application for a geographically widespread environment could be resolved by using the gUDT protocol. At SC07, the geographical distance was more than 8,000 km, from Osaka, Japan, to Reno, Nevada, USA. The RTT between them was about 157 ms through JGN2, APAN, TransPAC2, Internet2, and SCinet. The experiment for comparing gUDT with TCP (not tuned) was performed in the condition that there was no other traffic on the network. In this experiment, the CT volume data (256 Mb) was transferred from SRB storage in Osaka to SRB storage in Reno. With gUDT, it took 10.95 s (S.D. 1.76 s). On the other hand, with TCP, it took 12 min and 53 s (S.D. 36 s) (Fig. 25.8).

The tiled display wall (TDW) consisted of 12 display panels (3×4). The TDW middleware was the scalable adaptive graphics environment (SAGE). In order to use a TDW, fsManager of SAGE should be installed into a representative node. Other nodes should have SAGE receiver. AVS installed as a component of RDVS had the SAGE Application Interface Library (SAIL) module [8]. Hence, the RDVS portlet could export the image to the fsManager. The output resolution of image could be modified upon user request.

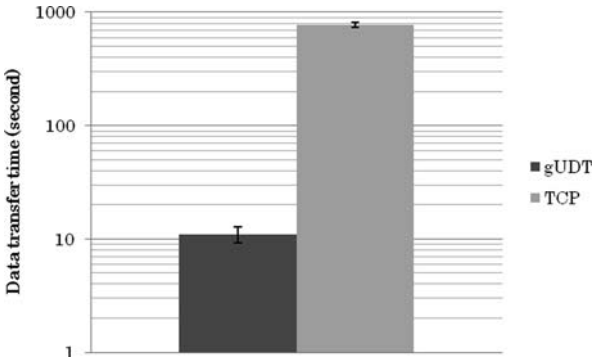


Fig. 25.8 Data transfer result at SC07

The visualized tensor data was displayed on 2×2 TDW at SC2007 and on 3×4 TDW at the JGN2 symposium in Japan (Fig. 25.9). This shows the velocity contour of an oral airflow and proves that highest air-flow speed is detected around teeth. The Web portal system worked correctly to retrieve the several kinds of volumetric data from the remote site. The reason the large-scale data needed to be transferred from the remote site to the local site was to maintain the interactivity for users. Without e-science middleware, it was difficult to achieve such kind of data management.

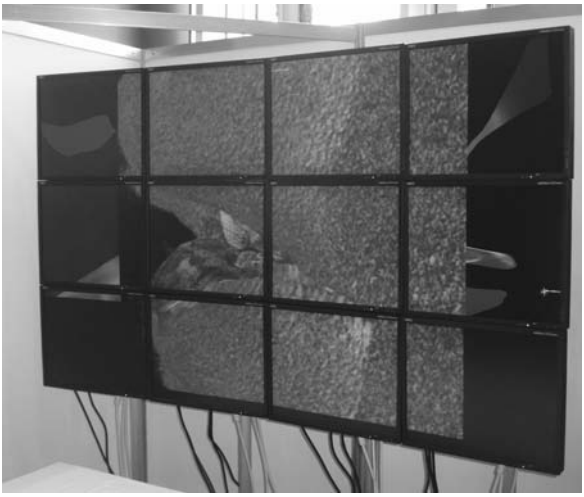


Fig. 25.9 Tiled display wall at JGN2 Symposium 2008

5 Conclusions

We achieved the first phase of the e-science infrastructure in a wide distributed network for the promotion of speech science through easy integrated simulation and the sharing of large data sets. In this phase, two key components were developed: a scientific tool for speech treatments with CFD and large-scale data-sharing technologies for wide-area networks. These key components enable users to visualize large-scale data sets and display high-resolution images on tiled display walls.

Acknowledgements We give special thanks to Hiroaki Yuasa, Toshihito Ito, Atsuji Ogasa, and Kiyota Sakamoto of Fujitsu Corp., Kenichiro Uchida of KGT Inc. who helped to develop the e-science infrastructure, Hiroo Tamagawa, Jiro Miuta of Osaka University Dental Hospital who gave us the chance to take CT image, Xavier Grandchamp, Annemie van Hirtum, and Xavier Pelorson of Grenoble Images Parole Signal Automatique GIPSA Lab, who are collaborative researchers in the speech science, and Jin Tanaka and Takatoshi Ikeda of APAN, who supported us to use the international network in SC07.

References

1. J. Alameda, M. Christie, G. Fox, J. Futrelle, D.Gannon, M. Hategan, G. Kandaswamy, G. vonLaszewski, A.M. Nacar, M. Pierce, E. Roberts, C. Severance, and M. Thomas. The Open Grid Computing Environments collaboration: Portlets and services for science gateways. *Concurrency and Computation: Practice and Experience*, 19:921–942, 2007.
2. C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC Storage Resource Broker. In *Proceedings of the CASCON'98 Conference*, Toronto, Canada, 1998.
3. R. Bondarescu, G. Allen, G. Daues, I. Kelly, M. Russell, E. Seidel, J. Shalf, and M. Tobias. The astrophysics simulation collaboratory portal: A framework for effective distributed research. *FGCS*, 21:259–270, 2005.
4. I. Foster and C. Kesselman. *The Grid 2nd Edition*. Morgan Kaufmann Publishers, Nov. 2003. 1558609334.
5. G. Heydecke, D.H. McFarland, J.S. Feine, and J.P. Lund. Speech with maxillary implant prostheses: Ratings of articulation. *Journal of Dental Research*, 83(3):236–40, 2004.
6. M.J. Lighthill. On sound generation aerodynamically I. General theory. *Proceedings of the Royal Society of London, Series A*, 211:564–587, 1952.
7. M. Lin and W.D. Walker. GECEM: A portal-based Grid application for computational electromagnetic. *FGCS*, 24:66–72, 2008.
8. R. Luc, R. Arun, S. Rajvikram, B. Jeong, K. Naveen, V. Venkatram, C. Vaidya, S. Nicholas, S. Allan, Z. Charles, G. Gideon, L. Jason, and J. Andrew. SAGE: The Scalable Adaptive Graphics Environment. In *Proceedings of WACE*, Nice, France, 2004.
9. J. Novotny, M. Russel, and O. Wehrens. Gridsphere: A portal framework for building collaborations. *Concurrency and Computation: Practice and Experience*, 16:503–513, 2004.
10. K. Nozaki, T. Akiyama, S. Maeda, H. Tamagawa, and S. Shimojo. Integration of computational fluid dynamics and computational aero acoustics simulations on Grid system for dental applications. In *Proceedings of IEEE CBMS*, Dublin, Ireland, pp. 1063–1071, 2005.
11. K. Nozaki, T. Akiyama, H. Tamagawa, S. Kato, Y. Mizuno-Matsumoto, M. Nakagawa, Y. Maeda, and S. Shimojo. The first grid for the oral and maxillofacial region and its application for speech analysis. *Methods of Information in Medicine*, 44(2):253–256, 2005.
12. K. Nozaki, M. Noro, M. Nakagawa, S. Date, K. Baba, S. Peltier, T. Kawaguchi, T. Akiyama, H. Tamagawa, Y. Tanaka, and S. Shimojo. Computational oral and speech science on e-science

infrastructures. In *Proceeding of the 2006 ACM/IEEE Conference on Supercomputing*, Tampa, FL, USA, vol. 298, 2006.
13. D.K. Ray and R.D. Charles. *The acoustic analysis of speech*. Singular Thomson Learning, Canada, 2nd edition, 2002. ISBN-10 0769301126.
14. H. Sakamoto. Data grid deployment for high energy physics in Japan. *Computer Physics Communications*, 177:239–242, 2007.
15. C.H. Shadle. Articulatory–acoustic relationships in fricative consonants. In W.J. Hardcastle and A. Marchal, editors, *Speech Production and Speech Modeling*, pp. 187–209. Kluwer Academic Publishers, 1990.
16. N.K. Stevens. *Acoustic Phonetics*. MIT Press, 1998. ISBN 026219404.
17. O. Tatebe,Y. Morita, S. Matsuoka, N. Soda, and S. Sekiguchi. Grid datafarm architecture for petascale data intensive computing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, Berlin, Germany, pp. 102–110, 2002.
18. C. Wagner, T. Hüttl, and P. Sagaut. *Large-Eddy Simulation for Acoustics*. Cambridge Aerospace Series, Cambridge University Press, 2007. ISBN 9780521871440.

Chapter 26
SynchroNet

High Performance Network Synchronization System

E. Varriale, D. Cretoni, F. Gottifredi, and M. Gotta

Abstract A central issue in designing complex networked systems for critical applicative domains is the possibility of keeping each node of the network synchronized with respect to a given reference system time scale. The problem is even more critical when the synchronization accuracy determines directly the performance of a whole system or service. In this chapter a distributed synchronization infrastructure is proposed providing:

– high accuracy synchronization performance (nano and pico seconds accuracy)
– flexible and scalable service topology (up to global scale)
– ease of integration in pre-existent infrastructures
– ease of customization both at user level and at system level in terms of performance and security

Keywords Synchronization · Network · Performance · Reliability · Continuity · Integrity time · Timing · GNSS · Flexibility · Scalability

1 Introduction

High performance synchronization (nanoseconds and picoseconds level) is a fundamental feature in many applicative domains. In particular, it may be directly bound to

- Positioning, localization, and range-based applications
- Complex monitoring and control systems spanning wide inter-node baselines
- Financial transactions
- Distributed production lines
- Power distribution lines

E. Varriale (✉)
Thales Alenia Space Italia S.p.A., Roma, Italy
e-mail: enrico.varriale@thalesaleniaspace.com

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_26,

- Environmental monitoring systems for SoL (safety of life) and civil protection applications

and, in general, to every, heavily loaded, time-tagging based distributed system.

SynchroNet is an innovative system, patented by Thales Alenia Space Italia, that aims to be the sole solution for every application type that needs a high accurate synchronization covering all levels of performance and coverage requirements, with the added value of the ease of integration also in pre-existent infrastructures and systems. This is possible thanks to a self-contained (both SW and HW, such as GALILEO/GPS Receivers, Atomic Clocks, Meteo Stations, and communication systems) node-based distributed architecture exporting its service through flexible, standard oriented interfaces.

Therefore, SynchroNet is able to cover, with a complete, flexible, and high performance product, the synchronization needs of a wide range market for any application domain. The aim of this chapter is to present the solution patented by Thales Alenia Space Italia, SynchroNet, with focus on how the distributed synchronization improve the system performance.

For additional information the reader can refer also to [1, 2, 3, 4, 6, 7, 8].

2 GNSS-Based Synchronization

In metrology, the synchronization of two clocks is the process required to determine the relative behavior of one clock with respect to the other. In particular, clock synchronization can be defined as the process required to compute at least two parameters: time offset and time drift.

Time offset is the relative, instantaneous, difference between the two time scales, while time drift is the divergence trend of the two clocks. In terms of phase and frequency, time offset is given by the phase offset of two oscillators and time drift is represented by the frequency offset.

It should be noted that if the relative behavior of the two clocks is known, then they can be considered synchronized even if no physical adjustment is carried out.

While synchronizing two co-located clocks could be as easy as measuring 1 PPS (pulse per second) offsets over time by means of an accurate time interval counter (TIC), synchronizing clocks several kilometers away could be much more difficult. To solve the problem of keeping two clocks synchronized even across long baselines with nanosecond accuracy and precision performance, GNSS (Global Navigation Satellite System)-based techniques have been preferred over other methods for many reasons, among which

- Availability of the GNSS navigation signals
- Performance
- Diagnostic information about GNSS system health

GNSS-based synchronization could be carried out by means of the common view technique, which exploits navigation signals broadcasted by GNSS SVs (space

vehicles) that are in a condition of "common view" from the observation points of the two sites to synchronize.

A schematic representation of this method and the related process is given in Fig. 26.1.

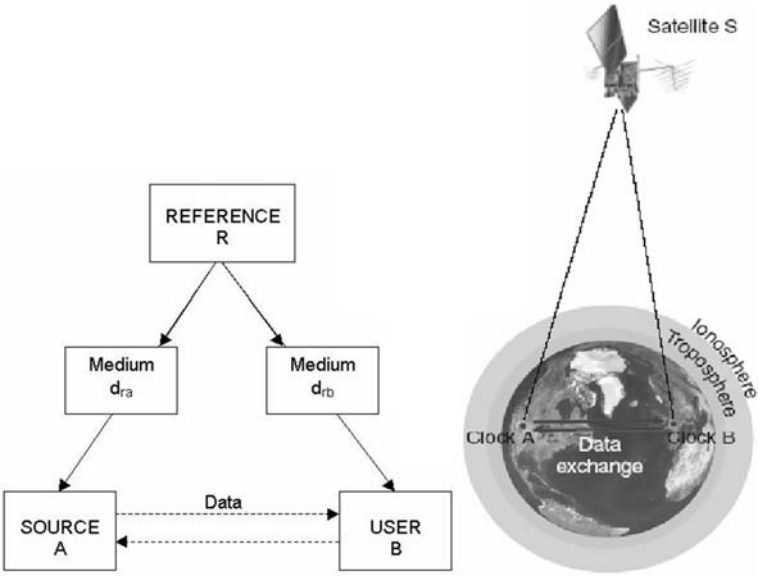


Fig. 26.1 Common view synchronization technique

The common view method is very simple in its theoretical formulation, but requires that a number of issues are carefully assessed and resolved.

From the sketch above, at least two such issues can be identified:

- Delay compensation
- Data exchange

Delays to be compensated are mainly due to the propagation medium through which the navigation signals are broadcasted; the medium can be subdivided into two segments: ionosphere and troposphere. For each segment a delay factor is computed using mathematical models; the adoption of models, derived by statistical data analysis, implies that these delays will not be able to cancel the whole delay. The residuals of delay corrections are among the limiting factors for the common view synchronization performance.

While medium delays affect only the synchronization computation, the data exchange affects, also, other aspects that system designers have to take into account

- Data acquisition process methodology
- Data types and formats
- Data and identity spoofing
- Data preprocessing and quality assessment
- HW equipment status monitoring

If the system is not solely oriented to late post-processing computation and minimum service availability is required, also the following aspects should be considered:

- Data acquisition and transfer scheduling
- Runtime configuration changes
- Risk analysis and fault recovery
- Monitoring of the processes
- Monitoring of performances

3 SynchroNet Overview

SynchroNet is a distributed approach to synchronization with centralized monitoring and control facilities (see Fig. 26.2).

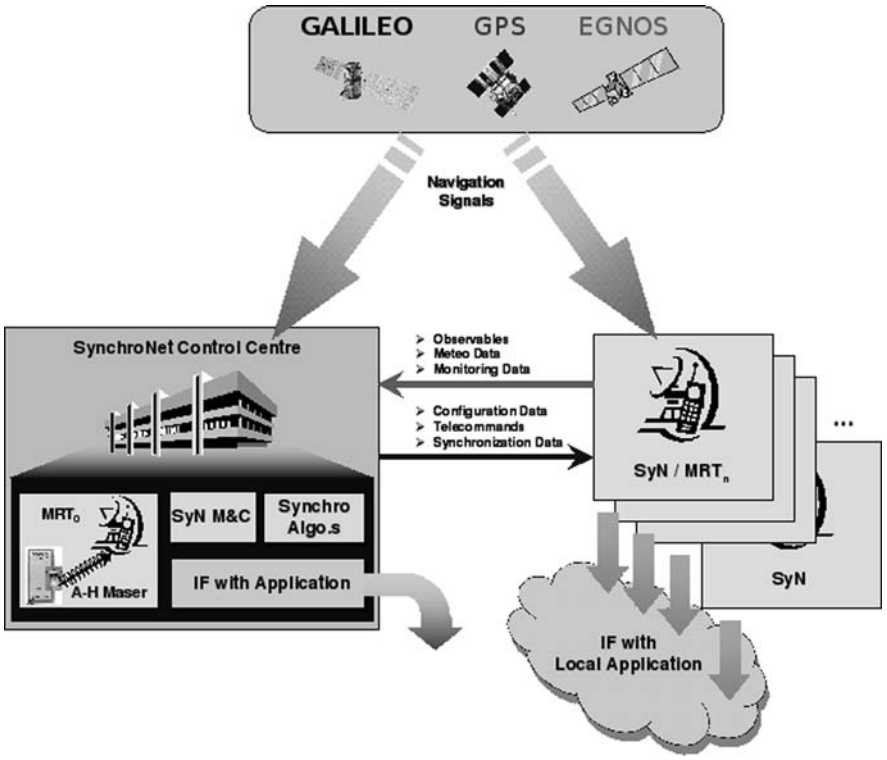


Fig. 26.2 SynchroNet high level architecture

To provide a synchronization system that deals with and solves all the issues described in the previous section, SynchroNet implements a networked infrastructure around the core time transfer algorithms and distributes the synchronization

process over a hierarchical network with hierarchical network nodes' roles. This approach allows to distribute and keep balanced the processing load and limits the propagation of failures.

In the SynchroNet system three kinds of nodes are present:

- The control center
- The MRT (master reference time)
- The SyN (synchronized node)

Fig. 26.3 gives a sample SynchroNet network topology overview involving each kind of node.

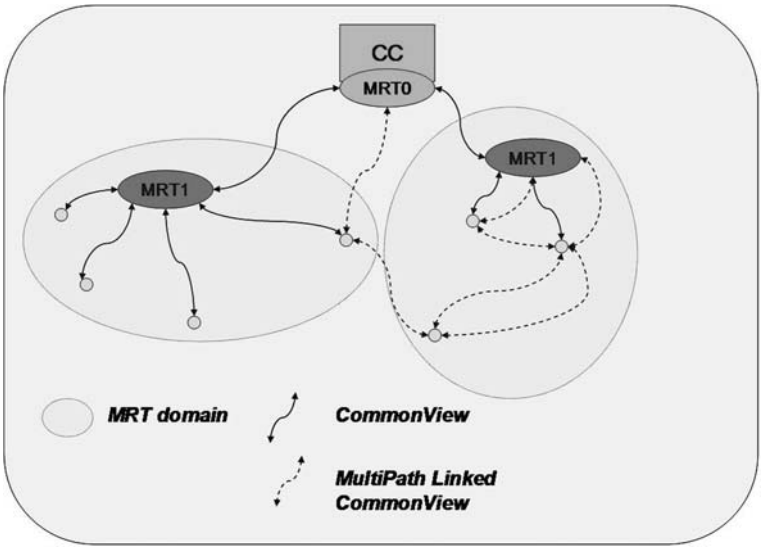


Fig. 26.3 SynchroNet network topology

The *control center* (CC) provides centralized monitoring of each node of the network in terms of overall health status, connection link and security, and synchronization performance. The CC is also responsible for network management allowing the supervisor to change the network topology (add a new node, remove a node, and assign a node to a different MRT) and reconfigure per node or network-wide parameters (acquisition periods, performance thresholds, etc.).

MRTs are the distributed time references and form the hierarchic synchronization backbone of the system. Each MRT computes synchronization of the allocated SyNs (and lower level MRTs) and receives the synchronization parameters from higher level MRTs

The logical root of the network graph is the MRT0, which runs the main reference time and is able to guarantee alignment to international time scales (UTC, TAI, GPS, and GALILEO, ...). Each node is synchronized with respect to this system time.

SyNs are the functional leaves of the graph representing the SynchroNet network; they collect GNSS observables and send them to the controlling MRT, which, in turn, computes and returns the synchronization parameters and the performance status information. The SyNs are, computationally, passive elements for what concerns synchronization.

SynchroNet is characterized, at each level, by a strong modularity of its components, in particular this is true also at node level.

Each SynchroNet node can be viewed as a functional logical entity composed of many sub-modules interacting with each other by means of well-defined interfaces and protocols; this means that a single node deployment may span many physical/virtual machines. The main sub-modules of a SynchroNet node are represented in the diagram of Fig. 26.4.

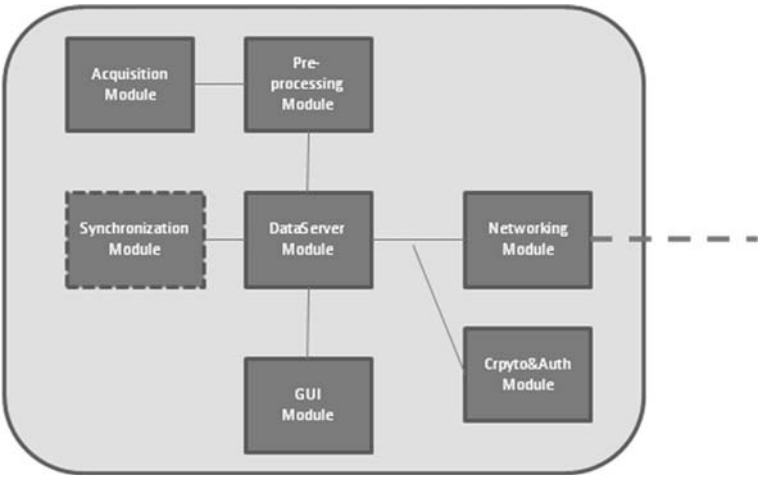


Fig. 26.4 SynchroNet node sub-modules (only MRT nodes have a synchronization module)

Each sub-module is well defined in terms of pre- and post-conditions and follows the principle of OOD (object oriented design) providing information hiding and interface-based asynchronous and concurrent access to the implemented service.

Furthermore, the data server module provides a subscription-based data delivery service able to route data both externally (through the Networking Module) and internally (with respect to the node context) allowing the sub-modules to physically reside on different machines on the same LAN/VLAN.

Inter-nodes data exchange is guarded by Networking Module, which applies a second, packet level, encryption, and crypto signature to the outbound data before routing information through the SynchroNet VPN.

Additionally, each node provides synchronization interfaces by exporting synchronization products in many different ways, for example,

- Exporting files describing computed clock models and integrity statistics
- Outputting corrected PPS and 10 MHz signals by means of a pico-stepper.

These products are exposed through standard interfaces (i.e., standard analog frequency and PPS signals, filesystem objects, or TCP/IP-based connections); this approach is consistent with the modular nature of the system and allows SynchroNet to be regarded as a higher level service providing black box, entirely defined by its interfaces. This design allows the easiest integration of SynchroNet in pre-existent infrastructures and permits effective maintenance cycles.

4 Distributed Synchronization

The choice to design a distributed common view-based synchronization system has been driven by the following considerations:

- Algorithmic efficiency
- Synchronization performance
- System and service scalability
- System and service robustness
- Nodes' hardware costs and upgrades' scheduling.

In particular one of the main design goals was to have a system that could be able to scale to an arbitrary service coverage area without requiring recurrent upgrades due to the increased load of servers and network equipment and that could cope with some limitations of the common view and linked common view synchronization techniques.

By definition, the common view technique effectiveness is determined by the ability of two base stations (actors in the synchronization process) to simultaneously receive the navigation signals from the same sub-set of space vehicles. In general, the higher the number of satellites in common view the lower the incidence of measurement noise on the synchronization values. By running both short-term and long-term performance analyses of common view algorithms exploiting the IGS (International GNSS service) network, the minimum optimal number of satellites in common view was found to be five (Figs. 26.5 and 26.6). Again processing GPS almanac data and cross correlating with IGS stations' network a direct common view synchronization was found to be possible along maximum baselines of 6,000 km (given the requirement of at least five satellites in common view).

For longer baselines the LCVTT (linked common view time transfer) approach should be applied. A generalization of the LCVTT technique is represented in Fig. 26.7.

Synchronization between stations A and C is carried out by computing

$$\Delta T(A,C) = \Delta T(A,B) - \Delta T(B,C)$$

where each ΔT is computed by means of the direct common view method described earlier. This technique can be, in theory, iterated for chains of arbitrary length; in practice, error propagation due to satellite pseudo-range correction algorithms residuals (iono and tropo factors), equipment noise, multi-path, EMI (electromagnetic

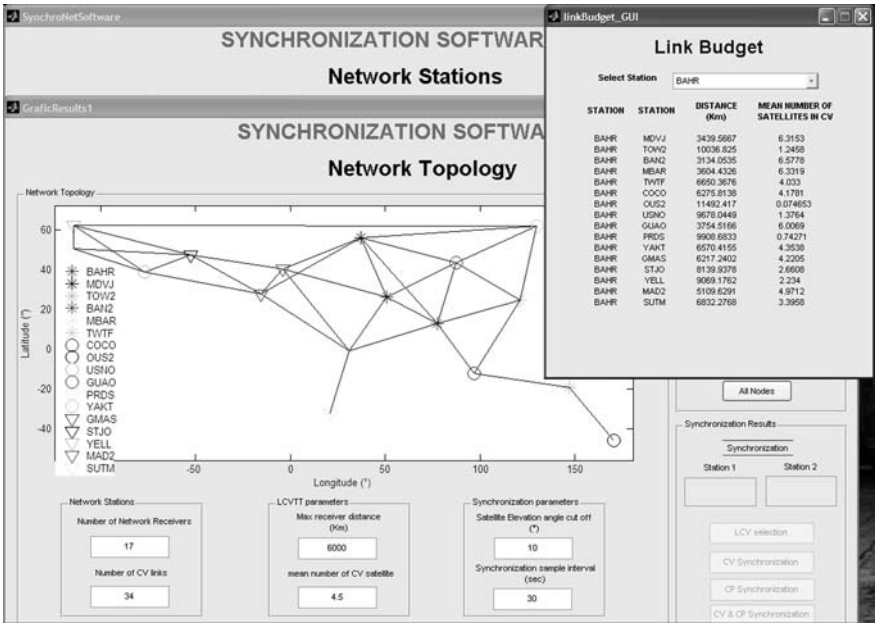


Fig. 26.5 Preliminary analysis tool for optimal baseline determination

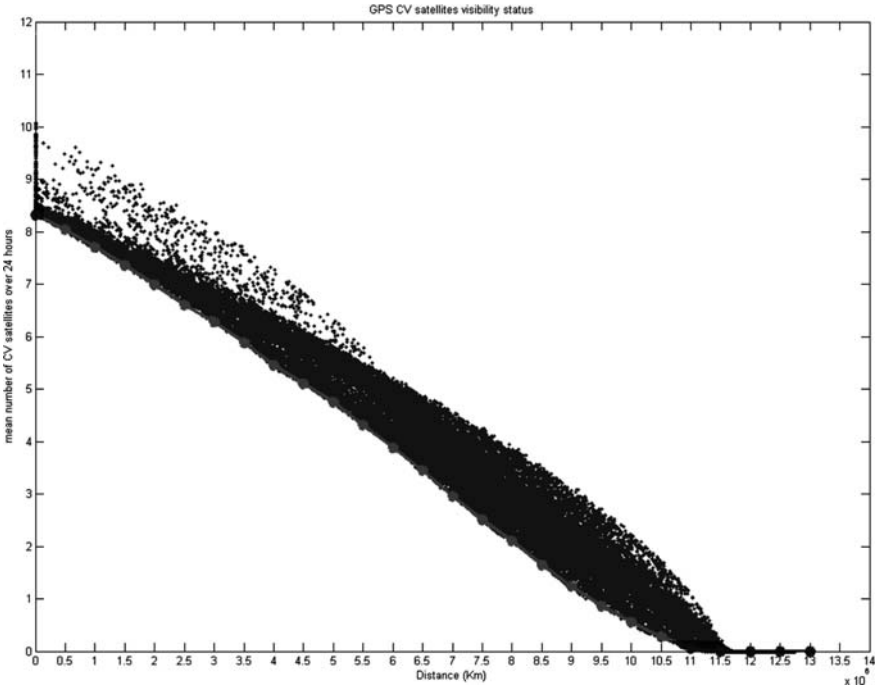


Fig. 26.6 Number of satellites in common view for different baselines

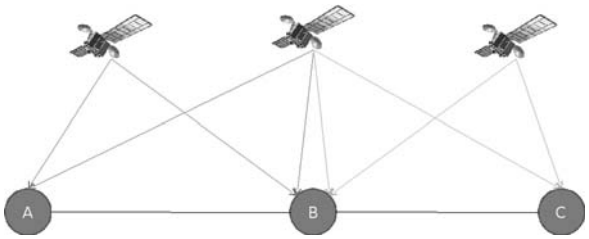


Fig. 26.7 LCVTT technique

interference), and other sources of errors that cannot be fully compensated by heuristics and models, limit the length of synchronization chains to which the LCVTT method can be applied. The next step strategy in order to extend GNSS-based synchronization reach is the MPLCVTT (multi-path linked common view time transfer) technique. Stations taking part in the synchronization process can be represented as nodes of a dynamic graph $G(T) = (V, E(T))$, where V is the set of nodes (i.e., the observing stations), E is a set of tuples (r, s), where $r, s \in V$, and T is a time interval ($T = (t1, t2)$). For each given observation interval T, and for a given pair (r, s), $(r, s) \in E$ if there are satellites in common view between stations A and B during time frame T (common view gaps tolerance in observations is a parameter that should be assessed taking into account the interval length, the sampling interval, the gap length, etc.).

Using IGS network products it is already possible to synchronize, in post-processing, a collection of additional observing stations using GNSS common view (or linked and multi-path linked common view). While IGS provides a worthy source for, among others, research and algorithm validation, there are both functional and performance limitations in using IGS for synchronization.

In particular

- No real-time synchronization is possible.
- No access to configuration and status parameters of each node (a node that is currently down should not be included in a multi-path or linked common view synchronization schedule).
- Each node should compute the whole synchronizations by processing the whole synchronization chain data.
- Data coming from the IGS network cannot be used for applications running under critical infrastructures domain requirements (security, authenticity, availability, etc.).
- No integrity algorithm applicable (for which a maximum TTA (TimeToAlert) should be known in advance and guaranteed to be compliant with fixed requirements).

SynchroNet design solves many of those issues and introduces some new features in terms of functionalities, by providing its own synchronization infrastructure, which

provides a near real-time synchronization service on top of a secured, runtime reconfigurable network.

The first improvement given by wrapping synchronization into a dedicated homogenous network is the ability to distribute the computational process. Each node (which is guaranteed to be, at nominal conditions, not farther than 6,000 km from its direct reference station) sends collected GNSS data to its reference node (MRT) and receives back synchronization products (clock model, synchronization noise, synchronization propagation errors, stability statistics, and a summarizing synchronization integrity indicator). The synchronization loop is depicted in Fig. 26.8 for a three-level depth network.

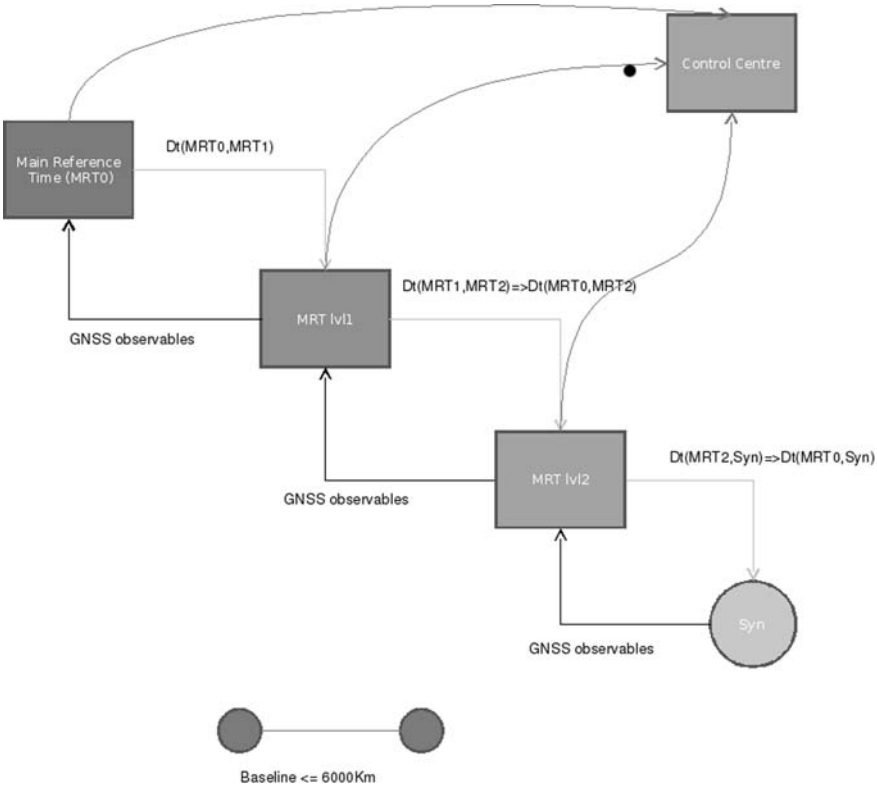


Fig. 26.8 SynchroNet distributed synchronization process

Each MRT is in charge of computing the synchronization only of lower level SyNs and MRTs and their detailed monitoring parameters; furthermore, a concise status report is sent to the control center. This architecture allows a higher level of customization both of the network topology and of the synchronization process.

The network, by means of data meta-routing (implemented at the data server module), which allows information being routed to multiple destination, may have,

for example, multiple control centers (a control hierarchy could be implemented to face network growth, just as paging and indirect indexing methods work for computers operating systems).

This flexibility allows a different approach to multi-path linked common view that can be realized by sending observables data to different, higher level MRTs. When nodes join the network the CC defines the path set for MPLCVTT by knowing the detailed status of each node already in the network and their performance; paths can be recomputed when events happen: nodes moved, added, removed, and status change (synchronization performance, link, observables quality, etc.).

From the synchronization algorithms point of view, customization can be achieved by integrating different oscillators, which in turn may require different synchronization parameters, such as synchronization window (the time over which GNSS observables are collected) and integrity thresholds; also different and customized observation correction models (i.e., ionosphere and troposphere) may be required for specific locations of nodes.

SynchroNet not only provides a configurable and customizable synchronization network but also allows a runtime automated or semi-automated network reconfiguration capability. In particular, SynchroNet can take automatic actions in case an MRT goes down by reallocating the whole branch under the failing node. An example is given in Fig. 26.9.

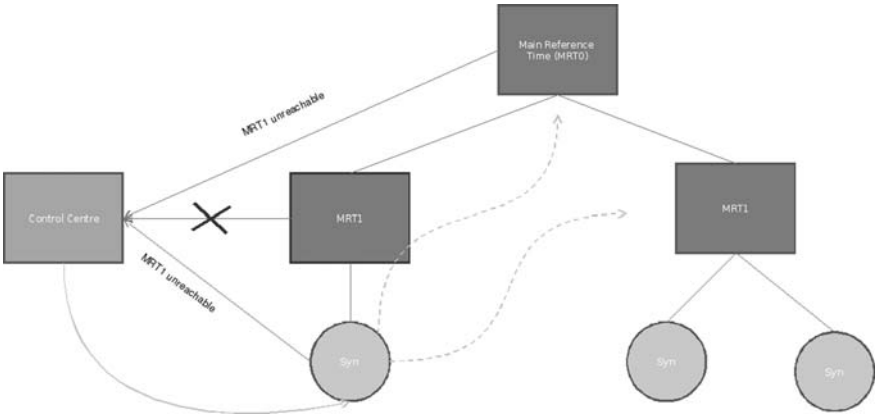


Fig. 26.9 SynchroNet runtime link fault recovery with SyN reallocation

Reallocation requires the following steps for each of the children of the failing MRT:

- Choice of another MRT by evaluating both baselines and MRT load. If a direct Common View cannot be obtained due to the baseline length between the moved node and the new MRT, then a multi-path linked common view is set up and the synchronization of the node is computed as a means of the synchronizations along each path.

- Cryptographic keys exchange (this may require, depending on the security level, human intervention)
- Networking module configuration
- Data server routing configuration

All the configuration steps are carried out by the main control center which is, in general, the only authority able to modify the network topology (add, move, remove, and fully configure a node). In particular, each node is connected to the control center by a dedicated VPN, which is separated by the network used for data exchange between other network nodes.

A solution to make recursive load/baseline balancing and optimization in the case of node movement is currently a research topic. While a recursive optimization in a tree-like network topology would take at most a logarithmic time, it is not easy to model exactly the complexity of such a change in the case of a mesh network layout, where the data exchange network, the synchronization network, and the control network overlap.

5 Implementation and Test Results

SynchroNet is a test-driven project in many aspects: code development, algorithm validation, and performance assessment.

The main issue was the validation of core synchronization algorithms in their many components: ionospheric and tropospheric models implementation, code (i.e., pseudo-range: CA and P code)-based common view, phase-based common view (L1 and L2), clock stability characterization by means of CV, and comparison to direct synchronization in lab (PPS biases analysis).

IGS again was selected as a test bench for algorithms validation, thanks to the wide set of products available allowing a significant statistical analysis.

Hereafter (Figs. 26.10, 26.11, and 26.12) some results are presented of both validation test results and of performance assessment of SynchroNet algorithms.

SynchroNet is at v1.0 verification phase, but its kernel is already running providing core synchronization service for the GALILEO test range (GTR), an advanced research facility for the experimentation and analysis of the Galileo signal, for testing and certification of user terminals and support services for the development of application services. In the frame of the GTR project, SynchroNet is used to synchronize pseudo-satellites OCXO (over-controlled crystal oscillator)-driven Rubidium clocks used for ranging measurements with a requirement of 5 ns ($\sim$1.5 m) 1σ accuracy. In this context the main reference time (MRT0) is a free running active hydrogen maser atomic clock (located at the time laboratory facility (TLF) of the GTR building), while control center features are customized and integrated in the control center facility of the GTR and in the monitoring facilities of the TLF.

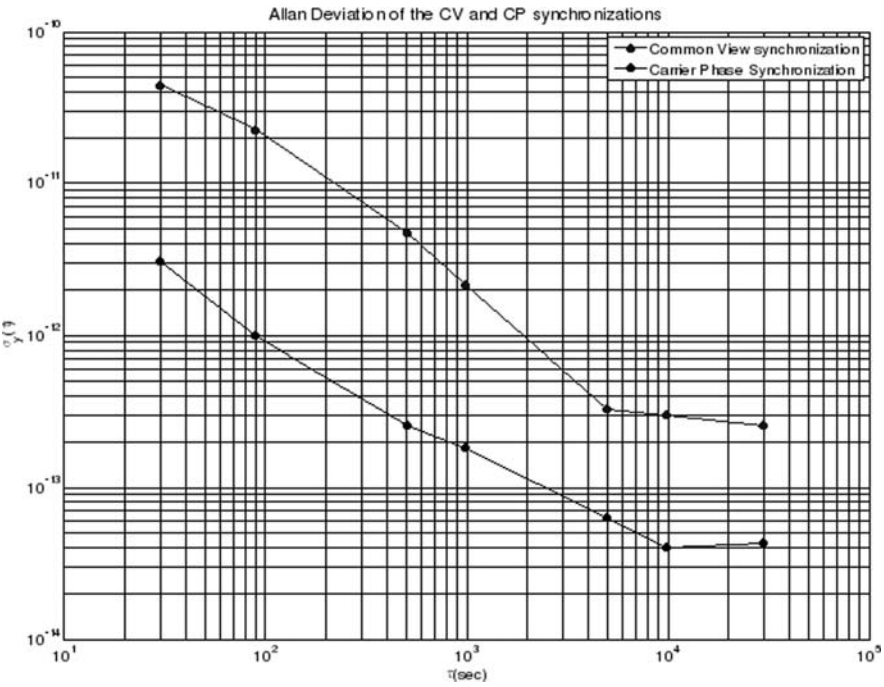


Fig. 26.10 Allan Deviation [5] computed for code and carrier phase common view synchronization shows the improvement in frequency stability performance of carrier phase common view. Data relative to an active hydrogen maser atomic oscillator

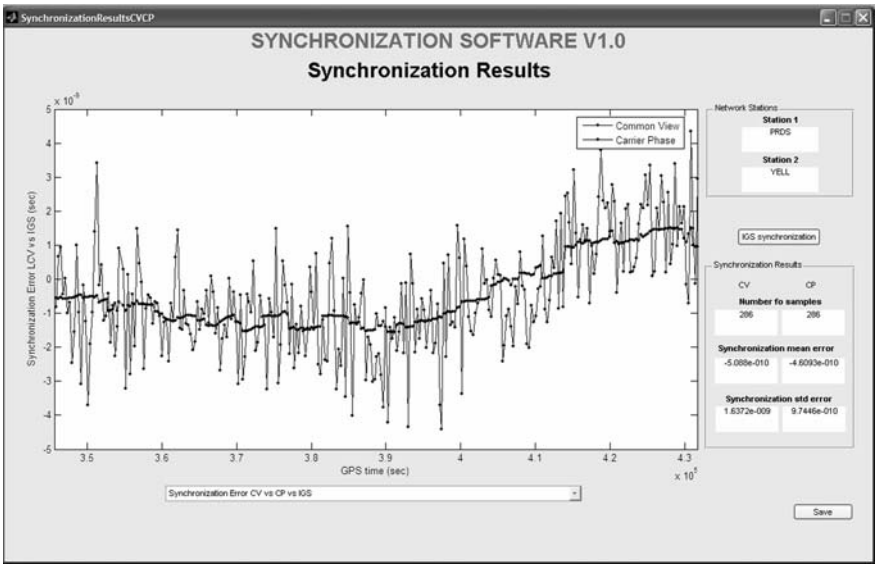


Fig. 26.11 Comparison of SynchroNet Code and Carrier Phase-Linked Common View algorithms results with respect to IGS products

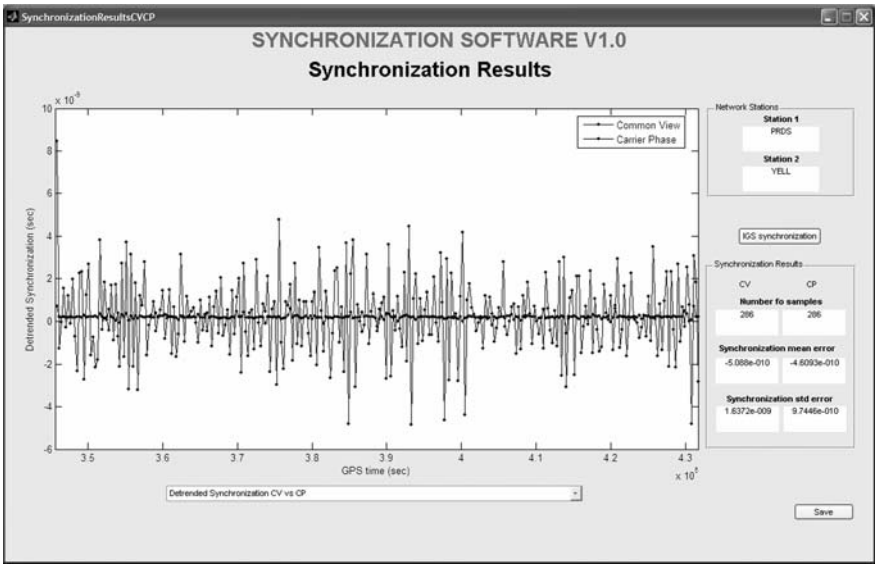


Fig. 26.12 An integrity check, after a synchronization model has been computed (by quadratic interpolation), the model is applied and residuals are analyzed in order to detect residual drifts

6 Conclusions

In this chapter a distributed approach to GNSS-based time and frequency transfer has been presented, which allows to exploit high accuracy (nanoseconds and picoseconds) synchronization of high performance atomic clocks along large baselines, by wrapping the synchronization service in a robust and flexible infrastructure providing security and scalability features.

Through separate and successful validation campaigns using the IGS network and in a full deployed system (the GALILEO Test Range), SynchroNet proved its customizability and performance.

Currently, SynchroNet is going through a further design and research phase in order to consolidate, expand, and improve its features and infrastructure to match critical systems requirements, in order to become a high performance solution for every applicative domain that requires synchronization as a fundamental service.

References

1. D.W. Allan and M.A. Weiss. Accurate time and frequency transfer during common-view of a GPS Satellite. In *Thirty fourth Annual Frequency Control Symposium*, pp. 334–346, May 1980.
2. F. Gottifredi, F. Martinino, Q. Morante, M. Eleuteri, E. Varriale, V. Valle, and G. Pesci. Galileo Test Range Performance test results. In *ION 2008*, San Diego, CA, USA.
3. K.M. Larson and J. Levine. Carrier-phase time transfer. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 46, pp. 1001–1012, Jul. 1999.

4. M.A. Lombardi, L.M. Nelson, A.N. Novick, and V.S. Zhang. Time and frequency measurements using the Global Positioning System. Cal Lab: International Journal of Metrology, 8:26–33, Jul.–Sep. 2001.
5. D.B. Sullivan, D.W. Allan, D.A. Howe, and F.L. Walls. Characterization of clocks and oscillators. NASA STI/Recon Technical Report N, 91:22539–+, Mar. 1990. http://adsabs.harvard.edu/abs/1990STIN. . .9122539S.
6. D.B. Sullivan, D.W. Allan, D.A. Howe, and F.L. Walls. Characterization of Clocks and Oscillators. Technical Report Technical Note 1337, NIST, Mar. 1990.
7. E. Varriale, M. Gotta, F. Gottifredi, and F. Lo Zito. Common-view technique application: An Italian use case. In *Tyrrhenian International Workshop on Digital Communication*, Ponza, Italy, 2006.
8. V.S. Zhang, T.E. Parker, M.A. Weiss, and F.M. Vannicola. Multi-channel GPS/GLONASS common-view between NIST and USNO. In *IEEE International Frequency Control Symposium*, pp. 598–606, Jun. 2000.

Chapter 27
Discovery of Resources in a Distributed Grid Environment Based on Specific Service Level Agreements (SLAs)

D. Kollia, S. Kafetzoglou, M. Grammatikou, and S. Papavassiliou

Abstract In distributed environments it is necessary to provide tools that describe and guarantee the level of Quality of Service of the provided services. These guarantees can be specified in terms of Service Level Agreements (SLAs), which in turn can be used for the management and monitoring of Quality of Service attributes. The provided services supply the appropriate interfaces through which the service providers can allocate and manage their resources. The emphasis of this work is placed on the discovery of resources in a distributed environment, based on Quality of Service attributes, which can be expressed in terms of SLAs between the service provider and the client. For this purpose a registry that supports dynamic registration and contextualized discovery of resources is used. GRIA software platform has been adopted as the basis for the provisioning of the resources to be managed and the design of the proposed SLA-based resource discovery mechanism, as it provides a service-oriented infrastructure, designed to support service provisioning in a secure, interoperable, and flexible manner.

Keywords Service Level Agreement (SLA) · Grid · Distributed environments · Web services · QoS · Contextualized discovery · Registry

1 Introduction

Grid computing has emerged as a paradigm of sharing resources for better collaboration and resource usage optimization purposes. A Grid consists of a finite set of nodes, where a node may be a single system or a cluster that can manage a set of resources. In principle, a resource being managed could be a network, a system, an application, storage space, CPU, or another piece of equipment. Grid computing has traditionally relied on a "best-effort" service paradigm; however, over the last few years this mode of operation has been proven insufficient to meet the users' needs

D. Kollia (✉)
Network Management and Optimal Design Lab. (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece
e-mail: dimkollia@netmode.ntua.gr

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_27,

and requirements, especially in commercial Grids [5]. Guarantees on the reservation and availability of resources should be provided to users who should be able to ask for a specific level of quality and availability, according to the application requirements and constraints.

Moreover, the extensive use of distributed environments in commercial and business applications motivates the need for developing appropriate enhanced monitoring tools that would guarantee the demanded Quality of Service. Given the fact that Grid technology is moving toward web services' technology, the corresponding guarantees and the monitoring of various Quality of Service attributes should be provided in terms of Service Level Agreements – SLAs [5]. An SLA is a contract between the service provider and the user, where the service provider can agree with the client on the level of availability and efficiency of the provided resources, the operation mode and functionality, the provided safety level, and the required delay times. In addition, charges for the resource usage and penalties in case of violation of the SLA terms can also be specified in an SLA. As mentioned earlier, potential resources include, but are not limited to, CPU, bandwidth, storage space, delay times, and any other entities that may require provisioning in a distributed environment.

As a result, SLAs can facilitate and support the development of QoS-aware discovery of resources in distributed environments. QoS-aware discovery of resources is mandated in an environment where multiple service providers provide the same resources, but at different level of Quality of Service. For example, in a multi-provider diverse distributed computing environment it is expected that more than one service providers could offer CPU usage; however, each one of them would provide different delay and response times. Consequently, users should be able to optimize their selection, by searching for the service provider satisfying the desired level of Quality of Service according to their needs and the respective charges. Therefore, Quality of Service attributes could be specified in terms of an SLA binding the client application with the required resources. Clients are then able to search for the SLAs which are mapped to the service providing the required resources.

1.1 Background Information and Objectives

The issue of efficiently discovering resources in a distributed Grid environment while meeting specific users' requirements is of high research and practical importance. In [1] an extension of the service abstraction in the Open Grid Services Architecture for Quality of Service (QoS) properties is presented and the architecture of G-QoSM framework is proposed for GRID service discovery of QoS criteria, by extending WSDL and UDDI and combining the use of resource managers. Moreover, in [7] an approach that utilizes Case-Based Reasoning methodology for modeling the web services discovery and matchmaking problem is presented. Finally, in [3] a semantic-based approach for QoS-driven service discovery

is proposed in order to enable the consumers of the services to select the currently optimal services matching their requests in a dynamic GGS (GIS Grid Services). Services are selected according to the functional requirements and dynamic Quality of Services (QoS) requirements.

In our work in this chapter the discovery of services is conducted using contex tualized registries deployed in different Grid sites while QoS attributes are specified in terms of an SLA. Our proposed approach and implementation is based on GRIA Grid middleware [11], and therefore a brief overview of GRIA middleware, its capabilities, and main software packages provided by GRIA are first described. Then, the overall proposed process of contextualized discovery of resources in a distributed environment is discussed in Section 3, while some concluding remarks with respect to the potential use of the proposed methodology are presented in Section 4.

2 The GRIA Grid Middleware

GRIA is a service-oriented infrastructure, designed to support service provisioning in a secure, interoperable, and flexible manner [6]. GRIA uses web service protocols based on key interoperability specifications [11] and provides the appropriate web service interfaces through which service providers can offer their resources.

In the following, for completeness purposes, we briefly describe the main software packages that were used throughout our implementation. Specifically, the Basic Application Services package was initially adopted that includes the Data and the Job Service. The Job Service is the web service interface through which service providers can provide jobs for data processing, whereas the Data Service provides data stagers in a similar way for data storage. Data stagers are resources that can be used for downloading or uploading data files and can be used by job applications. It should be noted that a data stager can contain only one data file.

The Sample Service is a web service created using the GRIA API and is integrated within GRIA's security and management capabilities. Its main functionality is to provide an interface through which resources, called sample resources, can be provided and managed. These resources are generic without a specific function, but they can be specified into any resource that can be managed by such a service.

Service Provider Management package includes the SLA Management Service and the Trade Account Service. Trade Account Service allows a service provider to manage relationships with customers who pay for the resource usage according to the pricing terms of the SLA, called Trade Accounts. A Trade Account represents a trust relationship between the service provider and the customer. This is achieved via the use of a credit limit for each account set by the service provider, which is the level of trustfulness for the specific customer [10].

Finally, the SLA Management Service provides and manages SLAs where service providers define Quality of Service terms satisfied by their services in SLA

offerings, called SLA templates. SLA templates are published through the SLA Service and customers can then select the SLA template that suits their needs and propose it to the service provider. The SLA can then be agreed and the service can be used if the service provider trusts – i.e., has an active Trade Account – the customer, and the requested resource capacity can be offered. SLA Service Management includes the management of the service usage, limiting the usage according to the Quality of Service terms agreed in the SLA and the charges for the resource usage according to the pricing terms of the SLA. Service management is done through usage reports sent by the functional services to the SLA Management Service. Usage reports, constraints, and pricing terms are defined in terms of metrics. Each resource is mapped to a metric, which is defined by the service provider. Metrics are generic and discriminated by a unique URI [10].

3 Contextualized Discovery Implementation

As stated earlier, the purpose of this work is the discovery of resources in the context of SLA templates in a distributed way. The specific implementation of the resource discovery mechanism was performed in a computer network consisting of three computer nodes; however, it can be easily extended to more computer nodes in a similar way. Each computer node has a GRIA site where the following GRIA services are deployed: SLA Management Service, Trade Account Service, Data Service, Job Service, and Sample Service.

A registry is installed in each computer node by the administrator of the GRIA site. For the implementation of the registry we based our work on Registry Component of the GRIA middleware [2]. This registry enables contextualized discovery of resources, using metadata, and is able to cope with any resource as far as they are represented in XML and identified by a WS-Addressing Endpoint Reference (EPR) [4]. It supports dynamic registration and discovery of resources using context information. This context information is defined in a registry domain model of the registry (RDM) in terms of concepts and relationships between these concepts. Relationships define dependencies between concepts and specify hierarchies of concepts where sub-concepts inherit relationships from their parent concepts. The Web Ontology Language (OWL) [12] was used by the administrator for modeling of the RDM. The administrator configures the registry by defining the concepts for SLA templates and the EPRs of Data Service, Job Service, and Sample Service. The Quality of Service attributes provided by each service provider are specified in the SLA templates offered through the SLA Management Service. The SLA templates are linked with the EPRs of the Service that can satisfy their constraints, using a relationship called “canbemanaged,” which has been previously defined in the RDM.

The overall process along with the sequence of events and actions required are illustrated in Fig. 27.1.

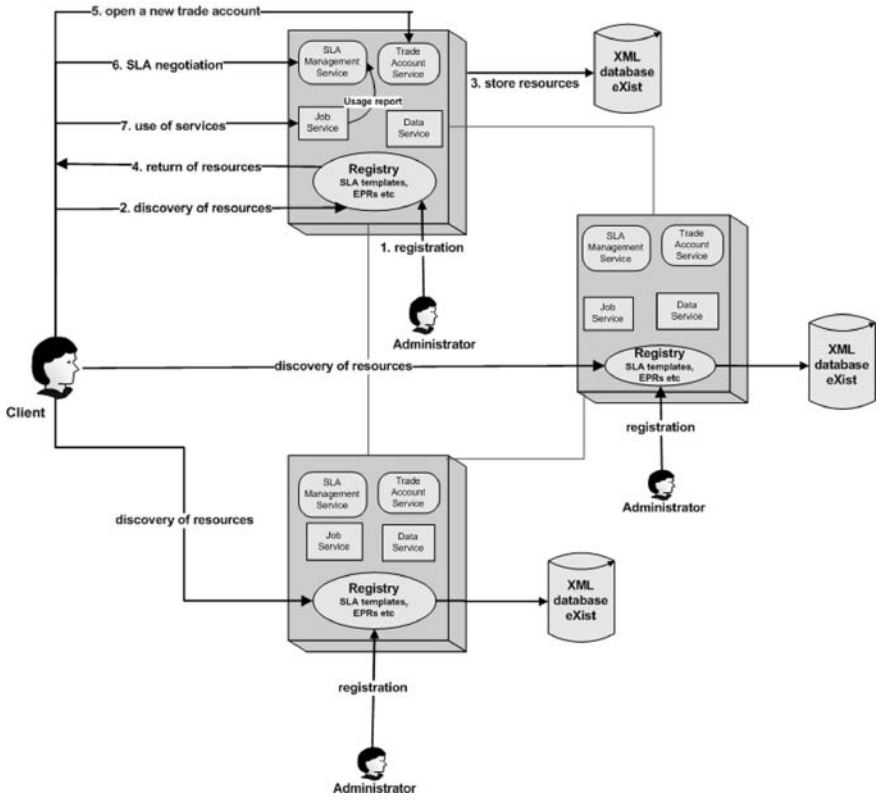


Fig. 27.1 Process of contextualized discovery of resources

Step 1: SLA templates, resources, and EPRs of services are stored into the registry as XML documents under the concepts defined earlier and can be retrieved with respect to these concepts and the relationship "canbemanaged."

Step 2: In the second step the user can search for resources, giving as input the desired Quality of Service attributes in the form of a query, expressed in an object-oriented XML-based query language (ooXmlQL).

Step 3: All the registered resources are stored in an XML database that is built on the open source native XML database eXist [9].

Step 4: If the requested resources are found in this database, they are returned to the user. The returned results include the SLA template in XML form and the EPR of the Service, and they are now available to the user. The Service is managed by the SLA Management Service, so the user has to agree on an SLA in order to use it.

Step 5: The user opens a new Trade Account, if he does not have one, and proposes the SLA template to the service provider of the service.

Step 6: The SLA is agreed if there is sufficient capacity of resources and if the user has enough balance in his account.

Step 7: Finally, the SLA is agreed and the user is now able to use the service (step 7). Usage reports are sent to the SLA Management Service from the functional service to check that the resource usage does not violate the constraints of the SLA.

4 Conclusions

In this chapter the process of contextualized discovery of resources in a distributed environment was discussed and presented. The context information was expressed in terms of concepts and relationships between the concepts. The resources to be discovered were stored into a registry under these concepts by the administrator of a GRIA site. The requested resources were those that could fulfill the constraints and pricing terms specified in SLA templates. This process allows customers to use the services that satisfy their demands with Quality of Service in a distributed environment.

In principle, contextualized discovery could be used for a vast amount of applications where users need resources and services based on specific QoS criteria. One possible example is the use of our proposed mechanism for implementation within the EU Argugrid scenario of oil detection [8]. This way, the user could be able to discover the satellite service that provides images, in the most profitable combination of image quality, response time, and regional coverage of the sea region contaminated by oil. The required extensions include new concepts representing the EPR or the WSDL form of the satellite services. The above Quality of Service attributes offered by the satellite services would be specified in SLA templates as explained in this work.

Acknowledgements We acknowledge partial support of the EU project ArguGRID-IST-035200.

References

1. R.J. Al-Ali, O.F. Rana, D.W. Walker, S. Jha, and S. Sohail. G-QoSM: Grid service discovery using QoS properties. *Computing and Informatics Journal*, 21(4):363–382, 2002. Special Issue on Grid.
2. Contextualized Registry Dev Kit Tutorial. `http://www.gria.org/downloads/download-page`
3. W. Li, S. Zhao, H. Sun, and X. Zhang. Ontology-based QoS driven GIS Grid Service Discovery. In *Proceedings of IEEE 2nd International Conference on Semantics, Knowledge, and Grid (SKG'06)*, Guilin China, Nov. 2006.
4. U. Radetzki, M.J. Boniface, and M. Surridge. Contextualized B2B registries. In *Proceedings of 5th International Conference on Service Oriented Computing (ICSOC 2007)*, Vienna, Austria, Sep. 2007.

5. A. Sahai, S. Graupner, V. Machiraju, and A. van Moorsel. Specifying and monitoring guarantees in commercial grids through SLA. In *Proceedings of 3rd IEEE/ACM International Symposium Cluster Computing and the Grid (CCGrid'03)*, Tokyo, Japan, May 2003.
6. M. Surridge, S. Taylor, D. De Roure, and E. Zaluska. Experiences with GRIA – Industrial applications on a Web Services Grid. In *Proceedings of 1st International Conference on e-Science and Grid Computing (e-Science'05)*, Melbourne, Australia, Dec. 2005.
7. D. Thakker, T. Osman, and D. Al-Dabass. S-CBR: Semantic Case Based Reader for Web services discovery and matchmaking. In *Proceedings of 20th European Conference on Modeling and Simulation (ECMS 2006)*, Bonn, May 2006.
8. The Argugrid Project. `http://www.argugrid.eu/`
9. The eXist Database. `http://exist.sourceforge.net/`
10. The GRIA Architecture. `http://www.gria.org/publications`
11. The GRIA Middleware. `http://www.gria.org`
12. Web Ontology Language (OWL). `http://www.w3org/TR/owl-features/`

Chapter 28
Inter-Domain SLA Enforcement in QoS-Enabled Networks

C. Marinos, V. Pouli, M. Grammatikou, and V. Maglaris

Abstract Traditionally, network Service Providers specify Service Level Agreements (SLAs) in order to guarantee service availability and performance to their customers. However, these SLAs are rather static and span a single provider domain. Thus, they are not applicable to a multi-domain environment (including edge networks of distributed Grid resources) and do not enable users with the flexibility to set up and monitor e2e services. In this chapter we present a framework for dynamic management of SLAs; the framework contains a collection of APIs and modules that allow for the dynamic creation, configuration, and delivery via automated management of an end-to-end SLA created from the merging of the per-domain SLAs. We assume that individual domains are QoS aware, since near real-time applications (e.g., IP voice and video sessions and Grids for remote control of sensors – instruments) may impose stringent performance requirements, even on over-engineered backbone and edge networks.

Keywords Service Level Agreement (SLA) · Multi-domain · SLA Framework · QoS · Inter-domain · Intra-domain

1 Introduction

A Service Level Agreement (SLA) is a contract between a network provider and a customer defining all aspects of the service offered. It may specify the levels of availability, serviceability, performance, and operation conditions [7, 12]. It may also define the procedures and the reports that must be provided to track and ensure compliance with the SLA or describe other attributes of the service, such as billing and penalties in case of SLA violations [2].

C. Marinos (✉)
Network Management and Optimal Design Lab. (NETMODE), National Technical University of Athens, Iroon Polytexniou 9, 15780 Zografou, Greece
e-mail: cmarinos@netmode.ntua.gr

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_28,

In Grid environments [4] where applications or users require resources from heterogeneous domains offering various performance guarantees, the issue of acquiring a specified level of QoS is of great importance. The lack of control in traditional Best Effort IP networks created the need for efficient QoS mechanisms, notably the DiffServ protocols [8]. Marking and policing of flows is implemented in edge routers according to per-domain SLAs. This way we may offer a variety of end-to-end services across multiple, separately administrated domains.

Based on the DiffServ architecture, the Premium IP (PIP) service was implemented in European Research and Education Networks [11]. Using DiffServ priorities, traffic can be classified as high-priority PIP, Best Effort (BE) for standard IP applications, and Less than Best Effort (LBE) for non-time critical bulk data transfer. The Premium IP service provides a virtual leased line service as PIP data packets are not expected to experience congestion in the network regardless of the load of the other traffic classes. It is suitable for near real-time applications, such as Voice over IP (VoIP), video conferencing, and time critical remote control of instruments and devices as demonstrated in the GRIDCC project [5].

In this chapter we present a framework for dynamic management of SLAs, developed for Premium IP e2e (end-to-end) services offered in a multi-domain network. The framework allows for the dynamic creation, configuration, and delivery via automated management of an e2e SLA created by merging SLAs provided by each domain.

Note that although our work concentrated on PIP offered by European Research and Education Networks and their interconnection network GÉANT2, the proposed framework is applicable to any QoS-enabled multi-domain environment.

The rest of the chapter is organized as follows. Section 2 presents some related work; Section 3 describes the SLA Framework along with its core components and its architecture. Sections 4 and 5 deal with SLA management and monitoring, respectively. Section 6 concludes the chapter and presents some future work.

2 Related Work

To date, several frameworks have been proposed to support the creation and management of services based on SLAs. Hoffner et al. [6] present a Contract Framework and Dan et al. [3] propose a schema to support the provision of Web Services to customers at different service levels. The Web Service Management Language (WSML) [10] provides a powerful tool for the formal XML-based specification of custom-made SLAs for Web Services. Such a language has been introduced with the specific aim of allowing the definition of SLAs in a precise and unambiguous way, an essential prerequisite for automating SLA management. Analogies also exist as to the perception of a service from the network perspective. The COPS [1] protocol, standardized by the IETF, proposes a logical approach for dispatching policies throughout the network. On the manufactures' side, CISCO indicates DiffServ as the preferred technology to achieve SLA management and considers delay, jitter,

packet loss, throughput, and availability as the most vital SLA parameters for an IP service.

Related to the above proposals, our framework proposes an automated mechanism for SLA generation, which can be deployed in real QoS-enabled networks.

3 End-to-End SLA Framework

3.1 End-to-End SLA Issues

In principle, an end-to-end SLA has to contain all necessary parameters to provide end-to-end control for a service instance specification between two end points. The e2e SLA should be transparent to the end user, regardless of underlying networks and intermediate domains.

For the generation of an e2e SLA, several issues have to be examined:

Different ISPs must agree to cooperate with each other, authorize and configure the framework's communication between other cooperating ISPs, and specify clearly what actions are taken and by whom.

Depending on the service request type (unidirectional, bidirectional), the SLA Framework will deliver the respective end-to-end SLA. This distinction is necessary since the downstream (closer to the destination) domain may not wish to participate in the e2e marking and policing setup requested by the upstream (closer to the source) domain.

In order to honor the guarantees stipulated in the e2e SLA, we need to continuously monitor, for the whole service duration, all SLA parameters in the contract between the user and the service providers. The e2e monitoring SLA will be based on the investigation and monitoring of all the per-domain SLAs along the path and must not impact on the network performance.

Due to the fact that we are dealing with a multi-domain environment where QoS provisioning is each domain's responsibility, the e2e SLA will be derived from the merging of each domain's SLA, as shown in Fig. 28.1. The merging rules are described in Table 28.1.

3.2 End-to-End SLA Template Description

An SLA is a set of technical and non-technical parameters agreed between the consumer and the provider. An SLA contract can be formed by two different parts: (i) an administrative and legal part that contains information that does not depend on the particular service (e.g., user authentication, availability information, privacy and security aspects) and (ii) the Service Level Specification (SLS) part, which is a set of parameters and their corresponding values that define the technical parameters of the service provisioning for a specific flow instance. While SLAs represent a high-level

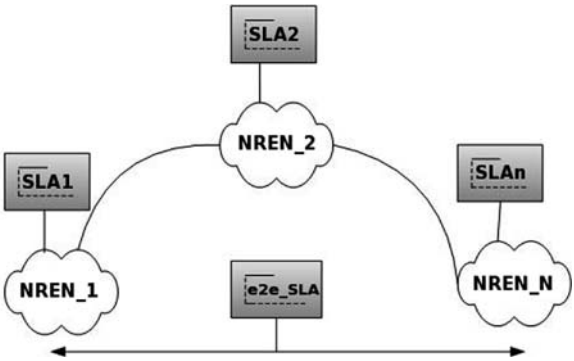


Fig. 28.1 End-to-end SLA template

Table 28.1 SLO part

Parameters	Description
Service instance scope	This field should contain the technical information of the ingress interface and the egress interface of each domain involved in the path
Flow description	This field should contain the DSCP value, the IP source–destination address pair along with the protocol and the ports that the application is targeted to
End-to-end performance guarantees	The SLA performance guarantees field of the end-to-end SLA is proposed to adopt the guarantees derived from the individual SLAs involved in the service provision across the path. More specifically, capacity will be less or equal to the minimum of the capacities in each SLA instance: $C_{e2e} \leq \min\{C_i\}$ MTU value supported by the end-to-end SLA will be less or equal to the minimum MTU value of the chain of individual SLAs: $MTU_{e2e} \leq \min\{MTU_i\}$ One-Way Delay is an additive metric and therefore the corresponding field in the e2e SLA will be produced as follows: $OWD_{e2e} \geq \sum\{OWD_i\}$ $IPDV_{e2e}$: We treat IPDV as an RMS value and use its square as an additive parameter Packet loss: This will be the sum of all the packet losses across the individual domains: $PL_{e2e} \geq \sum\{PL_i\}$
Monitoring infrastructure	This field should contain information on the monitoring capabilities of each domain in terms of which parameters are monitored, the points where measurements are possible, the availability of measurements
Reliability	Reliability should define allowed mean downtime per unit of time for the service provision and maximum allowed time-to-repair (TTR) in case of breakdown for the provision of the e2e service described by the e2e SLA, as the worst figures among the individual SLAs along the end-to-end path

description of a required service, SLSs are more formal documents containing a list of technical parameters.

It is useful to deal with the SLA components as two objects. An Administrative Level Object (ALO) referring to the administrative and legal part and a Service Level Object (SLO) referring to the Service Level Specification. As mentioned before the administrative details contained in the ALO remain the same in each domain for each service request while the information contained in the SLO part is specific for every service instance. The information of the SLO part should contain the fields shown in Table 28.1.

We can represent in an XML format, as shown in Example 28.1, a possible example of the two parts, the ALO and the SLO, along with the information that they contain.

Example 28.1 SLA template

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SLA_Template.dtd>
<SLA>
      <SLA_id>No#</SLA_id>
      <ALO>
            <Administrative_details/>
            <SLA_duration/>
            <Service_availability />
            <Repsonse_times />
            <Fault_handling_procedures />
      </ALO>
      <SLO>
            <ServiceScope>
                  <IngressInterface />
                  <EgressInterface />
            </ServiceScope>
            <FlowDescription>
                  <DSCP />
                  <SrcIP />
                  <DstIP />
                  <SourcePortRange />
                  <DestinationPortRange />
                  <Protocol />
            </FlowDescription>
            <PerformanceGuarantees>
                  <One-Way-Delay />
                  <PacketLoss />
                  <IPDV />
                  <Capacity />
<MTU />
            </PerformanceGuarantees>
            <MonitoringInfrastructure />
            <Reliability>
                  <MeanDownTime />
                  <TimeToRepair />
            </Reliability>
      </SLO>
</SLA>
```

3.3 SLA Framework Architecture

In the following section we describe our SLA Framework architecture. The SLA Framework should be installed in each domain across the end-to-end path. Well-defined APIs will carry out the communication between the SLA Frameworks of the involved domains, including all required intra-domain entities and the administrator who will be responsible for the SLA management within each domain. The APIs for the inter-domain communication can be categorized into three levels:

Administrative Interface: Through a web interface the domain administrator can manage and configure the SLA Framework.
SLA Communicator: It is responsible for the communication between the different entities of each framework. By calling the appropriate web services, each SLA Framework can collect required data, interchange messages, and generate either the per-domain SLAs or the end-to-end SLA if it is the first framework in the path.
SLA Manager: Through this entity SLA Frameworks in different domains communicate with each other.

The SLA Framework can be deployed in network reservation systems in order to provide end-to-end SLAs along with the network reservations. The end-to-end SLA calculation if required would be triggered by the end user during the service request. More specifically, the SLA installation procedure accompanies the service request and can be described as follows:

The service requests are received through the *Administrative Interface* by the first domain on the reservation path and they are forwarded sequentially to the rest of the domains in this path through the *SLA Communicator*.

Each domain evaluates the network reservation request and produces the local SLA. Afterward it responds backward through the *SLA Communicator* to its previous domain.

The first domain on the chain collects the SLAs of all the SLA Frameworks on the chain in order to produce the end-to-end SLA.

SLA Manager exposes the above procedure through well-defined web services.

SLA Framework, also, maintains its own SLA Database. This database keeps local SLAs of the domain, internal data of the SLA Framework, and end-to-end SLAs. The data can be viewed and edited via the administrative functions of Administrative Interface.

Figure 28.2 shows SLA Framework's architecture and its components.

4 SLA Management

In the following section we will describe the SLA service life cycle. It involves three phases: creation, publication, and negotiation.

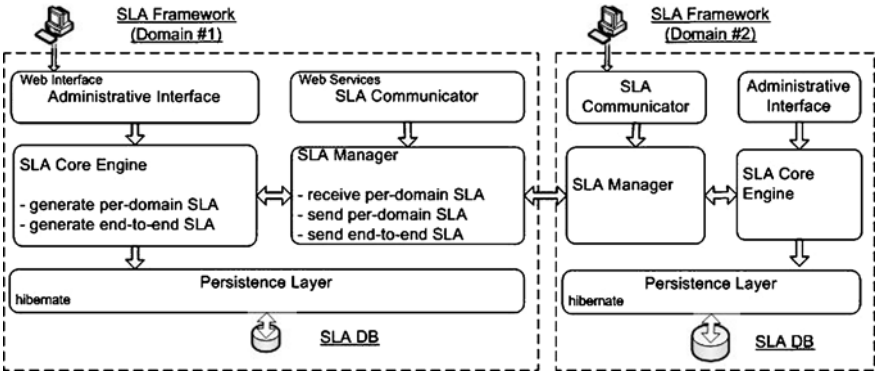


Fig. 28.2 SLA Framework architecture

1. *Creation Phase*: The main steps involved in the service creation process are the following:
   Define a description of the business process associated to service trading in a standard format.
   Fill the corresponding Administrative Service Object and the Service Level Specification.
   Store per-domain SLA in local database (SLADB in Fig. 28.2).
2. *Publication Phase*: After the creation phase, the newly defined per-domain SLA needs to be published to be available for merging e2e SLAs. Publication consists in the transfer to the first SLA Framework of the reservation path through SLA Manager component.
3. *Negotiation Phase*: Once the per-domain SLA has been defined and published, the first SLA Framework in the reservation path negotiates and merges all the per-domain SLAs in order to establish the end-to-end SLA. Afterward, it stores the e2e SLA to the local SLA database and it informs all other SLA Frameworks in the chain through the SLA Manager.

If a domain in the path does not deliver its respective SLA, the first SLA Framework informs the client that an SLA cannot be delivered for the requested IP service, so that the client can make a new request with different level of QoS.

5 SLA Monitoring

In an inter-domain environment that allows dynamic negotiation of a service, mechanisms for monitoring the performance parameters for a specific service instance must be available. Such information is critical for both the end user and the service provider, since the former is the entity that requests the service and the latter is the entity that delivers the service. Evaluating the monitoring results, a user can ask for SLA re-negotiation if the negotiated service level is not the appropriate

one. Similarly, a provider can ask for an SLA re-negotiation in order to optimize utilization of its resources.

In order for the monitoring procedure to be feasible, some monitoring requirements must be available:

Network performance data should be available for every SLA that corresponds to an end-to-end reservation.

The SLA should be monitored on an e2e basis, with the point of measurements as close to the end point as possible.

For every SLA monitored, the following metrics should be collected:

- OWD (One-Way Delay)
- RTT (Round Trip Time)
- IPDV (IP Delay Variation)
- Packet Loss
- Capacity
- Achievable bandwidth
- MTU (Maximum Transmission Unit)

Measurements should be taken frequently, e.g., every 5 min, in order to provide an up-to-date view of the network performance.

It should also be possible to make on demand measurements with no impact on the network performance.

6 Conclusions

In this chapter we propose an SLA Framework for automated SLA management, creation, negotiation, and publication in an inter-domain environment.

The automated management and monitoring procedures have been presented as they have been designed for an inter-domain system. Performance and monitoring issues needed for the deployment of an SLA enforcement framework in the IP network have also been taken into consideration.

The proposed framework has been adopted within the GÉANT2 operations and will be deployed in its Advanced Multi-domain Provisioning System (AMPS) [9]. The integration of our e2e SLA Framework in AMPS will give us the opportunity to evaluate and measure its performance in real QoS-enabled networks such as the European Research and Education Networks and their interconnecting backbone GÉANT2.

Acknowledgments This chapter was partially supported by Projects GN2 (GÉANT2) and GRIDCC of the European Union's 6th Framework Program of Research and Technological Development.

References

1. J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry. *The COPS (Common Open Policy Service) Protocol*, Jan. 2000. RFC2748.
2. C. Courcoubetis and V. Siris. Managing and pricing service level agreements for differentiated services. In *Proceedings of the IEEE/IFIP IWQoS'99*, UCL, London, UK, May 31–Jun. 4, 1999.
3. A. Dan, H. Ludwig, and G. Pacifici. *Web Services Differentiation with Service Level Agreements*. IBM Software Group, 2003. Web Services Web site.
4. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 1998.
5. GRIDCC project. http://www.gridcc.org
6. Y. Hoffner, S. Field, P. Grefen, and H. Ludwig. Contract driven creation and operation of virtual enterprises. *Computer Networks*, 37:111–136, 2001.
7. J. Martin and A. Nilsson. *On Service Level Agreements for IP Networks*, New York, 2002.
8. K. Nichols and B. Carpenter. *Definition of Differentiated Services Behavior Aggregates and Rules for their Specification*, Feb. 2000. draft-ietf-diffserv-ba-def-00.txt
9. A. Patil, B. Belter, A. Polyrakis, T. Rodwell, M. Przybylski, and M. Grammatikou. *Advance Multi-Domain Provisioning System*. http://tnc2007.terena.org/programme/ presentations/show.php?pres_id=20
10. A. Sahai, A. Durante, and V. Machiraju. Towards automated SLA management for Web services. In *Res. Rep. HPL-2001-310 (R.1)*, Hewlett-Packard (HP) Labs Palo Alto, Jul. 26, 2002.
11. SEQUIN project. http://www.dante.net/sequin/
12. D.C. Verma. Service level agreements on IP networks. *Proceedings of the IEEE*, 92(9): 1382–1388, Sept. 2004.

Part V
eVLBI and Cosmic Rays Detection

Chapter 29
High-Bandwidth Data Acquisition and Network Streaming in VLBI

J. Wagner, G. Molera, and M. Uunila

Abstract Astronomers want to have faster response times to stellar events and an increased sensitivity of radio astronomic interferometry methods that they use often. This requires a new data acquisition system to be developed to upgrade the several decades old existing systems in Europe. Due to the nature of radio interferometry and the several radio observatories involved, such a data acquisition system is inherently a distributed system. We will describe here some of the requirements of a new 10 Gbps network-connected data acquisition system. In addition, we present some results from evaluation tests that we have performed with consumer computers and 10 Gbps networking.

Keywords Gigabit data acquisition · 10 Gbps networking · UDP user protocols

1 Introduction

In radio astronomy, Very Long Baseline Interferometry (VLBI) is an observation method that provides a resolution superior to other methods. A widely dispersed array of radio telescopes is used to create a very large virtual telescope by taking the measurement time series of all the telescopes and combining them computationally. This virtual telescope can be as large as the radius of the Earth, with a resolving power nearly the same as that of a physical telescope of the same size. In space-based VLBI, satellites are used to form an even larger virtual telescope.

In astronomy, VLBI is used to observe distant objects called quasars. Quasars are part of Active Galactic Nuclei (AGN) and have central super-massive black holes. The AGN look like very bright point sources and are only about 1 parsec in size, thousands of times smaller than the much darker host galaxy. The radiation of the AGN can vary rapidly and strongly, which also proves that the radiation originates in a small area [10]. With VLBI, AGN can be imaged at a very high resolution.

J. Wagner (✉)
Metsähovi Radio Observatory, TKK, FIN-02540 Kymälä, Finland
e-mail: jwagner@kurp.hut.fi

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_29,

Geodetic VLBI (geo-VLBI) observes quasars, too; however, the data is used to determine the orientation of the Earth's axis and the Earth rotation. Geo-VLBI also provides precise Universal Time estimates.

The VLBI quality improves proportionally to the distance between the telescopes and the data rate of the data collected. The data is currently processed in a central computing center (correlator). Telescopes are separated by thousands of kilometers making data transfer an interesting problem. The prior art is to store station data on disks and send disks via mail to the correlator. However, costs for 1–10 Gbps connectivity have plummeted and even faster 40G/100G links are already being built; networks now provide low-cost capacity even beyond VLBI demands. Gigabit connections now reach abandoned areas such as stations near the Arctic Circle. It is now possible to network telescopes and transfer the data to the correlator using either real-time streams or files. This removes weeks of mail delivery delay and even allows data processing to start in real time.

In the Express Production Real-time e-VLBI Service EU project (EXPReS), the essential goals are to connect all telescopes of the European VLBI Network (EVN) to at least a 1 Gbps connection, create a production quality real-time data transfer method and update the old data acquisition systems at the stations, and the JIVE correlator in Netherlands to cope with the higher bandwidths, in order to meet the target of 4 Gbps from each of over 16 EVN stations. Several institutes are involved in this development process. We at Metsähovi are to provide a new network-connected multi-gigabit data acquisition system. Some test results leading to such a system are described in this chapter.

2 High-Bandwidth Data Acquisition

In radio astronomy, the current data acquisition systems (DAS) used in each radio observatory take the radio signal of the telescope, process it, and store the digitized data onto a recording system. The hardware is currently the limiting factor in improving the scientific data in radio astronomy. Table 29.1 qualitatively demonstrates how VLBI sensitivity improves when the observation data rate is increased [6]. Higher rates reduce image noise and allow smaller brightness temperature differences to be resolved.

Table 29.1 Brightness temperature and images RMS noise at 8.4 GHz depending on the data recording rate

	Brightness temperature	RMS noise
Rate	8.4 GHz	8.4 GHz
128 Mbps	8e4	30
1 Gbps	3e4	10
4 Gbps	1e4	5.2
8 Gbps	9e3	3.7
16 Gbps	7e3	2.6

In many European stations the DAS has been working non-upgraded for the last 10–20 years at 128–512 Mbps rates. Developments in radio, network, and computer technology would offer a huge improvement to the old DAS capabilities. Already in the past 5 years the old DAS could be improved to acquire and transfer data at $\leq$1 Gbps rates. However, these old DAS cannot provide the larger bandwidth demanded by astronomers. A new DAS that natively supports network streaming is required. The requirements are detailed in the next section.

2.1 10G Data Acquisition Design Requirements

The telescope data consists of samples of radio noise from an astronomic source. Old data formats are a legacy of the magnetic tape recording era. They have to be encapsulated for network transmission. Later processing is cumbersome except in hardware. Newer formats such as the Mark 5C Data-Frame Specification are designed with networking in mind, however most are still in draft stage. They essentially contain time-stamped samples and additional channel information. They are similar to formats often seen used in conjunction with the Real-Time Streaming Protocol (RTSP).

Astronomers and geodesists have traditionally used the same analog VLBI data acquisition systems. A new digital DAS should preferably have the capabilities required by both users. Over a decade of Software Defined Radio (SDR) is now paving its way into VLBI, too; all-digital SDR allows to design digital RF downconversion and digital filtering in ways that suit both VLBI user demands. Many SDR designs are reconfigurable in terms of signal channelization, channel bandwidths, and filter steepness. While analog implementations provide superior signal quality, they take a long time to design, are less agile, and perform less consistently. In these regards SDRs are very attractive for a new DAS.

Several FPGA-based SDR systems exist: the Japanese 4 Gbps ADS-3000, Chinese 4 Gbps CDAS, Italian dBBC [9, 13], and US DBE1. Inexpensive competing designs are based on FPGA computing boards [7]. The boards are a spring-off of the Berkeley Wireless Research Center (BWRC) high-performance computing platform BEE2 [1]. The iBOB and Roach boards have several advantages: they are commercial off the shelf, have low cost in respect to performance, and include ready component libraries.

The DAS requirements in EXPReS are not clearly stated. Support for the old hardware correlator system would require the DAS to sample 1–4 IF signals that are band-limited to below 1 GHz. From these IFs the DAS would extract 2^N baseband (BF) channels with a bandwidth of 2^M MHz. The 1–2-bit samples from all channels would be interleaved into a legacy tape drive data format. However, for the new distributed software correlators better data formats are preferable. The selection of channels, bandwidths, and bit resolution are very free. As the iBOB toolflow is based on Matlab and Simulink, the DAS signal processing part can be a basic design on which the automated optimization of Simulink block input parameters is used to get

the best FPGA program for either old hardware or new software correlators. As the output data format we will use software and network-targeted non-legacy formats.

The DAS interface has to be decided, too. Old DAS use a short-distance bus called VLBI Standard Interface (VSI). The dBBC and DBE DAS maintain support for it because hardware correlators interface to VSI. VSI does not electrically support data rates over 2 Gbps and a Metsähovi VSI board is needed to access the bus from PC software. As distributed computing outperforms hardware correlators in scalability and cost, standard 10 Gbps networking instead of VSI or other proprietary interface is the best choice as the DAS interface.

Data storage is another consideration. Real-time correlation can be preferable and it requires practically no storage. Current correlators are hard-pushed to process several stations at 2 Gbps real-time rates each. For scientific reasons, too, real-time correlation may not be an option for all VLBI observations. This means that the data from the DAS may have to be stored before processing. Current consumer computers are capable of over 4 Gbps storage per computer, as we will show later, allowing high-speed data storage nodes.

With 10G connectivity, storage nodes can easily receive the real-time DAS data and store it to disk. This data can later be accessed by the correlator using, for example, an improved version of the Tsunami UDP [11] protocol discussed in Section 3.3.

2.2 10G Data Acquisition Hardware and Test Design

The iBOB board has two copper 10 Gbps (CX4) ports and two ports for I/O extension cards. The board is based on the Virtex II Pro FPGA with access to 4.5 MB of fast on-board SRAM. One of the available I/O cards is the iADC card; the iADC has an Atmel 8-bit analog-to-digital converter chip for two analog IF inputs with external sampling clock. The required 512–1,024 MHz sampling clock is generated from a stable 5 MHz hydrogen maser clock reference using low-cost National LMX3521 or Analog ADF4360 -based frequency synthesizer boards. The highest iADC sampling rate is 2G sample/s. With two iADC boards the raw data rate is 32 Gbps. The developer needs to design a Simulink model and VHDL code to process the high-speed data and send it out over the two 10G links.

Figure 29.1 shows one of the iBOB systems at Metsähovi. The upper part of the figure shows the iADCs with clock and dual analog channel inputs. The iBOB is below. One of the 10G-streaming Simulink designs that we created is shown in Fig. 29.2. This design digitizes two analog IFs and offers configurable sample rate conversion and configurable 10G streaming of the two IFs.

The already fully functional test design of Fig. 29.2 will need future refinement. To extract arbitrary channels from the IFs, the addition of digital downconversion (DDC) blocks is required. For a clean 10G output data format with samples from each channel placed into separate data streams, buffering will be needed. The small size of SRAM memory limits the possible number of channels to around 512,

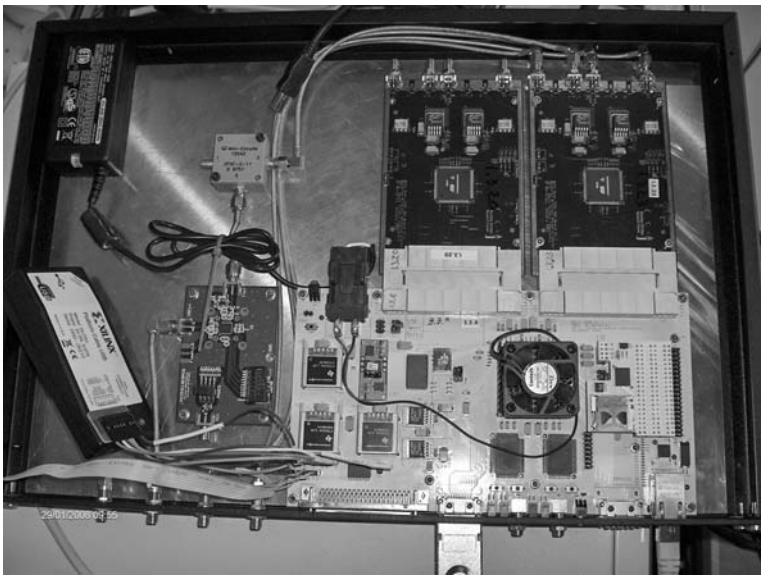


Fig. 29.1 The iBOB hardware with two ADC boards and a frequency synthesizer. January 2008

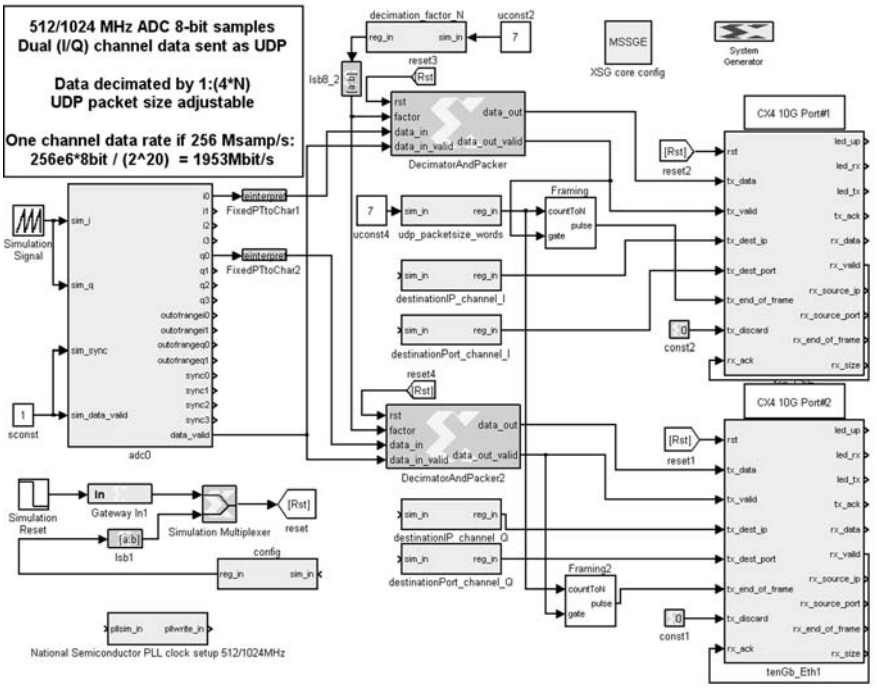


Fig. 29.2 An iBOB test design for IF-to-Ethernet streaming. The design digitizes two analog channels, followed by a configurable sample rate conversion step and 10G UDP streaming

still more than the old DAS's 16–32. The design of the DDC is under work. Of the options, the least flexible is the fast Fourier transform polyphase filterbank, a uniformly distributed multi-channel filter used, for example, by MIT Haystack. Quadrature sampling, N-path or Cascaded Integrator Comb filters present a more agile solution [5]. It is possible to implement such a DDC without using FPGA multipliers. Once the DDC design is fully implemented, it will be reconfigurable through on-board software.

The two PowerPC processors on the FPGA run on-board control software to configure the implemented hardware. A basic serial and Telnet shell (TinySH) is provided by BWRC. Hardware settings in the test design, like the iADC configuration, data rate, and stream clients, can be changed from within TinySH. This allows the iBOB to be configured remotely.

3 High-Bandwidth Network Streaming and Stream Capture

Unlike heavily compressed video data, VLBI data is radio noise and not very sensitive to data loss. The signal-to-noise ratio of the VLBI correlation result degrades only proportional to $\sqrt{1-f}$, where f is the loss fraction. Data loss of even 10% is acceptable. For real-time applications a UDP/IP-based user protocol becomes the natural choice. UDP is extremely simple to implement on an FPGA. In addition, UDP allows control of the transmit rate and can maintain the real-time constraints of the DAS. This is not possible with TCP. The seemingly obvious advantages of UDP versus TCP have been investigated in the VLBI field for years, generally finding that UDP-based transfer can minimize real-time VLBI data loss [8]. As the data integrity requirements for real-time VLBI are very relaxed, a user UDP protocol even simpler than standard real-time RTCP/RTP protocols is hence suitable for a VLBI DAS. The sufficient properties for a usable VLBI UDP protocol are summarized below:

1. UDP packets will be sent with a fixed length, with radio noise payload.
2. A 64-bit sequence number is enough to deduce the timestamp of each sample.
3. Even 10% data loss is acceptable, re-transmissions are unnecessary [8].
4. Network packet reordering is too low to justify client-side reordering.
5. There is no true need to verify the integrity of the UDP payload data or PSN.
6. Optional discrete congestion control: skip samples, reduce channels, or bits.

Congestion control can be implemented in the above ways even in real-time streaming. However, most shared traffic links in GEANT2 and the NRENs run at low utilization ($\leq$15%). Congestion would occur rarely, if at all, especially if the current VLBI trend with lightpaths continues. For this reason see attempts to create "TCP-friendly" real-time UDP protocols with congestion control as unnecessary. Again, this simplifies the implementation of the UDP protocol. With these points in mind, a simple UDP streaming setup was implemented on the iBOB board.

If the stream is captured to files on disk and these files are to be transferred elsewhere later, a reliable transfer method is preferred. The Grid tools have several

transfer methods. All are relatively slow. For high throughput, UDP-based reliable protocols like Tsunami UDP should be used. In a later section we present test results for Tsunami UDP transfers and also UDP real-time streaming.

If the stream is captured to disk, another challenge is disk system performance. The main hardware issues in recording are as follows:

1. *RAID performance:* RAID level 0 has the highest sequential I/O rate. Hardware RAID controllers and subsystems are slow ($\leq$300 MB/s) and expensive compared to the throughput. Several parallel systems would be required for 10G rates. In VLBI, software RAID 0 on a 10G PC is the best choice.
2. *Chipset and Serial-ATA II bus:* Many consumer motherboards support 6–12 SATA disks. Single-disk transfer rates are typically between 50 and 125 MB/s. With 6–12 disks a 2.4–12 Gbps RAID-0 is possible in theory, but the chipset and architecture may not allow the full rate.
3. *Hard disk capacity:* Current 1 TB disks allow a 10G storage server with 6–12 TB of capacity. This corresponds to only 1 day of 4 Gbps single-telescope data. For more capacity, systems have to paralleled.

Real-time processing of higher than 1 Gbps station rates is not yet possible at the European correlator stations. For high rates, temporary storage is still required. The correlator stations are perhaps understandably not so interested to provide this capacity at their end. Real-time storage at the stations followed by later remote access to the data by software or hardware correlator systems currently seems to be the only solution. Thus, the first DAS client will have to record the DAS data stream onto RAID at the station.

To develop such a 10G DAS client, we assembled two 10G test computers at the end of 2007. We purchased additional networking hardware to add a 10G LAN to our 10 Gbps Funet NREN Internet connection. A detailed list of the hardware is provided in Table 29.2. The total cost of the new HP switch and 10G computers was around €10k.

Table 29.2 Testing hardware used at Metsähovi Radio Observatory

Hardware	
Internet	10 Gbps over XR fiber to Funet/CSC
Switches	1 x Extreme Networks Summit X450-24t with a XR and SR Xenpak 1 x HP 6,400cl CX4 with dual SR fibre
Network cards	1 x Chelsio N320E-CX dual-CX4 eval kit (two cards and a cable) 1 x Myrinet CX4 1 x Myrinet XFP with SR fiber
iBOBs	2 x iBOB with ADC, connected to the HP switch
PCs	Adibal: Asus LIN64-SLI WS, two dual-core AMD 2212 CPUs, 4 GB RAM 12 x SpinPoint F1 SATA in RAID 0 Juliano: Asus P5WD2-E Premium, Core 2 Duo 3.2 Ghz, 2 GB RAM 4 + 4× SATA in RAID 0
Settings	Locally 9,000 MTU, Funet 4,470 MTU

3.1 UDP-Based Streaming

The UDP protocol can be kept very simple as described in the previous section. We considered it sufficient to perform *iperf* and disk tests to benchmark our hardware first. After this we proceeded to implement UDP tools based on the findings.

We ran a number of local *iperf* tests between the two computers on a 10G LAN. It came as a surprise to us that, out-of-box, the theoretically lightweight UDP was more CPU intensive and by half slower than hardware-offloaded TCP. The main problem turned out to be UDP packet checksums. These were verified in software by the network card driver and the high CPU load reduced performance. The issue could be worked around in two ways. With two 4.95 Gbps UDP senders (*iperf* clients) running concurrently, 9.9 Gbps aggregate through the 10G card was possible. Alternatively, a single UDP sender with UDP checksums disabled using the *SO_NO_CHECK* socket option allowed a 9.05 Gbps rate.

VLBI UDP payload data integrity is not an important issue, so UDP checksums can be disabled safely. Today's network links have a low-bit error rate and the probability that the typical CRC-32 of the data link layer misses an erroneous bit is very small, 2×10^{-9}. Even gone undetected these errors cause no noticeable degradation in the VLBI correlation result.

Another important point for performance is to use as large a path maximum transfer unit (PMTU) as possible. The effect of the MTU in local *iperf* tests is shown in Table 29.3; only large MTUs allowed good throughput. UDP packet size is a further consideration. On shared networks the packet loss can be minimized if packets not larger than the MTU are used, avoiding IP fragmentation. This is a compromise as the CPU load on the receiving client may increase.

Table 29.3 The effect of the MTU setting of both PCs on the CPU usage with non-checksum forced

MTU (bytes)	UDP size (kb)	Target (m)	Result (Gbps)	One core 'top' on sender
1,500	32	9,000	4.88	54.0%sys iperf 100%
4,470	32	9,000	8.55	58.8%sys iperf 100%
9,000	32	9,000	9.04	49.8%sys iperf 100%
9,000	32	9,900	9.04	56.0%sys iperf 100%

The conclusion from the tests is that with the current drivers and hardware we have to use jumbo frames and have to disable UDP checksumming. We developed simple UDP server and client programs that take these findings into account and that give performance similar to *iperf*. These programs will be discussed in the next section.

3.2 Stream Recording

To test UDP stream recording to disk, we first had to benchmark the performance of the 12-disk software RAID-0 on the Abidal PC (Ubuntu, kernel 2.6.23, 64-bit).

Initially we used over 1-year-old 320–500 GB disks and tested several file systems on the RAID-0. Out of XFS, ext2, ReiserFS, and FAT32 the best throughput with any I/O size ≥8 kB was possible using *XFS*, which gave 3.8 Gbps performance for writing and 4.5 Gbps for reading. We tested the raw */dev/md0* block device with no file system, too, using the *wr-nexgen* program and 0.5TB files. At the beginning of RAID-0 capacity 4.9 Gbps write rates were possible [3].

An upgrade to SpinPoint F1 750 GB disks gave a noticeable improvement to sustained write rate. Individual F1 disks gave a rate of 105 down to 45 MB/s. For the 12-disk RAID we expected a theoretical maximum rate of 1,200 MB/s or 9 Gbps. The result of one set of actual throughput tests is shown in Fig. 29.3. It shows the write performance of a 8–12 disk RAID-0 with or without an XFS file system. The jump in write rate at the end of RAID capacity is peculiar to XFS; we could not see it with other file systems. The thoughput test confirmed that 4 Gbps recording is possible using 12 F1 disks or with 8 F1 disks up to 50% capacity.

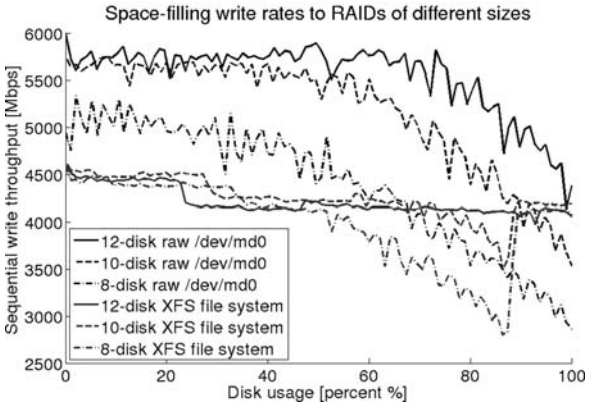


Fig. 29.3 Sequential write throughput of a software RAID-0 built from 8–12 750 GB SpinPoint F1 disks and with or without an XFS file system

In order to capture 10G DAS streams to disk, taking into account the performance considerations of Section 3.1, we wrote new (*raid2udp*, *udp2raid*, *raw2raid*) client and sender programs. Source code can be found on our web site [4]. For the combined performance of disk and network I/O, it is important to use either zero-copy and explicit flushing or create I/O worker threads to utilize all CPU cores. With these programs and considerations we achieved a sustained *udp2raid* recording rate of at least 4.2 Gbps from the 10G iBOB test design to RAID disk.

The remaining task was to see whether the Tsunami UDP protocol could be used for ≥4 Gbps data transfer and disk recording on a 10G network.

3.3 Tsunami UDP

Tsunami is a fast UDP-based reliable file transfer protocol [11]. In 2004 we extended it to stream VLBI data in real time from Metsähovi PC-EVN DAS [14]. Tsunami is also widely used by European geodesists to transfer terabytes of data files at near 1 Gbps speeds, and it has often been used in real-time transfer and processing tests, too.

We wanted to see if locally recorded DAS data could be copied out at above real-time rates. To test, we let Tsunami transfer files on the 10G LAN from a server PC to a client PC that either discarded or wrote the data to RAID. It was immediately possible to get 3.5 Gbps peak rates, even combined with RAID disk writing. Further tests revealed some Tsunami v1.1 protocol settings to be 32 bit. These posed certain limitations such as capping the rate at 4 Gbps. To improve the protocol, we started a new petabit version 1.2. It is available on the SourceForge site [11].

With the v1.2 improvements we achieved 7 Gbps for memory-to-memory transfers, while the same test with the competing UDTv4 protocol [12] achieved only 5 Gbps. Transfers to RAID using the old set of SATA disks were not faster than 3.5 Gbps peak. With the new SpinPoint F1 disks, however, throughput increased similar to Fig. 29.3. An example of a 4 Gbps Tsunami file transfer of a 20 GB file is shown in Fig. 29.4.

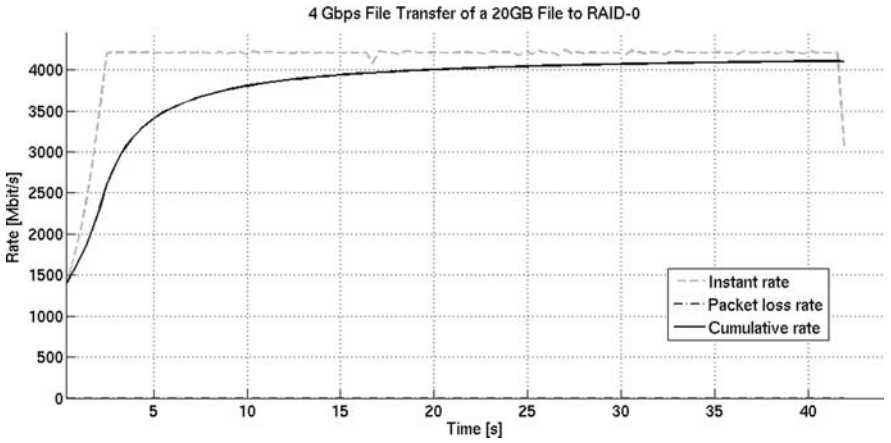


Fig. 29.4 Tsunami transfer of a 20 GB file with a 4.3 Gbps target rate

Currently no other VLBI station with a working 10G connection and test computers was available, so we were restricted to local 10G LAN tests. Based on GEANT2 link utilization and previous Tsunami experience, we can estimate that rates will be very close to the above 10G LAN rates.

4 Conclusions

The results from the stream recording tests proved that it is already today possible to record network data with a single computer at a sustained 4 Gbps rate using inexpensive consumer hardware. It is a low-cost, low-power consumption, and high-speed storage solution for the temporary VLBI station data storage, offering unprecedented data rates. Hardware is developing quickly and single computer recording rates around 10 Gbps should be possible before 2010.

The VHDL and software for the iBOB DAS, with network streaming and configurable RF channelization, are currently under active development at Metsähovi. A successful first test design for streaming sampled analog data was presented.

Not only 4 Gbps streaming and stream capture are now possible but later access to recorded files is possible at high data rates, as well. The improved Tsunami UDP file transfer protocol allowed disk-to-disk transfer rates over 4 Gbps.

Tsunami transfers and iBOB DAS streaming outside of our 10G LAN, over the Internet, will be interesting to trial in the future.

We also plan to improve the UDP protocols introduced here. Considered additions are UDP multicast mode and future integration via a Globus XIO, OpenMPI, MPI-CH, or VSI-E transfer protocol. A further application may be in our real-time software correlation and RF spectrometer tools for the Playstation 3 that are currently under development [2].

Acknowledgments This work has received financial support from the European Commission (DG-INFSO), within the Sixth Framework Programme (Integrated Infrastructure Initiative contract number 026642, EXPReS).

References

1. C. Chang, J. Wawrzynek, and R.W. Brodersen. BEE2: A high-end reconfigurable computing system. *IEEE Design and Test of Computers*, 22(2):114–125, Mar. 2005.
2. IBM Cell Miniature Software Correlator, Jan. 2008. http://cellspe-tasklib.sourceforge.net/
3. Metsähovi 10 Gbps Testing Lab Notes, Jan. 2008. http://www.metsahovi.fi/en/vlbi/10gpbs/index.html
4. Metsähovi PC-EVN Source Code, Jan. 2008. http://www.metsahovi.fi/en/vlbi/vsib-tools/packages/src/
5. L.E. Pellon. A double nyquist digital product detector for quadrature sampling. *IEEE Transactions on Signal Processing*, 40(7):1670–1681, Jul. 1992.
6. C. Phillips. eVLBI Scientific Benefits, Jul. 2004.
7. Roach Wikipedia Website, Jan. 2008. http://casper.berkeley.edu/wiki/index.php?title=ROACH
8. R. Spencer, R. Hughes-Jones, A. Mathews, and S. O'Toole. Packet loss in high data rate internet data transfer for eVLBI. In R. Bachiller, F. Colomer, J.F. Desmurs, and P. deVicente, editors, *Proceedings of the 7th European VLBI Network Symposium*, Toledo, Spain, Oct. 12–15, 2004.
9. H. Takeuchi, M. Kimura, J. Nakajima, T. Kondo, Y. Koyama, R. Ichikawa, M. Sekido, and E. Kawai. Development of a 4 Gbps multifunctional very long baseline interferometry data acquisition system. *Publications of the Astronomical Society of the Pacific*, 118:1739–1748, Dec. 2006.

10. The Metsähovi Web Pages, Mar. 2008. http://www.metsahovi.fi/en/
11. The Tsunami Web Page, Jan. 2008. http://tsunami-udp.sf.net
12. The UDT Web Page, Jan. 2008. http://udt.sf.net/
13. G. Tuccari, S. Buttaccio, G. Nicotra, W. Alef, R. Keller, M. Nalbach, and M. Wunderlich. DBBC – A flexible environment for VLBI and space research: Digital receiver and back-end systems. In J. Boehm, A. Pany, and H. Schuh, editors, *Proceedings of the 18th European VLBI for Geodesy and Astrometry Working Meeting*, vol. 79. Geowissenschaftliche Mitteilungen Heft, 45, 2007.
14. VLBI Research at Metsähovi, Jan. 2008. http://www.metsahovi.fi/en/vlbi/

Chapter 30
Real-Time Software Correlation

N.G.H. Kruithof and D.C.P.M. Marchal

Abstract In this chapter we present the progress of the SCARI*e* project, where we investigate the capabilities of a next generation grid-based software correlator for VLBI. We will mostly focus on the current design of our software correlator and on the challenges of running real-time scientific experiments on top of grids infrastructure. This chapter also contains experimental results on software correlation as well our current experiments on the DAS-3 grid and StarPlane, its user-controllable dynamic photonic network.

Keywords Radio astronomy · VLBI · Grid computing · Real time · Distributed applications · High-speed networks

1 Introduction

Very Long Baseline Interferometry (VLBI) [10] is a type of interferometry used in radio astronomy, in which data received at several telescopes is combined to produce an image with very high angular resolution. This is important as recent astronomical research studies the deep-sky and thus requires high angular resolution to capture all the details of the observed sources. The angular resolution depends on the size of the radio-telescope dish, but due to mechanical constraints, it is difficult to build moveable telescope dishes with a size much larger than 100 m. With VLBI, it is possible to simulate a dish of a size equivalent to the maximal distance between the farthest telescopes and so making a virtual telescope with a dish of the size of the Earth. This is done by having several distant telescopes observe the same source; the signals are recorded and sent to a central facility for processing. The central step in processing the data is computing the correlation function between each pair of incoming signals. Within the SCARI*e* project we are developing and analyzing the capabilities of software correlation using grids technologies. Given the amount of

N.G.H. Kruithof (✉)
Joint Institute for VLBI in Europe (JIVE), Postbus 2, 7990 AA Dwingeloo, The Netherlands
e-mail: nico@nghk.nl

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_30,

processing power needed to perform software correlation, we are distributing the computation using a master-worker approach.

SCARI*e* is a typical example of a recent trend of the e-Science community in which the worldwide computation centers and the scientific instruments are connected through high-speed networks. We think that to generalize the utilization of such world-size inter-connected facility, the grids and their middleware have to offer services that match the three following important aspects:

- *Performance*: solving bigger problems implies the use of larger and/or more efficient hardware.
- *Isolated environment*: some users require the guarantee of isolated execution environment in which the other applications cannot interfere. This is important for real-time applications or for benchmarks.
- *Scheduling*: it may be needed when the application requires to be synchronized with "external events": like a radio telescope observing at a specific date.

In SCARI*e* we are facing these three challenges, as we want to build a high-performance software correlator able to process in real-time a VLBI experiment. These three aspects are hot topics in the grid community, especially the networking side; probably because worldwide networking (Internet) has a long history in being a best-effort shared resource. As best effort does not provide traffic isolation we are conducting our experiments using the experimental DAS-3 grids and a user-controllable dynamic photonic network called StarPlane that could permit us to build on demand an application specific, isolated virtual network on top of the complete grid.

The rest of the chapter is organized as follows. Section 2 contains a general introduction to VLBI and its recent development called *e*-VLBI. Section 3 describes the architecture of the software correlator. Section 4 describes experiments and benchmarks on executing SCARI*e* on StarPlane DAS-3. Section 5 concludes the chapter with future work.

2 VLBI

To achieve larger and larger resolution in astronomical imaging, it is necessary to build larger telescopes or to revert to interferometry. Interferometry combines the measurements of several telescopes to simulate a dish of a size equivalent to the maximal distance between the farthest telescopes on the plane orthogonal to the viewing direction. Numerous arrays (groups of telescopes) use this technique, e.g., the VLA (Very Large Array), Lofar (Low-frequency array), the EVN (European VLBI Network), or the VLBA (Very Large Baseline Array). Interferometry with telescopes that are geographically very far apart is referred to as Very Large Baseline Interferometry (VLBI). With VLBI it is possible to build a virtual radio telescope with a dish of the size of the Earth. As the angular resolution of a VLBI experiment depends on the longest projected distance between two radio telescopes,

VLBI achieves unsurpassed angular resolution with the drawback that the virtual telescope has relatively low sensitivity compared to an equally large real physical but unfeasible telescope [10]. Another important property is sensitivity, as it allows to detect fainter astronomical objects. Increasing the sensitivity is possible by adding more radio telescopes or by increasing the sampling rate or resolution. Increasing the sampling rate or resolution increases the data rate per telescope.

In order to get the final astronomic observation the signals gathered from the radio telescopes have to be correlated at a central place, the Joint Institute for VLBI in Europe (JIVE). JIVE is operating a dedicated hardware correlator [8].

The maximal capacity of this hardware correlator is 16 telescopes at a data rate of 1 Gb/s each. The requirements on both the data streams and the computing power are shown in Table 30.1.

Table 30.1 Network bandwidths and computing power needed for an *e*-VLBI experiment based on a XF architecture

Description	# telescopes	# sub-bands	Data rate (Mb/s)	Spect/prod	Tflops
Fabric-demo	4	2	16	32	0.16
1 Gb/s, full array	16	16	1,024	16	83.39
future VLBI	32	32	4,096	256	~21,457

e-*VLBI*: Traditionally, in VLBI, the data is recorded at the telescopes on disk packs during an experiment. After the experiment the disks are shipped to a central institute. There can be several weeks between the experiment and the time when the correlated data becomes available.

Currently, JIVE is in the transition phase from traditional VLBI to *e*-VLBI [9]. In an electronic VLBI (*e*-VLBI) experiment, data from the telescopes is transferred directly over the Internet to JIVE, where it is streamed into the correlator in real time. The data transport from the telescopes to JIVE goes over several networks like local connections, paths provided by NRENs, and the GÉANT backbone in Europe.

Transporting the data over the network has several advantages over a traditional experiment. Obviously, the results of the experiments are almost immediately available. This opens up the possibility to change the course of an experiment based on earlier findings. Also, *e*-VLBI allows for real-time analysis of the data and helps to identify and resolve minor technical problems in the data collection during the experiment.

Several experiments in the past have shown that real-time *e*-VLBI is possible. The EC funds the EXPReS project [7] which aims at building a production-level *e*-VLBI instrument of up to 16 intercontinental telescopes connected in real time to JIVE and available to the general astronomy community.

Correlation: Correlation is the process by which data from multiple telescopes is collected and combined to measure the spatial Fourier components of the image of the sky. It consists of two steps: first applying a delay correction to align the signals

and second computing the correlation function for each pair of telescopes called a baseline.

To align the signals from the different telescopes, we project all the telescopes on the plane through the center of the Earth and orthogonal to direction of the source, see Fig. 30.1. We will correlate the signals received by the virtual projected telescopes. To compute these signals, the signals from the real telescopes are delayed with the distance between the telescope and its projection multiplied by the speed of light (the signal travels with the speed of light). Note that the delay changes during the observation because the Earth rotates. We conveniently split the delay in an integer number of samples and a remaining fraction. While the integer delay is easily done by an offset in the sample buffer, the fractional bit shift is usually implemented as a phase rotation in the frequency domain. In a final step, called the phase rotation, we change the sample rate to match the rate of the delay function.

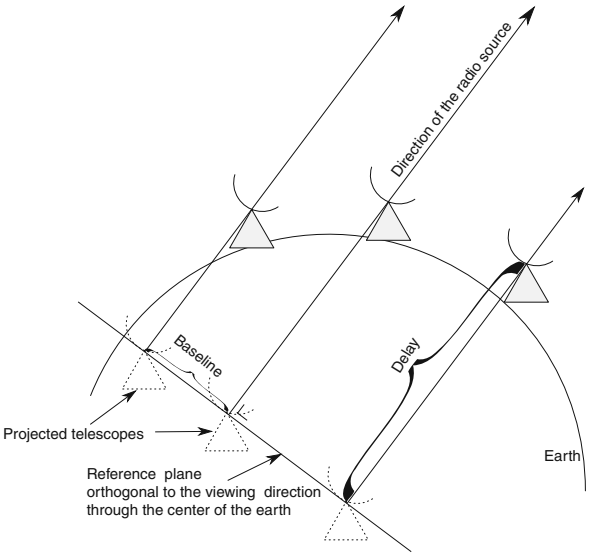


Fig. 30.1 Aligning the signals before correlation

After the delay has been applied, the signals are ready for correlation. Correlation [5] is mathematically defined as a function on two signals in which the first signal is delayed with discrete steps and the integral is computed of the delayed signal multiplied with the second signal. The correlation is done for each baseline (pair of telescopes). The correlation is called an auto-correlation if the signal of a station is correlated with itself and a cross-correlation if the signals are from different stations. Note that the complexity of the correlation is quadratic in the number of telescopes.

To increase the signal to noise ratio, the correlated signal is averaged over a certain period of time. Typical averaging times lie in the range of $0.25-4$ s. Finally, the averaged signals are Fourier transformed. Correlation in this order is referred to as XF, because the correlation is done before the Fourier transform. This is how

the correlation is implemented in most hardware correlators, as it allows for large parallelization.

One of the properties of correlation is that the Fourier transform of two correlated signals is equal to multiplying the Fourier transformed signals [2]. This relation is used in most software correlators, where correlation is more expensive than multiplication. After being delayed, the signal from each telescope is Fourier transformed and the signals for each baseline are then multiplied elementwise. This implementation is also refered to as FX: the Fourier transform comes before the correlation.

3 Software Correlator

If the data in an *e*-VLBI experiment can be streamed over the Internet to JIVE, it can also be sent to another correlator. Within SCARI*e*, we are investigating the possibilities of a next generation software correlator using a computing Grid. A similar attempt is presented in [6]. The advantages of a software correlator over a new dedicated hardware correlator lies in its flexibility and accuracy. The main advantage of a dedicated hardware correlator is the greater performance. The flexibility of its architecture allows the software correlator to change with the individual needs of researchers. In fact, the first version of the software correlator was developed to track the Huygens spacecraft during its descent through the atmosphere of Saturn's moon Titan. Due to the nature of this experiment, special requirements are put on the correlator, which the current hardware correlator is not able to provide. Moreover, we expect that the costs of developing a software correlator are much lower than the costs for a hardware correlator.

Currently, the software correlator is used in the production environment for doing ftp-fringe tests for the EVN network. Since the EVN is an ad hoc array, the EVN telescopes are reconnected before every VLBI session. In order to test the EVN network, a ftp-fringe test is done in which the telescopes observe a well-known source and transmit the data to JIVE where it is processed immediately. The ftp-fringe tests provide quick feedback to the stations on their performance.

Computationally the correlation is relatively inexpensive, in the sense that it requires only few operations per received byte. However, due to the high data rates, the absolute number of flops required by the application is still extremely high for computing grids. Moreover, the problem is quadratic in the number of telescopes participating in the experiment since it is linear in the number of channel pairs that have to be correlated. The huge need for networking and computing power together with its flexibility makes a cluster a good platform for this application. Moreover, using grid technology the software correlator can easily be executed on a large number of different clusters.

3.1 Design

In the software correlator, we split the computation in time slices. These time slices are processed in parallel (see Fig. 30.2). The assignment of time slices to computing

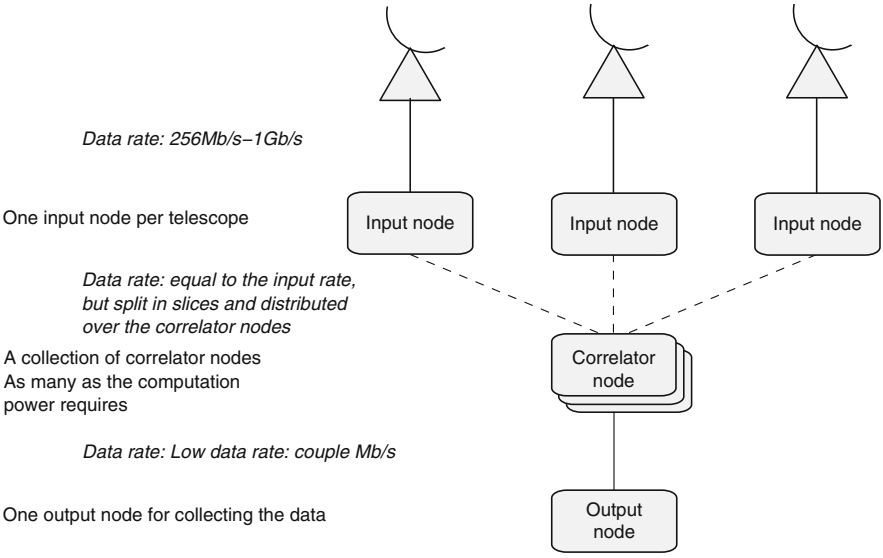


Fig. 30.2 Outline of the network connections between different components in the software correlator

nodes is done by a unique manager node. For every telescope there is a unique input node that receives the raw stream and converts it into time slices. These time slices are sent to correlator nodes which do the actual correlation. The size of the output of the correlation is much smaller than the input size and is collected and stored by a single output node.

The manager node is the central node that controls the workflow of the entire software correlator. It assigns time slices to available correlator nodes and handles errors.

The input node receives the data from a data stream, which can either be a file, a TCP connection, or directly from the dedicated hardware used to record and play back the data. It performs the integer delay correction and then sends the data to the proper correlator node.

The correlator node gets data for the same time slice from every input node. First it compensates for the fractional delay and performs the phase rotation. These manipulations are not done on the input node because they require floating point samples, hence the data stream expands from 2 bits per sample to 32 bits or even 64 bits per sample, which would require much more network bandwidth. After the fractional delay correction and the phase rotation, the signal is ready to be correlated. The auto and cross-correlation are then computed by Fourier transforming the input signal and elementwise multiplying all baselines. These values are accumulated over a certain period of time and the accumulated values are sent to the output node.

The output node receives the data from the correlator nodes, sorts the data, and stores it at a specified location.

4 Execution and Deployment

In the context of *e*-VLBI the software correlator can be executed in two different operational modes that are *batch execution* and *real time*. Grids have a long history in running batch jobs and grid-middleware is now doing a good job on this task. The execution of real-time applications is much more complex; grid infrastructure and middleware have to provide guarantees on the Quality of the Service to ensure successful execution of the job. Quality of Service management in grids is still evolving rapidly. To experiment with these aspects we are running SCARI*e* on a research grid called DAS-3 and its manageable network called StarPlane.

4.1 Real Time and Quality of Service

The term *real time* has many definitions in the computer science community; in this chapter we will consider that a *real-time* computation is a computation in which *the amount of buffering for an infinitely long experiment will only require a finite amount of buffers*. This is a formal way to define a process in which the incoming data are "consumed" by the computation as fast as they are generated. This definition also implies that once the application is started the allocated "space" on the resources will be maintained during the complete execution.

The main resources SCARI*e* is using are the network bandwidth, the computation resource, and the disk space. Sharing of the computational resource is now a well-understood process and most of the time it is part of the execution service that allocates the requested resources and if all resources are acquired, it deploys and executes the application.

The simplest way to offer guaranteed service over a shared resource can be done by restricting the access to only one user at a time. This approach is used in the DAS-3 grid (based on SGE) in which an allocated node is simply unusable by other users. The same principle could be applied to the complete grid including its networks and other resources. A more flexible approach consists in sharing the resource under the arbitration of a third party that will ensure that each application is using only the allocated part of the resource. This is the case with Layer-3 QoS for networks or the Completely Fair Scheduler (per application CPU time allocation) combined with the Process Containers recently introduced into the Linux Kernel [1]. In a very general point of view all these technologies virtualize the resource and thus they permit to build on top of a real grid a complete isolated environment based on user requirements.

*4.2 Running SCARI*e* on DAS-3 and Starplane*

Networking performance and QoS management are the most challenging aspects of SCARI*e*. The regular Internet Layer3 IP routing based on the best-effort policy has a great flexibility but is often slow and unpredictable; on the other hand, we have

dedicated *lightpaths* as available in *lambda Grids* [4], which, with their predictable delays and throughputs, offer good performances and a good basis to offer Quality of Service. Giving end users access to dedicated connection has been implemented in many of the current research and education networks. The Dutch National Resarch and Education network SURFnet is one of them. This is used to deliver the data from the radio telescope to the computation center.

In SCARI*e* having lightpaths between radio telescopes and the computation center is not sufficient to ensure real-time operation as data have to be transmitted between the Input Node and the Correlator Nodes. SCARI*e* also uses the network to distribute the correlation. Therefore, a per application manageable network is then required. The DAS-3 supercomputer [3] is composed of five clusters located in the Netherlands and connected by a photonic network called StarPlane. The StarPlane project manages eight wavelengths with the goal to build a network service that permits *an application-controlled photonic network and node-to-node traffic isolation*. The novelty of the StarPlane project lies in its attempt to build a virtual network service at the lowest possible networking layer: the photonic layer for the optical part and ethernet-layer2 for the connection to the nodes. Another property of StarPlane is that the photonic lightpath can dynamically be reorganized to match the requirements of the user-application. By using StarPlane, a complete virtual network over distributed cluster sites can be built on demand. The lighpaths can also be reorganized at runtime if the network load changes. For SCARI*e* this allows us to distribute the workload over several cluster locations while taking profit of the high bandwidth with a relative good QoS control over the complete network domain.

4.3 Benchmarks on DAS-3

SCARI*e* and StarPlane have parallel roadmaps. Hence, the complete approach cannot be tested yet. We have conducted correlator performance tests using DAS-3. These tests include evaluation of the complete correlation process, the data extraction that takes place in the Input Nodes, and actual correlation that takes place in the Correlator Nodes. The current software correlator is currently able to perform a 4×256 Mbps experiment at 30% of the real-time speed using a total of 15 (quad core 2.0 GHZ cpu) nodes. The experiment was performed at the speed of 4×256 Mbps as this corresponds to a real astronomical observation. Analysis of the software correlator shows that the bottleneck is in the extraction of the data from the input stream (which we can do at 110 Mbps). Hence, we are not able to do real-time correlation, but an optimized prototype shows that we will be able to resolve this bottleneck.

In parallel with the benchmarks of the correlator, we are also testing the capabilities of StarPlane to do high-performance traffic isolation that is needed for real-time correlation. At the time of writing StarPlane has implemented the service that allows a program to build and allocate a lightpath between two clusters on demand.

We tested this feature by running two client-server applications transmitting data between clusters.

The network traffic of the two applications is depicted in Fig. 30.3. At the start of the application, a request for lightpath allocation is issued (arrow 1). After a while the throughput drops because the second application starts sending data as well (arrow 2). As long as the lightpath is not ready (photonic switching takes 10 min) the application is sending the traffic to the default 1 Gb/s ethernet route; when the lighpath is ready (arrow 3) the traffic is rerouted to the lightpath.

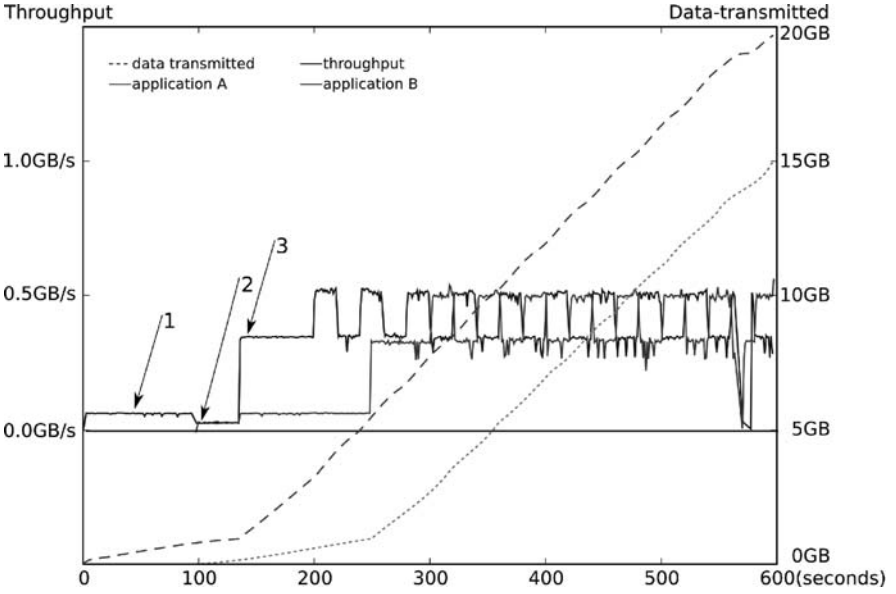


Fig. 30.3 Two applications are started at time (*arrow* 1 and *arrow* 2). The two applications have to share the 1 Gbps network bandwidth. When a lightpath is allocated (*arrow* 3) the increase in performance is clearly visible

The results of this experiment are encouraging as they show that good network performance can be obtained between several cluster locations. Lighpath dynamic switching also permits to adapt the photonic part of the network to the software correlation workload and distribution. Neverthless the results of this experiment also rise questions, the secured lightpath is supposed to deliver reliably "550 MB/s" of throughput between a pair of clusters. In Fig. 30.3 we can see a periodic artifact, the traffic falling down to 300 MB/s for few seconds, for which we have no explanation. A second issue to investigate is that from time to time the lightpath connectivity disappears entirely (e.g., at second 580). We are currently working with the Starplane team to understand and solve these problems.

5 Conclusions

In the last year we laid the foundation for a flexible software correlator based on distributed computing technologies. SCARI*e* is now usable for batch correlation and is used for ftp-fringe tests for the EVN.

In order to reach our real-time correlation goal, we are collaborating intensively with the StarPlane project to test new network architectures for grids with guaranteed Quality of Services.

Future work: Within SCARI*e* and a related project FABRIC, which is a joint research activity in the EXPR*e*S project, we are currently improving and testing the software correlator. We still see possibilities to improve the efficiency of the software correlator, which we would like to investigate further.

On the batch processing aspect of SCARI*e* we want to investigate how resources can be added dynamically at runtime to accelerate the computation. This includes node joining and leaving as well as setting up additional lightpaths to the new nodes.

On the real-time processing aspect, more work has to be done on the traffic isolation features of the incoming networks as well as the isolation provided by Starplane.

Acknowledgments SCARI*e* is a joint research project between JIVE, the University of Amsterdam, and SARA funded by the Netherlands Organization for Scientific Research (NWO).

References

1. Completely Fair Scheduler and Process Containers. `http://en.wikipedia.org/wiki/Completely_Fair_Scheduler`
2. Definition of the Correlation Function. `http://mathworld.wolfram.com/Cross-CorrelationTheorem.html`
3. Distributed ASCI Supercomputer. `http://www.cs.vu.nl/das3/`
4. C. de Laat. Lambda-Grid developments: RDF/NDL, AAA and StarPlane. In *ESLEA Presentations*, 2007.
5. Definition of the Correlation Function. `http://en.wikipedia.org/wiki/Cross-correlation`
6. A.T. Deller, S.J. Tingay, M. Bailes, and C. West. *DiFX: A Software Correlator for Very Long Baseline Interferometry Using Multi-Processor Computing Environments*, 2007.
7. EXPReS Project Website. `http://expres-eu.org`
8. R.T. Schilizzi, W. Aldrich, B. Anderson, A. Bos, R.M. Campbell, J. Canaris, R. Cappallo, J.L. Casse, A. Cattani, J. Goodman, H.J. van Langevelde, A. Maccafferri, R. Millenaar, R.G. Noble, F. Olnon, S.M. Parsley, C. Phillips, S.V. Pogrebenko, D. Smythe, A. Szomoru, H. Verkouter, and A.R. Whitney. The EVN-MarkIV VLBI data processor. *Experimental Astronomy*, 12:49–67, 2001. `http:// adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=2001ExA....12...49S& db_key=AST`

9. A. Szomoru, A. Biggs, M.A. Garrett, H.J. van Langevelde, F. Olnon, Z. Paragi, S. Parsley, S. Pogrebenko, and C. Reynolds. *From Truck to Optical Fibre: The Coming of-Age of eVLBI*, 2004. http://www.citebase.org/abstract?id=oai:arXiv.org:astro-ph/0412686
10. J.A. Zensus, P.J. Diamond, and P.J. Napier, editors. *Very Long Baseline Interferometry and the VLBA*, 1995. http://adsabs.harvard.edu/cgi-bin/nph-bib_ query?bibcode=1995ASPC...82.....Z& db_key=AST

Chapter 31
AugerAccess – Virtual Infrastructure for Simulating Complex Networks

M. Sutter, H.-J. Mathes, K. Daumiller, T. Jejkal, R. Stotzka, A. Kopmann, and H. Gemmeke

Abstract The aims of the AugerAccess project are to improve the communication link between the south site of the Auger Observatory (currently deployed in Argentina) with the international networks of the participating institutions and to develop software for remote control and remote monitoring. The upgrade of the speed of the communication link is also necessary for the functionality of new software components in respect to remote features, because the required transmission rate for measured data exceeds the available capacities. The system for data acquisition in the observatory consists of a large number of computers, several networks, and software and is already running during deployment of the observatory. Because of these reasons the existing system cannot be used for development of the new software components. But a testbed is required for the integration of the new software with the several subsystems in the observatory. The costs for duplicating the needed hardware are too high for development reasons only. The presented solution is to virtualize all needed subsystems of the observatory on one dedicated server and to use this one as a testbed. This chapter shows an easy manageable and flexible solution for virtualizing a complex network with several subnets and a huge amount of computers.

Keywords Virtualization · Instrumentation · Grid · Middleware

1 Introduction

In large scientific projects with detectors for data acquisition, the access to the detectors is very important for management and control. In such projects the detectors can be distributed over a few thousands of square kilometers, e.g., the Auger Observatory [11]. In such distributed systems the access to the sensors must be managed over

M. Sutter (✉)
Karlsruhe Institute of Technology, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany
e-mail: michael.sutter@kit.edu

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_31,

uniform interfaces from a central campus. Inside the campus the security aspects for accessing the detectors are very high if they are distributed over such a huge area. Any unauthorized access must be blocked to ensure that the measured data is authentic.

The security requirements are still higher if the access to the detectors is allowed from outside the central campus from worldwide. This remote functionality is necessary for developers and scientists to get status information about running components of the system and also intended to give the best possible support for operators at experiment site in case of severe problems with the detectors. The access to the data acquisition (DAQ) and Slow Control systems for the "outside world" has to be handled very carefully and unauthorized access has to be strictly prohibited to ensure that the measured data is authentic. The Slow Control System is responsible to control runtime and to check if conditions are accurate for data acquisition.

Nevertheless, software with remote functionalities has more requirements than only the security aspects. Especially for the Auger Observatory these are as follows:

- During deployment of the observatory the deployed part of the system is already used for measuring data with the detectors.
- The existing software for some components of the Auger Observatory was not developed to be remotely accessible from worldwide.
- Access from abroad to a few important components of the existing systems can sometimes be very slow. One of the main objectives of the AugerAccess project is to provide a new, fast, and reliable link to the Auger Observatory.

These are the reasons why the existing system of computers and networks can not be used for development of new software components, as it is not possible to use a running system for development and to risk losing data during that time. So, another solution must be found in order to develop the new software. Therefore two different approaches are possible:

- All necessary hardware components (like computers and switches) can be bought and deployed the same way as they are deployed in the existing system in Argentina.
- The virtualization of all computers and equivalent networks as virtual machines (VM) on only one dedicated server. A VM is the abstraction of the resources of a computer and hides the physical characteristics of these resources from the way the VM is interacting with the resources [19]. Working with a VM is like working with a normal computer, but in reality a special hypervisor interacts with the physical hardware. In this way the resources of a computer can be reused more efficiently by several VMs.

2 Fundamentals

Extending the software of the fluorescence detectors (FD) (Section 2.1) of the Auger Observatory with remote functionality requires major changes in the existing

software. Therefore a testbed during development for testing the integration of the new software with all sub systems of the Auger Observatory is necessary.

In the following sections the aims of the Auger Observatory and the AugerAccess project, as well as the requirements for a testbed, are discussed.

2.1 Auger Observatory

The Auger Observatory is intended to study the universe highest energy particles with energies of over 10^{20} electron volts (eV). Those particles shower down on earth in form of cosmic rays. Cosmic rays with low to moderate energies are well understood, but not much is known about those with extremely high energies. The Auger Observatory is detecting and studying those rare particles and revealing the enigmas of their origin and energy distribution [11].

Cosmic rays are charged particles, which constantly hit earth. When those particles reach the atmosphere they collide with other particles and produce cascades of secondary particles called an "extensive air shower."

Cosmic rays with energies above 10^{19} eV arrive on earth at a rate of only 1 particle per km^2 per year. The most interesting ones, having energies of above 10^{20} eV, have an estimated arrival rate of just 1 particle per km^2 per century. In order to collect a sufficient number of events a very huge area of detectors and a long observation time are required.

The Auger Observatory is a "hybrid detector" measuring cosmic ray showers with two independent methods. The water-Cherenkov detectors, which are also referred to as surface detectors (SD), are a ground-based technique and detect particles in the shower when their electromagnetic shock waves produce Cherenkov light [19] during the interaction with water in the SD. If such a cosmic ray shower strikes in a SD, Cherenkov light is produced, which is detected by photomultiplier tubes mounted inside the water tanks. The Auger Observatory consists of 1,600 SDs, spaced in a triangular grid of 1.5 km. In total the corresponding detector area is about 3,000 km^2, roughly 30 times the area of Paris [11].

The FDs are the second method of observation and measure the longitudinal distribution of cosmic rays emitted by the excited molecules in the air along the shower tracks. Therefore the Auger Observatory consists of four telescope buildings (FD Sites), each one with six telescopes, where every telescope covers a view of $30° \times 30°$. The measurement of data with the telescopes is done by the so-called front-end crates, which are specialized hardware devices developed for the Auger Observatory. The complete real-time behavior of the measurement with the telescopes is handled by the front-end crates, as they are developed especially for that purpose. To cover the complete atmosphere above the SD area, the FD Sites are arranged at the borders of the SD area (see Fig. 31.1).

The whole Auger Observatory consists of several more systems than only FD and SD. For example, a LIDAR network is deployed for the measurement and online monitoring of the atmospheric optical parameters [8] and a ballooning station for atmospheric monitoring exists.

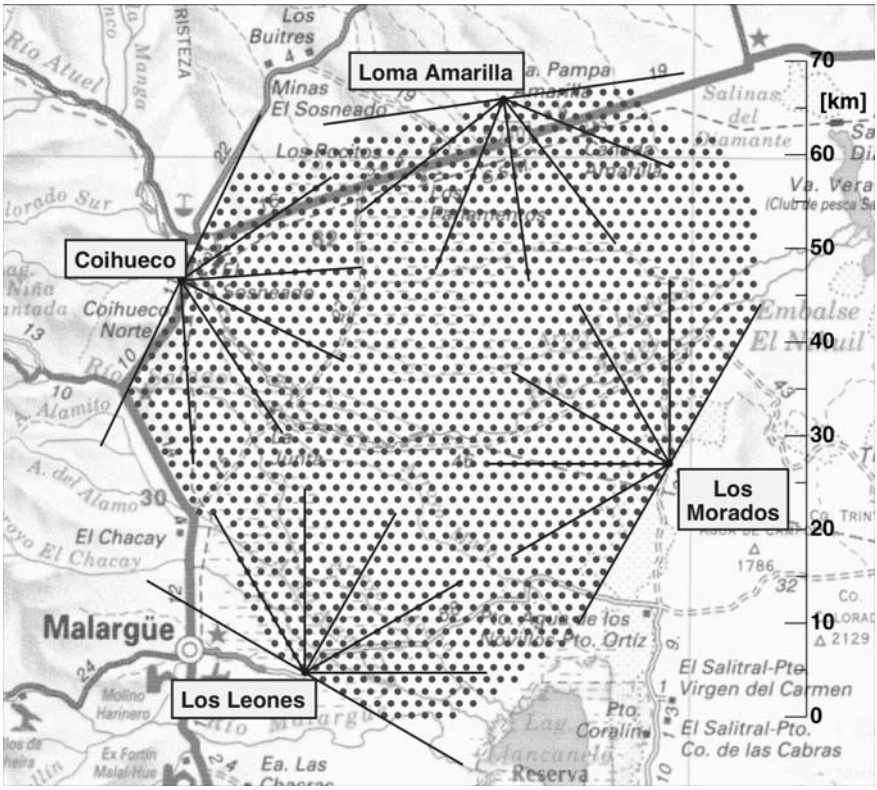


Fig. 31.1 The whole detector area for measuring cosmic ray showers, which is currently deployed in Argentina. The *dots* mark the surface detectors positions and the *lines* show the field of view of the 24 fluorescence detectors [11]

2.2 AugerAccess

As already said the Auger Observatory is located in a remote region far from the participating research institutions. The presence of scientists and technicians from the collaborating institutions on site is currently necessary during the phases of installation and commissioning of the detectors. However, in the long term it will be very difficult to maintain such a level of involvement on site when scientists will be very busy with data analysis at their home institutions all over the world.

The main goals for AugerAccess are the improvement of the communication link between the Auger Observatory and the international networks and the remote monitoring and remote control functionality of the software to provide access to the running experiment for the community worldwide [10].

At the moment the access to the central campus in Argentina for the "outside world" is blocked by a firewall, allowing only very few developers to connect from outside. To allow the access for the Auger collaboration, methods have to

be provided that only authorized users can access. The Grid Security Infrastructure (GSI) [14] of the Globus Toolkit 4 (GT 4) [16] is the chosen technique to provide the security requirements. The main features of the GSI [14] are as follows:

- Security across organizational boundaries, prohibiting a centrally managed security system.
- Support of "single sign-on" for users of the Grid, including delegation of credentials for computations that involve multiple resources.

To provide the GSI, a GT 4 Grid Services Container has to be installed at the central campus and the firewall must be opened for communication with the Grid Services Container. Every user operating with the remote system needs a valid X.509 certificate from a certificate authority (CA), containing information to identify and authenticate the user over the GSI by comparing the users X.509 certificate and the known CA keys. Therefore the user firstly must be authorized from an administrator to operate with the GT 4 installation. If a user or a CA is unknown the request will be rejected and the access denied. Worldwide official CAs exist in every country, signing personal certificates only after checking the identity of the potential users [19].

2.3 Testbed

During development of the new software components with remote functionality a testbed is required, as the original system in Argentina is already in production during deployment. The requirements for the testbed are as follows:

- The testbed must have similar structure as the original system to simulate the interaction of the different sub-systems of the observatory.
- The used components must be similar in the number of computers and network hardware. The computers have to be connected to the network as in the original system to simulate similar interaction between them.
- The software for DAQ and Slow Control must be installed and usable on the testbed. Also the installed services in the Auger Observatory must run, e.g., a big amount of the computers in Argentina are diskless clients and boot over the network.
- The testbed must be easy to configure and administrate. Also the installation must have a high availability – such as the original system.

Before a decision for a concrete solution of the testbed was done, the existing system was analyzed to choose the best available one. The systems in the Auger Observatory consist of several private networks with different subnet masks (see Fig. 31.3). The FD LAN is the network in the central campus and responsible for the communication with DAQ, Slow Control, and the "outside world." Every FD Site has its own Eye LAN, containing several computers (see Fig. 31.2). The main ones are Eye PC, Slow Control PC, and Calibration PC. On the Eye PC the software

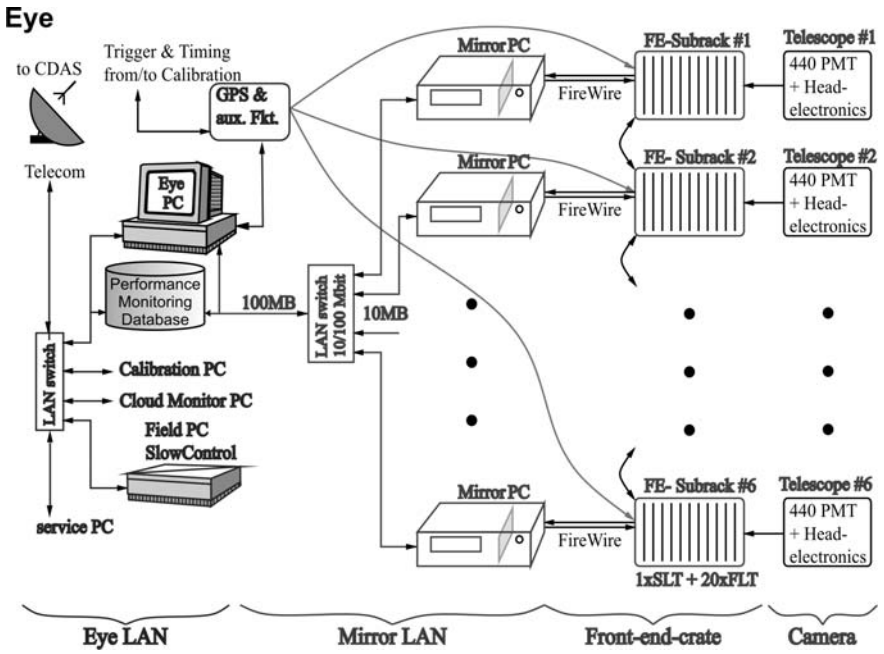


Fig. 31.2 The structure of one FD Site of the Auger Observatory with Eye and Mirror LAN. Behind the Mirror PCs the front-end crates, responsible for collecting the data from the telescopes, visible behind the crates, are shown

for controlling and getting data out of the telescopes is installed. The Slow Control PC is running the Slow Control system and the Calibration PC is for the calibration of the telescopes at beginning and end of measurements. The Eye LAN connects to the Mirror LAN, containing six Mirror PCs each. The Mirror PCs are responsible for the communication with the telescopes and for getting the measured data out of them. In total all the three LANs have nine switches with nine different subnets and 39 computers – only half of the amount of computers in the observatory.

The configuration of the software has similar complexity as the hardware. Starting with a fixed version of the operating system (OS), it goes over several network services like Domain Name System (DNS) [6, 7] or Dynamic Host Configuration Protocol (DHCP) [2] and ends with the network boot of several computers.

As the system with all networks and computers has such a complexity the solution for the testbed with physical hardware is too expensive [13], especially for the necessary administration efforts and with respect to the needed space to deploy the hardware and network components. So the virtualization of the system was the chosen solution, having some advantages. The major ones are as follows:

- If one server has not enough computational power, two or more can be integrated in a cluster to have a better scalability of the VMs. A VM can be deployed on

one server of the cluster and communicate with a VM on another server as if they were on the same server.
- The installation effort for VMs is much less than the installation effort for physical hardware. As already mentioned, the OS is a fixed version on the computers and this is also for the software. A VM must be installed once and is cloneable as often as needed. In the cloned VM only the configuration of hostname and network must be changed, which is a very fast procedure.
- The installation and administration of the VMs can be done from only one client – simplifying their administration.
- The costs of the complete testbed including space and cooling could be minimized to the costs of a virtual system [13].

The convenient testbed is configured once and may be used everywhere in the world. If other members of the Auger collaboration need the whole system – or parts, they can use the existing one or simply a copy. We aim to explore the usability for software testing of the DAQ. For usage of the system, administrators and users must learn how to use and configure the VMs and to handle the problems occurring by using this technology. To get an idea of the complexity of the FD system, Fig. 31.3 shows all networks and the connection of every computer needed for measuring data with FD in Argentina.

3 Architecture

For the virtualization of the Auger observatory a powerful server and software to run the VMs is necessary. Therefore an IBM Blade server with two Dual Core Xeon CPUs and 16 GB of memory is used. On the server VMware ESX Server [18] as hypervisor for abstracting the physical hardware and virtualization software is installed. The ESX Server was chosen, because for this product the handling of a huge amount of VMs is no problem as the server was designed for this kind of applications.

The architecture of the virtualized network is nearly the same as in the original system in Argentina, with a small difference concerning the networks. In Argentina every FD Site is connected to the central campus over a radio link, a requirement which is not possible to simulate with the ESX Server. The possible solution is the deployment with virtual switches. A virtual switch is the virtualization of a hardware switch and for the connection of virtual network cards to virtual networks responsible. The small disagreement with the not to virtualize radio link is negligible, as it makes no difference for the software how the network is connected – only the connection and communication protocol is important.

Also some additional behavior of the original system cannot be virtualized, as the ESX Server does not support the functionalities. For example, in the original system some computers reboot intermittently because of frequent power outages in Argentina. Another example is potential network problems, which can result in slow and unreliable networks, where some of the sent packets are lost

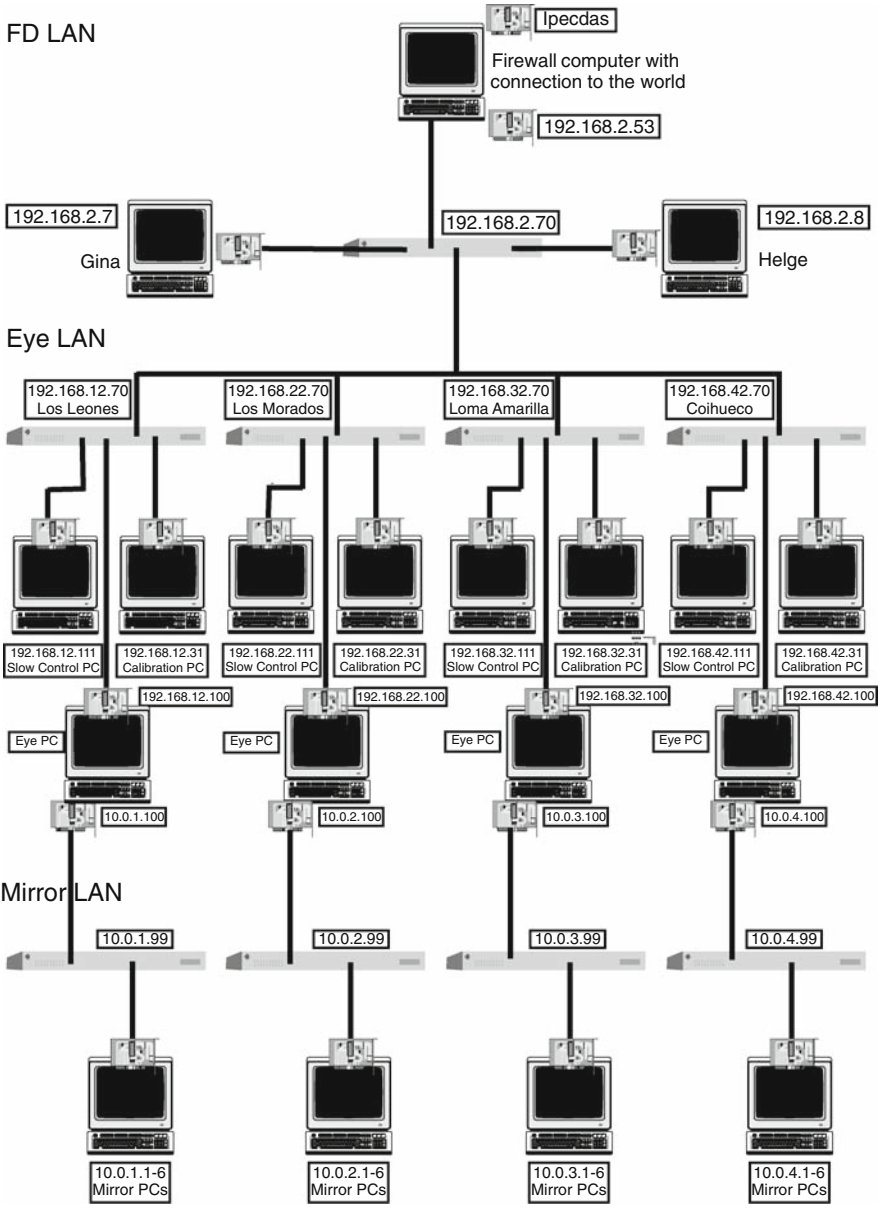


Fig. 31.3 The scheme of the network needed for FD. Every computer and the connection to the network with switches and network cards are visible

during transport. The simulation of the network failures is possible, but only with a lot of configuration for the VMs. As such a behavior is not needed it was not configured.

3.1 Virtual Machines

As first step for the virtualization every existing computer of the FD system must be deployed as a VM on the ESX Server. Therefore one VM as template for all VMs was installed. The creation of a VM on an ESX Server is very simple with the provided client of the server installation. Only the name and the hardware (like network cards or hard disks) to virtualize must be indicated. After creation the VM is bootable like a normal computer, also from a CD-ROM, so that the installation of an OS is straightforward.

Currently on the ESX Server three computers of the FD LAN, visible in Fig. 31.3, are deployed. These are Ipecdas, Helge, and Gina. Ipecdas is the gateway for all computers in the private network to the Internet and contains the firewall responsible for denying access to the Auger networks for the "outside world." Gina is responsible for the interaction with DAQ and Helge for the interaction with Slow Control. Los Leones and Los Morados as two examples of the four FD Sites are deployed at the moment. Each of them consists of Eye PC, Calibration PC, Slow Control PC, and six Mirror PCs. The FD Sites Loma Amarilla and Coihueco are not deployed on the testbed, because two FD Sites are sufficient to fulfill the current requirements and already provide the needed complexity of the structure for development of the new software components. If the absent FD Sites are needed, the deployment of them is straightforward by using the created template of the VMs.

After creation the necessary OS in the template VM was installed. The FD systems consist of three different OSs. The Calibration PCs are running Red Hat Linux 9 [12], the Slow Control computers and Helge using Windows 2000 [4], and the rest of the VMs are running SuSE Linux 9.1 [9]. As three different OS versions are needed the creation of three different templates, which can be used to clone the corresponding VMs from the accordingly template, is necessary. In the cloned VMs only the network identification and hostname must be changed. The name of the different VMs and their location in the network can be seen in Fig. 31.3.

3.2 Virtual Networks

To simulate the communication of the VMs as in the original system, they must be connected to virtual networks. This is done by virtual switches. To connect a VM to a virtual switch, the VM needs a virtual network card – nearly the same procedure as if physical hardware is used.

Each VM has at least one virtual network card and some VMs have more than one to virtualize the whole network the correct way. For example, the Eye PCs have two network cards. This is necessary because the Mirror LAN has its own subnet and for the communication of the Mirror PCs with the DAQ (running on the Eye PCs), it is necessary that each Eye PC has two network cards, routing between both subnets. The network cards are also required, because the Mirror PCs are diskless clients booting their file system over the network from their corresponding Eye PC.

There exists one difference between the network shown in Fig. 31.3 and the deployed virtual network: The default gateway for every computer in the Eye LAN has 192.168.*.70 as IP (Internet Protocol) address, but the setup of a default gateway for an ESX Server virtual switch is not possible, as this option is not available in the configuration of a virtual switch. For the right behavior of the testbed a communication of all VMs in the FD LAN with the worldwide networks is important. The only way to provide this is to add virtual network cards in the Ipecdas computer (see Fig. 31.3), having 192.168.*.70 as IP address. The default gateway in the network settings of each VM in the FD LAN is 192.168.*.70, so the communication with the worldwide networks is possible, if the Ipecdas computer is acting as router. To act as router the Ipecdas computer must provide NAT (Network Address Translation), which is very easy to realize in a Linux OS, as IP-Tables (program iptables under Linux) [17] are usable. NAT means rewriting of source and/or destination addresses of IP packets as they pass through a router or firewall [3, 19].

3.3 Services on the network

For the correct behavior of the virtualized network, it is important that all services running in the original system in Argentina are also running in the virtualized network. Otherwise the usage of the testbed for development purposes is not possible.

For the communication of the VMs, the software, and the users, the usage of IP addresses during operating with the system is not comfortable. A better way is to use a name when addressing a computer. Therefore, a DNS installation is provided in the virtual network.

To have a working lookup for the complex virtual network the installation of several DNS servers on different computers is important. First, on Gina (see Fig. 31.3) runs the DNS server for the FD and Eye LAN, allowing name resolution of all computers available in both LAN segments. The communication of the Mirror LAN with Gina is not possible, as the Mirror LAN has an own subnet. Therefore a DNS server on every Eye PC is running, providing name resolution for the Mirror PCs in the respective Mirror LAN.

Most VMs have static IP addresses in the network configuration of the corresponding OS. Nevertheless, this is not possible for the Mirror PCs, because they are diskless clients and mount their root file system over the network. Therefore, the IP addresses of the Mirror PCs are retrieved by DHCP. DHCP is used by network clients to allow requesting and obtaining an IP address from a server, which has a list of addresses available for assignment [2, 19]. During boot up of a Mirror PC, the IP address is obtained from the DHCP server running on the corresponding Eye PC.

After resolving its IP address the network boot of a Mirror PC can be executed. This is provided by the Preboot Execution Environment (PXE). PXE is a specification for booting an OS from a network server using a network ROM (read only memory) that conforms to the Intel PXE specification [1]. The scenario for booting over PXE is very simple:

1. An IP address is requested from a DHCP server.
2. A Trivial File Transfer Protocol (TFTP) server runs on the computer with the DHCP server. “TFTP is a very simple file transfer protocol with the functionality of a very basic form of FTP” [19] and delivers the configuration (what kernel image to boot) to the booting diskless client.
3. The location of the kernel image for booting the diskless client, as well as the address of the TFTP server, is declared in the DHCP configuration.
4. The file system(s) to mount from the booted kernel image must be exported via Network File System (NFS). NFS allows the access to files over the network as easy as if they were on a local disk [15, 19].

At least the needed file systems for the Mirror PCs are exported via NFS from the Eye PCs. These are the root file system, the home directories for the users of the Mirror PCs, the directories for the measured data with FD, and the directory for the log files produced by the software.

4 Status and Results

As a consequence all the VMs and virtual network devices required for the testbed are deployed and configured. The configuration of the virtual network environment is setup; hence all the virtual network cards are connected to their corresponding virtual switches. Furthermore, the substantial network services, as already mentioned in Section 3.3, have been configured and are up and running. As a consequence it can be stated that the deployed testbed offers very similar capabilities as the original system deployed in Argentina, despite the following minor differences:

- Variability of the radio link between the central campus and every FD Site;
- The integration of the default gateway in every subnet via a separated virtual network card;
- The NAT of Ipecdas (see Fig. 31.3) is managed by a VM with IP-Tables, instead of a router.

Nevertheless, these differences are not relevant for the deployed software, since the physical transport has no impact in this case. The correct transmission of the sent packages is the only requirement.

The software needed for measuring data with FD is already installed and the DAQ system is running without errors. However, the DAQ is producing only “Dummy-Data,” which is based on real data, including different time stamps. This is necessary due to the unavailability of a telescope and a front-end crate.

In regard to the Slow Control system the two PCs for Los Leones and Los Morados FD Sites are installed and can be accessed in the same way as in the original system in Argentina, though any invoked operation will have no further effects because of the missing physical hardware. For example, the “open shutter” procedure will cause an error, because of the missing responses from the corresponding shutter hardware.

So, the testbed is usable for the development of the remote software. Another advantage is that the testbed is available for other partners of the Auger collaboration if needed. They can use the existing one or simply a copy.

The virtualization also has some limitations. Especially, hardware failure scenarios cannot be simulated, e.g., a sudden shutdown of a device because of power outage. If needed, such a behavior has to be simulated by manually switching the corresponding VMs. The telescopes of the Auger Observatory also cannot be of simulated, since they consist of special hardware of which no virtualized component is available. Nevertheless, real telescopes can be connected by configuring the ESX Server to add physical hardware to the virtual infrastructure.

4.1 Faced Problems

Several problems occurred during deployment of the virtual system, most of which have been solved. One problem was the time synchronization of the VMs. Initially we used the NTP (Network Time Protocol) [5] client of the VMs to synchronize time with a public available time host. However, the VMs time differed strongly from the real time after uptime of a few hours. The reason is that the VMs are sent into a sleep mode by the ESX Server when they produce no system load. After resuming the VM the NTP does not recognize the gap between the current system time and the actual time, which leads to the discovered time difference. The solution is to use the time synchronization offered by the VMware tools, which requires additional kernel parameters during start up of the VM.

Another problem arises from the fact that all VMs are stored in a storage area network (SAN). A SAN provides storage devices to remote computers, e.g., disk array controllers, to server systems so that the operating system recognizes the SAN storage device as a local storage device [19]. If the response time of the SAN exceeds a certain timeout value due to heavy system load, the VM changes the state of the mounted file system to "read only." In this mode the VM is not able to work in an appropriate manner and therefore has to be rebooted. This is a known problem by VMware and a patch is already provided. However, the application of this patch is not possible in the current configuration of the used operating systems, since the patch is only available for newer Linux kernel versions and the operating systems deployed in Argentina use older versions of the Linux kernel. We considered this problem as negligible, as it occurs only once in 2 or 3 months, which exceed the time for development and testing by far.

4.2 Remote Client

As the testbed is fully deployed with all required VMs, the services for communication in the testbed are configured and the existing Auger software is able to

"measure" data, the testbed can be used for development of the remote client to control the DAQ and Slow Control systems of the observatory.

The remote client is a service-based software and uses the GSI of the GT 4 for communication with (see Fig. 31.4) the respective software inside the observatory. The security features are only needed for the access to the system from outside the Auger campus. No additional security features are needed inside the campus. A remote user can only connect to the system after the authorization and authentication via his X.509 certificate is permitted. This is only possible if the X.509 certificate is known by the GT 4 Grid Services Container the user is communicating with. To control the system the GT 4 Grid Services Container is extended with several Grid services, whereas these services are responsible for performing operations with the DAQ. All operations are performed via the remote client, which is also responsible for the authorization and authentication of the user.

After the access of a user is permitted, the so-called to accessed Grid service performs the according operation on the Eye PC of the corresponding telescope building. Therefore the operation to perform, which is declared in a special protocol, is sent over a SOAP message to the data acquisition software on the Eye PC, performed, and the remote client is notified about the status of the

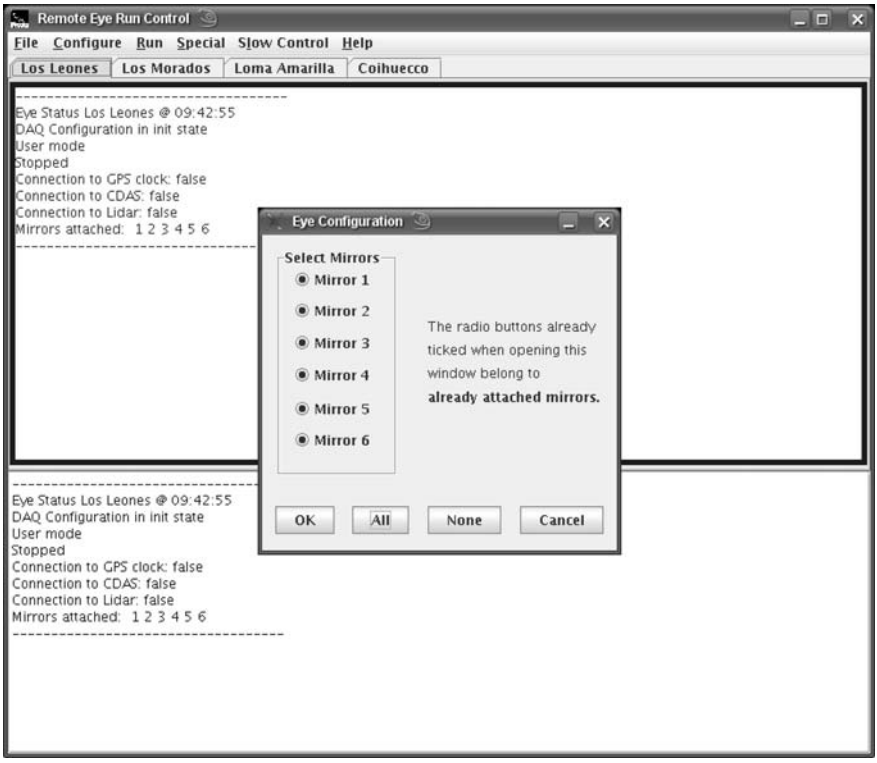


Fig. 31.4 Screenshot of the remote client with opened dialog to add or remove telescopes. You can see the current status of the telescope building in Los Leones

performed action. In order to retrieve, evaluate, and perform the SOAP messages the existing data acquisition software has to be adapted, because at the moment the data acquisition software is only controllable over a graphical user interface inside the central Auger campus. This graphical user interface at the moment implements the FD communication protocol.

The Slow Control system is already controlled over SOAP messages. This would allow the development of a uniform access method to the DAQ and Slow Control system. The problem hereby is that the Slow Control system is polling the status every several seconds and in order to develop a successful system the remote software has to poll the status again, as the server responsible for the status is not accessible. This would double up the communication and the network load and is not advisable. So the decision is to use a GSI SSH server and a port forwarding of the Slow Control server to be able to access the Slow Control system from worldwide. The opening and closing of the SSH connection is possible within the remote client, making the interaction with the Slow Control system with only one performed operation possible.

5 Discussion and Future

The visualized system for virtualization of the FD network as testbed for the development of remote control and monitoring software is a very helpful solution and much easier to use than an implementation with physical hardware. The virtualization is very flexible and easy to extend – if necessary. Just clone and connect a new VM or even an entire FD Site to the virtual network. The virtualized solution also has some restrictions, e.g., not to simulate radio links. However, these problems are not considered for the development of the new software components.

The virtualized system fulfills all our requirements and the current DAQ system is already installed and running without any problems. The development of new software components has already begun and so the testbed is in productive use for its destined purpose.

After the new software is completely developed and tested the currently running version in Argentina can be replaced with the new one. The remote functionality will only be available after an upgrade of the communication link to the Auger Observatory.

Another possibility we are exploring is to run the whole FD system in Argentina with VMs. Therefore the hardware must be virtualized, so that VMs can be executed. To virtualize the hardware, software like the ESX Server or XEN [20] can be used. Such a solution would have some advantages: If a new software version must be installed, the whole VM can be configured and tested on our testbed. If all tests are passed successfully, the currently running VM in Argentina could be halted and the "new version" started. This would offer a very fast solution for changing the software and even offer a very easy fallback solution in the case of any error. The old VM simply has to be booted and the old system would be up and running, which is done within a few minutes.

6 Conclusions

The virtualization is a very powerful and flexible solution for deploying a complex testbed as the computer and network infrastructure of the Auger Observatory. The deployment of a virtual testbed is easier than to deploy a second infrastructure using real hardware. The advantages for virtualization are as follows:

- The virtualized infrastructure is very easy to extend with new components – e.g., cloning existing virtual machines and integrate them in the system.
- The administration of the whole system can be done by using one single client from wherever installed. So, changing the system configuration or reboot a VM is very easy.
- The host server can be shared with other projects, if the server has enough resources or several servers can be added to a cluster to provide the needed resources.
- After the lifetime of the project the server can be used for other projects, and also the backup of VMs is easier than to depollute physical hardware.

The virtualization not only has advantages, as it is a new technology and users have to learn how to use the provided features. One example is the time synchronization with the provided VMware tools and not over NTP. However, it is not difficult to learn the new technology and the advantages outweigh the disadvantages.

Acknowledgments The project "AugerAccess: Integrating Auger Observatory and European Research Institutions into a worldwide Grid" was funded by the Commission of the European Communities in the Sixth Framework Program with the contract number 026457.

References

1. H.P. Anvin. *PXELINUX – SYSLINUX for Network Boot*, Jan. 2008. http://syslinux.zytor.com/pxe.php
2. R. Droms. Dynamic Host Configuration Protocol. Technical Report RFC 2131, Mar. 1997.
3. K.B. Egevang. The IP Network Address Translator (NAT). Technical Report RFC 1631, May 1994.
4. Microsoft. *Microsoft Windows 2000*, Jan. 2008. http://www.microsoft.com/windows 2000/default.mspx
5. D.L. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. Technical Report RFC 1305, Mar. 1992.
6. P. Mockapetris. Domain Names – Concepts and Facilities. Technical Report RFC 882, Sep. 1983.
7. P. Mockapetris. Domain Names – Implementation and Specification. Technical Report RFC 1035, Nov. 1985.
8. R. Mussa, S. Argiró, R. Cester, M. Chiosso, A. Filipčič, M. Horvat, J. Matthews, M. Mostafà, M. Roberts, G. Sequeiros, D. Veberič, D. Zavrtanik, and M. Zavrtanik. The LIDAR systems for atmospheric monitoring in Auger. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 518, Feb. 2004.
9. Novell. *Linux Operating Systems*, Jan. 2008. http://www.novell.com/linux/
10. Pierre Auger Collaboration. *Auger Access*, Jan. 2008. http://www.augeraccess.net/

11. Pierre Auger Collaboration. *Pierre Auger Observatory – Studying the Universe's Highest Energy Particles*, Jan. 2008. http://www.auger.org/index.html
12. Red Hat. *Red Hat Website*, Jan. 2008. http://www.redhat.com/
13. S. Seetharaman and K. Murthy B.V.S. Test optimization using software virtualization. *IEEE Software*, 23(5):66–69, 2006.
14. B. Sotomayor and L. Childers. *Globus Toolkit 4: Programming Java Services*. Morgan Kaufmann, 2005.
15. Sun Microsystems. NFS: Network File System Protocol Specification. Technical Report RFC 1094, Mar. 1989.
16. The Globus Toolkit Alliance. *Welcome to the Globus Toolkit Homepage*, Jan. 2008. http://www.globus.org/toolkit/
17. The Netfilter Project. *Netfilter, Firewalling, NAT, and Packet Mangling for Linux*, Jan. 2008. http://www.netfilter.org/
18. VMware. *VMware Infrastructure – ESX Server*, Jan. 2008. http://vmware.com/products/vi/esx/
19. Wikipedia – The Free Encyclopedia. *Welcome to Wikipedia*, Jan. 2008. http://en.wikipedia.org/
20. XenSource, Inc. *Welcome to xen.org, Home of the XEN$_R$ Hypervisor, the Powerful Open Source Industry Standard for Virtualization*, Jan. 2008. http://xen.org/

Part VI
Metrology Issues

Chapter 32
Challenges and Design Issues in a Distributed Measurement Scenario

L. Benetazzo, M. Bertocco, G. Gamba, and A. Sona

Abstract In the last few years wireless sensor networks (WSNs) have gained an increasing interest in the scientific community, due to their wide variety of applications and technical issues to be faced. Many challenges and design issues arise in this ad hoc scenario. In this chapter, some concepts are discussed having in mind a twofold point of view: the telecommunication researcher or GRID designer perspective, and the metrologist or instrumentation and measurement designer one. Such mixed view allows pointing out aspects that are somewhat seldom considered and that instead could provide interesting results in both the above research areas.

Keywords Distributed measurement systems · Wireless sensor networks · Ad hoc network · Design issues

1 Introduction

Recent advances in telecommunications have had a noticeable impact in the measurement field. New scenarios are emerging and new concepts of "measurement system" are under discussion in the scientific community.

The concept of measurement instrument has assumed different meaning in the last few years. A first rather recent change in the architecture has been observed when bulk and almost analog measuring equipment has been replaced by computer-based instrumentation embedding one or more processors in order to provide supervision functionalities to analog acquisition channels, post-processing of digitally acquired samples, and presentation of some kind of display. A second important recent change has been introduced by the so-called virtual instrument approach, where the pre-post-digital processing and coordination of analog parts including A/D or D/A channels has been performed by a computer [2]. Moreover, some

L. Benetazzo (✉)
Department of Information Engineering, University of Padova, 35131 Padova, Italy
e-mail: luigino.benetazzo@unipd.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_32,

simple-to-use programming environments have been developed (e.g., National Instruments LabVIEW [21] or Agilent technologies VEE [27]) providing a virtual instrument whose features actually depend on the developed software application. The concept of virtual instrument has been then extended by noticing that network communication schemes could be advantageously exploited in order to add interesting functionalities. In this context, client/server architectures have been proposed in order to provide the user the opportunity to use a "remote" virtual instrument, possibly shared with other users according to some policy schemes [7, 6, 13]. The creativity of designers and scientists has not been obviously limited to the trivial usage of a digital storage oscilloscope through a TCP/IP connection. Instead, the scientific community quickly understood that communication infrastructures already developed for other purposes could be advantageously adopted for measurement applications [5]. The GRID architecture is one example, while the distributed system paradigm (in the computer science sense) is a more general framework that now researchers are considering as a fundamental component for the development of distributed measurement applications. Moreover, the availability of low-cost and network-aware sensing devices has recently allowed the proposal of distributed sensor networks. As a matter of fact, sensor networks now may benefit from IEEE standards (i.e., the IEEE 1451 series of standards [17]) that constitute a very useful scheme in the development of new applications. In the last few years many researchers have understood the advantages from using wireless technologies in order to provide communication capabilities to sensors so that Wireless Sensor Networks (WSNs) are now a "hot topic" in a number of research and industrial areas ranging from telecommunications to automatic control, computer science, and measurement fields.

The adoption of wireless technologies, as opposed to "wired" solutions, is now inducing a number of issues in measurement systems. First, "wireless" adds an improvement to a measurement system comparable to the one that gave the adoption of the virtual instrumentation paradigm to stand-alone physical measuring equipment. For instance, mobility associated to such a technology allows the development of applications that were just not possible until now. Second, a measurement system, because of its intrinsic nature, induces requirements to a communication infrastructure that usually are not considered when the network itself is designed or conceived. One example is constituted by time synchronization in a networked context, where the point of view of a metrologist and the one of a system manager of a computer network are considerably different.

In this chapter, wireless networks technologies best suited for the development of WSNs are discussed, with a particular emphasis to ad hoc networks, trying to melt together the point of view of a telecommunication and computer science engineer and the one of a metrologist. This attempt provides suggestions to both people, and possibly new criteria for the choice of characteristics or architecture features. Also challenges induced by such an integrated point of view are highlighted, in the attempt to stimulate the reader toward possible research topics.

2 Topology Issues

Distributed Measurement Systems (DMSs) are designed having in mind quite different applications. Hence, the optimal network architecture best supporting a DMS and its management strategy may considerably vary. For instance, a time synchronization having an order of magnitude of one millisecond between measurement nodes is required for the evaluation of power quality [10, 12]. In a monitoring system collecting environmental chemical quantities, instead, time synchronization is so mild constraint that in practice any network system will be able to satisfy. Again, for land surveillance purposes time synchronization is in practice almost not required, but instead, the reliability of nodes and network connections is a key point. In a similar manner, the bandwidth required may considerably differ if one considers again the power quality assessment application, or instead plans to design a distributed system that collects somewhere real-time data sampled at high speed for subsequent post-processing at some far located high-speed computing facility.

This observation may lead to the erroneous conclusion that in theory any network topology may be adopted as far as the previous time/bandwidth constraints are satisfied. As a matter of fact, the different available options have, from the measurement point of view, additional different advantages and drawbacks, briefly discussed in the following for some fundamental network topology (see Fig. 32.1).

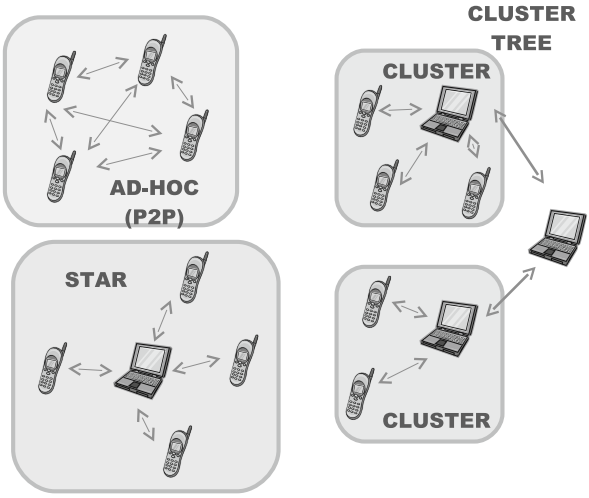


Fig. 32.1 Network topologies in WSNs

Ad hoc topology: In ad hoc networks, nodes can directly communicate with each other; nodes may act as routers, forwarding messages over multiple hops on behalf of other nodes. An intrinsic feature associated to this topology is its "stochastic" nature, namely the lack of a priori knowledge of some network parameters such as the number of nodes or their associated routes, and the impossibility of determining in advance the network status at a give time instant. This topology has the advantage,

from the measurement perspective, that estimation techniques based on the so-called "consensus" [14] are highly simplified, e.g., convergence may be much quicker with respect to other topologies. Moreover, such topology prevents the "single point of failure" problem due to its redundancy: an interesting property that ensures measurement will not be lost along the path to its final destination. Ad hoc networks are scalable, which in turn implies the ability to let the number of sensing nodes considerably grow without additional efforts. On the other side, ad hoc networks require nodes with a sufficient computing and transmitting power, i.e., the nodes will need adequate hardware resources, and hence may require a non-negligible (electrical) power to let the network work. The absence of a coordinating infrastructure may lead to some limitations in the geographical extension of these networks. In practice, the adoption of this topology may assume a limited interest for the case of data collection and post-processing from geographically distributed sensing nodes (e.g., landscape temperature and pressure data for a weather forecast analysis).

Star topology: The star topology is highly hierarchical, and a single hop communication exists between a central base station and several surrounding nodes. In comparison with ad hoc networks, this topology can be seen as a rather "static" one, and hence less flexible. Among the advantages of the star topology for measurement applications, one can note the possible low requirements to sensors in terms of computing power and energy consumption associated to transmission. With respect to ad hoc solutions, this topology allows satisfying real-time constraints in an easier manner: in fact, it is often considered (say in IEEE-802.15.4 solutions [28]) as a reference topology for industrial applications where measurements meet automatic control [8]. Again, the star topology is the "natural" choice for distributed monitoring systems that are coordinated by one central unit. Nevertheless, this topology offers worse scalability features with respect to ad hoc architectures, and hence is best suited for the case of a limited number of sensors where the actual position can be chosen at design time.

Cluster-tree topology: In cluster-tree topology many star networks are connected in a tree where a super-coordinator rules the whole network and a prefixed number of base stations act as relay nodes (cluster heads). Cluster heads can be elected and chosen in a time-varying way. Many wireless sensor networks use an infrastructure-based topology, but often the infrastructure is dynamically created and not prefixed. This latter topology is in practice an improved version of the star one, and hence presents similar advantages and drawbacks. In practice it balances an increased complexity with scalability, even though the hierarchy that it implies can be exploited in order to simplify the management of the network itself. On the other side, it is still exposed to single point failures, and hence its usage may be questionable when the loss of measured data is a detrimental fact.

3 Design Issues

Wireless Sensor Networks, in any of the topologies already recalled, present a number of design issues. A deep discussion is beyond the scope of this chapter. Instead,

some design aspects can be viewed in a different manner from a network designer and a metrologist. In the following some critical aspects will be pointed out underlying such differences.

Deployment: A WSN must be physically created by inserting a number of sensing nodes in a specific environment. The *deployment* of sensor nodes may be in random fashion or manually installed in well-known chosen points. Usually, in a telecommunication scenario the nodes are very similar to one another. Instead, in a measurement context, the nodes may differ in a way so that a plug and play feature is highly recommended.

Mobility: *Mobility* is a second important issue. For example, the net can be immobile, partly mobile, or highly mobile; varying the speed; the number of nodes involved; and the pattern. Mobility can be both active, if nodes can move, or passive, if the environment moves (i.e., sea monitoring or wild animals-attached nodes). Mobility has a high impact on the expected degree of network dynamics, influencing networking protocols and distributed algorithms. Despite the high interest for mobile technologies for telecommunication study groups, in a measurement context the term mobility refers almost to "mobility of the device under test," which can be, for instance, some kinds of product along a manufacturing chain of an industrial plant, a living animal, etc. Less frequent is the case of a "mobile engineer performing a test," even though some interesting exceptions are found. One is the case of supervision of an industrial plant where wireless users' mobility stems for an easy way for interfacing with large production machines. Another example comes from the distance learning series of applications [3], where students' mobility with respect to training remote labs in a wireless access scenario could be an intriguing feature.

Another aspect where mobility is seen from a different perspective by the metrology and telecommunication approaches is when considering the associated concept of coverage area. In theory, mobility aims at providing physical access to a communication facility worldwide. Instead, the dimension of the coverage area in a measurement context is limited by the physical phenomenon under investigation. Even though rather large examples can be found, e.g., in the case of some environmental quantity spanning a city or a regional district, many applications exhibit a limited extent, so that radio coverage may not need all the complication associated to cell handover and so on.

Moreover, it is noticed that a sensor network designed to cope with environmental monitoring often needs wireless connections just for the sake of deployment simplicity rather than for some actual hard constraints. There are instead some emerging applications where "mobility" is associated to sensors inside large machines of industrial plants, where wireless access means a considerable simplification in cabling and hence cost reduction of the machines themselves. Unfortunately, in such applications reliability is a very hard constraint, and designers are almost concerned about interference that potentially may severely corrupt communication links causing hazards. A final interesting aspect associated to mobility is handover. In many telecommunication applications a short failure associated to a handover procedure may be considered an acceptable drawback as far as it means just the corruption

of one speech word, or one video frame, etc. Instead, in measurement applications data loss due to handover means accuracy decrease; in many cases, this is a price that one would like not to pay. From this consideration, the stimulus for better handover schemes may arise.

Heterogeneity: In a WSN, *heterogeneity* has interesting implications. But, in practice, research works tend to standardize as far as possible the nodes in such a way that they can be considered almost equal. In measurement applications this is an assumption that is sometimes very hard to satisfy for many good reasons. To quote just one among the various, the nodes can be similar from the functional point of view (e.g., A/D converters), but the way they exchange data with external controllers may highly vary because of a lack of standardization. As a consequence, a practical application may easily require the development of "middleware" layer that let measuring devices keep pace with their remote controllers. Despite the presence of some de facto standards written having in mind measurement applications [25, 18] or scientific proposals [4], the need is felt of some kinds of standardization at the application layer, in a similar manner to the one offered by the widely known HTTP protocol for the data exchange of web contents. The latter may be an interesting research issue.

Energy: *Energy*-constrained networks constitute a large research topic [24, 16], assuming different viewpoints:

- *Signal processing viewpoint*: optimize distributed detection and estimation network tasks while minimizing the use of communications.
- *Communications viewpoint*: support network-specific goals while minimizing idle listening, network setup, and network maintenance.
- *Systems viewpoint*: exploit low-power hardware and external assets to the greatest extent possible.

In the above non-exhaustive list, one really interesting item in the perspective of measurement sciences is the signal processing one, with the additional consideration that an energy constraint implies practical limitations of hardware resources for sensor calibration tasks or compensation of systematic effects: this in turn means accuracy reduction. Hence, energy optimization can be viewed as the search for compromise solutions that optimize overall accuracy rather than parameters like packet latency.

Synchronization: *Network synchronization* is yet another large research area tightly linked with WSN applications. In fact, signal processing, energy-aware transmission, cooperative sensing, detection, and estimation need some levels of synchronization between sensor nodes. The degree of synchronization could vary from coarse synchrony of events (many nodes detect the same event) down to analog-to-digital converter (ADC) sample time accuracy. In addition to signal processing tasks, synchronization enables scheduling by the medium access control (MAC) layer, allowing low-duty cycling especially in energy-consuming hardware, as transceivers. Each sensor node usually has at least one crystal oscillator used for the clock generation. The clock is characterized by a nominal frequency (typical watch clock uses a 32 kHz quartz) and an accuracy expressed in parts per million

(PPM, for example, 1 PPM implies an accuracy of 1 μs after 1 s). The accuracy is influenced by humidity, temperature, and aging. For example, in [24] several oscillator types are compared: a clock oscillator provides a 200 PPM accuracy consuming 1 μW, a temperature controlled crystal oscillator (TXCO) provides 6 PPM accuracy at 6 mW and one GPS achieves a 10^{-2} to 10^{-5} PPM accuracy consuming 180 mW. The most accurate oscillators are too expensive and energy consuming for typical WSN applications. Many algorithms for a global clock estimation all over the network have been proposed, relying on both distributed or hierarchical protocols [20, 19]. The first approach tries to converge to a virtual clock estimation, while the second approach tends to distribute a central clock reference. Synchronization messages must be passed at a minimum rate to assure a chosen level of synchrony, given the oscillators accuracy and environmental conditions.

One should note that time synchronization performance may assume a central role in measurement applications, where the correctness of extracted information directly depends on time events. In many cases data that is not acquired "simultaneously" by sensors become useless, where "simultaneous" means within a time skew of a predetermined amount. In this context it is interesting to compare the telecommunication literature to the instrumentation and measurement one: the de facto standard LXI is one noticeable example [22], while the concept of "trigger" coming from the measurement sciences provides a new perspective to a WSN.

Localization: In addition to synchronization, a second fundamental feature in WSNs is *node localization*. If a random deployment has been used, nodes are not aware of their local relative position nor of the absolute one (in a Cartesian space). Localization is a basic assumption in tracking applications. *Passive* localization uses the receiver signal strength indication (RSSI) embedded in almost all transceivers or acoustic (ultrasonic) signals to triangulate the source, or uses angle of arrival (AoA) or time of arrival (here fine synchronization is required). *Active* localization uses several beacon sources, usually connected to GPS (high accuracy but also high cost and limited to outdoor GPS coverage). Radio-based geo-location is a long-studied problem, but sufficient accurate geo-location may require a time-bandwidth product that exceeds that necessary for the WSN to operate. One possible solution is ultra wide band (UWB), where the transmission through very short-duration pulses allows a geo-location based especially on time and angle of arrival (very precise synchrony needed). Good results have been achieved with some kind of acoustic sensors, that must be mounted on each node.

Localization assumes the same metrological importance that time synchronization has for applications that aims at estimating some geometrical quantity (e.g., a pressure curve alongside a surface). Despite one could find many proposals that show how to calibrate single nodes of a WSN (one for all the "traveling standard" paradigm [9]), the *calibration* of the whole system is still under discussion in the scientific community.

Routing and medium access control: Typical WSNs are composed by hundreds or even thousands of nodes and thus the medium access control protocol achieves a great importance to share the radio channel among these users. Because of the usage

of battery-supplied devices, the first goal in system design is the maximization of network lifetime, while performance metrics assume less importance. To cope with this objective, communications must be minimized. The *medium access control* and the *routing* mechanisms in a dense network composed of nodes with low-duty cycles are both challenging problems [16, 11, 1].

In particular, routing induces some issues that assume relevance both for a telecommunication engineer and from the measurement point of view.

First, due to the relatively large number of sensor nodes, it is not possible to build a global addressing scheme for the deployment of a large number of sensor nodes as the overhead of identification (ID) maintenance is high. Thus, traditional IP-based protocols may be hardly applied to WSNs. Nodes that are deployed in an ad hoc manner need to be self-organizing. In WSNs, sometimes getting the data is more important than knowing the IDs of which nodes sent the data. Moreover, WSNs are usually data-centric networks in the sense that data requests are based on certain attributes (i.e., attribute-based addressing). An attribute-based address is composed of a set of attribute–value pair queries (e.g., humidity $> 50\%$). These concepts are stressed in GRID architectures tailored for measurement purposes. The IP addressing mode is no longer a key factor: instead the services provided by the GRID or system (hardware, software) components are the interesting building blocks, and the relationships being established among them are a key feature.

Second, almost all applications of sensor networks require the forwarding of sensed data from multiple sources to a particular sink. This, however, does not prevent the flow of data to be in other forms (e.g., multicast or peer to peer). Another interesting aspect that mixes up measurement requirements and telecommunication technologies is the fact that sensor networks are application-specific (i.e., design requirements of a sensor network change with application). For example, the challenging problem of low-latency precision tactical surveillance is different from that of a periodic weather monitoring task. Moreover, by itself the particular routing algorithm considerably affects time synchronization performance. It is interesting to observe that in many measurement research works concerning time synchronization routing is not considered at all, while cross-research coming from experts in both fields may probably produce interesting new results.

Finally, data collected by many sensors in WSNs is typically based on common phenomena, so there is a high probability that this data has some redundancy. Such redundancy needs to be exploited by the routing protocols to improve energy and bandwidth utilization. Moreover, redundancy reduction may affect accuracy, and hence again the metrology point of view could meet the computer science approach in order to improve currently available results.

Reliability: The *reliability* of communication and *availability* of connections both assume interesting meaning in measurement applications. A lost packet, or unavailable connections, imply loss of information at the end side where such information should be used in order to provide an actual numerical measure. Hence, even a limited packet loss determines a degradation of accuracy, namely it has an effect that may be seen as more unacceptable with respect to a sporadic degradation in a received speech.

Testing: Another relevant aspect is *testing*. Sometimes WSNs can be very complicated, so that it may be hard to discover reasons of unexpected low performance. An interesting well-assessed concept coming from the electronic literature that could be advantageously borrowed "*mutatis mutandis*" is boundary scan [26, 15]. In electronics, complex equipment is provided with an added functionality that in practice allows the system to be forced in a testing status. Then, dedicated serial lines (or even the same lines used for exchanging useful data during normal operation) are used to let devices exchange data and commands for testing purposes in a daisy chain topology. Sometimes, it would be very helpful at design time, but also for maintenance purposes to have ready a way to put all devices in a "test mode" status. In this status, ordinary traffic is stopped, while only specialized packets could be forwarded by network nodes, possibly following predefined routes. The main advantage coming from this "testing mode" is that specific properties of the whole system or the behavior of selected system components could be deeply analyzed. This could greatly speed up the process of understanding the behavior of the network itself. Clearly, this mode would require a modification to protocols and device drivers. At the present time, such functionality is just not provided by networking communication systems; the hope is that in the next revision to communication schemes, testing could be considered with more attention by designers of protocols.

4 Conclusions

The above discussion is a selection among a number of other issues, such as "cross-layer design" and "bandwidth" issues, or the role of signal processing, that have not been addressed for the sake of conciseness. Moreover, research and development of products are rather set in motion, so that many practical results can be found. Reference [23] recalls some meaningful examples. Moreover, large projects are both motivating and boosting new interesting results. To quote some, one could consider the active work of the Open Geospatial Consortium (that glues together the efforts of more than 300 companies and public agencies), the Bird Observation program (Great Duck Island, Maine, USA), or ZebraNet (a wide wild-life monitoring WSN deployed in Kenya), glacier or ocean water monitoring, grape soil monitoring, cold chain management, rescue of avalanche victims, vital signal monitoring, and so on. This chapter has tried to let the reader merge concepts coming from two different research fields: measurements and communications. Of course, it cannot cover all the corresponding issues, and anyway the authors hope that after this reading the reader could feel the importance of adding an improved inter-disciplinary perspective to future research work.

References

1. J.N. Al-Karaki and A.E. Kamal. *Routing Techniques in Wireless Sensor Networks: A Survey*. IEEE Wireless Communications, Dec. 2004.
2. H. Spoelder. Virtual instrumentation and virtual environments. *IEEE Instrumentation and Measurement Magazine*, 2:14–19, Sept. 1999.

3. G. Andria, A. Baccigalupi, M. Borsic, P. Carbone, P. Daponte, C. De Capua, A. Ferrero, D. Grimaldi, A. Liccardo, N. Locci, A.M.L. Lanzolla, D. Macii, C. Muscas, L. Peretto, D. Petri, S. Rapuano, M. Riccio, S. Salicone, and F. Stefani. Remote Didactic Laboratory G. Savastano, the Italian experience for e-learning at the technical universities in the field of electrical and electronic measurements: Overview on didactic experiments. *IEEE Transactions on Instrumentation and Measurement Magazine*, 56(4):1135–1147, Aug. 2007.
4. L. Benetazzo, M. Bertocco, S. Cappellazzo, and P. Pegoraro. Platform independent architectures for cooperative remote control of automated test laboratories. In *Proceedings of the IEEE International Conference on AUTOTESTCON/2003*, pp. 52–57, Anaheim, CA, USA, 22–23 Sep. 2003. IEEE.
5. L. Benetazzo, M. Bertocco, and C. Narduzzi. Networking automatic test equipment environments. *IEEE Instrumentation and Measurement Magazine*, 8:16–21, Mar. 2005.
6. M. Bertocco, S. Cappellazzo, A. Carullo, M. Parvis, and A. Vallan. Virtual environment for fast development of distributed measurement applications. *IEEE Transactions on Instrumentation and Measuring*, 52:681–685, 2003.
7. M. Bertocco, F. Ferraris, C. Offelli, and M. Parvis. A client-server architecture for distributed measurement systems. *IEEE Transactions on Instrumentation and Measuring*, 47(5): 1143–1148, 1998.
8. M. Bertocco, G. Gamba, A. Sona, and S. Vitturi. Performance measurements of CSMA/CA-based wireless sensor networks for industrial applications. In *Instrumentation and Measurement Technology Conference Proceedings*, pp. 1–6, 1–3 May 2007. IEEE.
9. A. Carullo and M. Parvis. A GPS-synchronized traveling standard for the calibration of distributed measuring systems. In *IEEE Proceedings of the Instrumentation and Measurement Technology Conference*, 2005.
10. S. Chen. Open design of networked power quality monitoring systems. *IEEE Transactions on Instrumentation and Measuring*, 53:597–601, 2004.
11. I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: A survey. *IEEE Communications Magazine*, 44:115–121, Apr. 2006.
12. A. Ferrero. Measuring electric power quality: Problems and perspectives. *Measurements*, 41:121–129, Feb. 2008.
13. A. Ferrero, S. Salicone, C. Bonora, and M. Parmigiani. ReMLab: A Java-based remote, measurement laboratory. *IEEE Transactions on Instrumentation and Measuring*, 52: 710–715, 2003.
14. G. Gamba and L. Schenato. A distributed consensus protocol for clock synchronization in wireless sensor network. In *Proceedings of the IEEE Conference on Decision and Control (CDC 07)*, New Orleans, USA, Dec. 2007.
15. C.H. Gebotys and M.I. Elmasry. *Optimal VLSI Architectural Synthesis: Area, Performance and Testability*. Kluwer, 1992.
16. A.J. Goldsmith and S.B. Wicker. *Design Challenges for Energy-Constrained Ad Hoc Wireless Networks*. IEEE Wireless Communications, Aug. 2002.
17. IEEE-1451 website at NIST. http://ieee1451.nist.gov
18. Interchangeable Virtual Instrumentation (IVI) Consortium Documentation. http://www.ivifoundation.org
19. International Standard. *Network Time Protocol (version 3) Specification, Implementation and Analysis*, rfc-1305 edition, Mar. 1992. http://www.ietf.org/rfc/rfc1305.txt
20. International Standard IEEE 1588. *Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, iec 61588(e):2004 – ieee std. 1588(e):2002 edition.
21. LabVIEW website at National Instruments. http://www.ni.com/labview/
22. LXI Lan Extension of Instrumentation. http://www.lxistandard.org/
23. K. Romer and F. Mattern. *The Design Space of Wireless Sensor Networks*. IEEE Wireless Communications, Dec. 2004.
24. B.M. Sadler. Fundamentals of energy-constrained sensor network systems. *IEEE A&E Systems Magazine*, 20(8):17–35, Aug. 2005.

25. SCPI Consortium. *Standard Commands for Programmable Instruments*, 1997. http://www.scpi.org
26. F.F. Tsui. *LSI-VLSI Testability Design*. MC-Graw-Hill, 1987.
27. VEE Documentation at Agilent Technologies Website. http://www.agilent.com (note: use the search box with the keyword VEE).
28. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, ieee std 802.15.4-2006 (revision of ieee std 802.15.4-2003) edition, 2006.

Chapter 33
The Distributed Measurement Systems: A New Challenge for the Metrologists

A. Ferrero and R. Ottoboni

Abstract The recent years have witnessed a dramatic improvement in the performances of the analog-to-digital (ADC) and digital-to-analog (DAC) converters, both in terms of resolution, linearity, and conversion speed, and the computing devices dedicated to digital signal processing (DSP). As a consequence, the measuring instruments have evolved toward a virtual instrument (VI) architecture, where the measuring activity is performed by a computer equipped with dedicated conversion peripherals. Since computers can be easily connected to a network of shared resources, the most recent evolution of the measurement systems has involved distributed architectures. These architectures, however, have given rise to a number of new metrological issues, mainly related to their synchronization, that, if not considered, might lead to incorrect results. This chapter is aimed at addressing such problems and their possible solutions.

Keywords Distributed measurement systems · Virtual instruments · Synchronization · Uncertainty

1 Introduction

Probably, one of the most impressive technical revolutions of the last years involved digital signal processing (DSP) in its broadest meaning. In fact, all devices considered in the whole process, from the analog-to-digital (ADC) and digital-to-analog (DAC) converters to the storage and computing devices, have increased their performances to such a level that it is now possible to convert and process signals in the discrete-time domain ensuring a bandwidth as high as a few gigahertzs.

As a direct consequence of this technological revolution, the whole area of the Instrumentation and Measurement has gone through some major modifications that have changed not only the design of the measuring instruments but also the way a measurement process is modeled.

A. Ferrero (✉)
Dipartimento di Elettrotecnica, Politecnico di Milano, Milano, Italy
e-mail: alessandro.ferrero@polimi.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_33,

The first significant change occurred in the 1990s, when the concept of Virtual Instrument (VI) was developed [24, 13]. VIs are actually computers equipped with peripherals dedicated to AD and DA conversion. Therefore, in these instruments, the analog signal processing techniques are confined to the front-end circuitry that realizes the interface to the field, and the measurement procedure is fully developed in the discrete-time domain on the digital signal samples. Therefore, different measurement procedures can be implemented on the same hardware, by changing the algorithm that processes the acquired samples.

The development of VIs has definitely changed the measurement philosophy from the old concept: *one measurement, one instrument*, to the new concept: *one measurement, one algorithm*. The instrument design philosophy has also changed, shifting its focus from the design of highly specialized dedicated architectures to the design of highly efficient, general-purpose architectures for DSP applications. The concept of VI carried, implicitly, the concept of universal instrument.

Since VIs are built around a computer architecture, they have benefit of all recent evolutions of computers. The most important one is, probably, the capability of nowadays computers of interconnecting on a large geographical scale, thus realizing, virtually, a single network of shared, cooperating resources. The Internet is the most evident example of this evolution.

If we apply this concept to VIs, we realize that any computer on which a VI is running can be connected to any other computer on which another VI is running and becomes a part of a larger cooperating system of Distributed Instruments.

The possible applications of this distributed architecture of cooperating instruments are practically countless and provide a viable solution to measurement problems otherwise difficult to deal with. Some of them have been already implemented in different areas, from electric power quality measurements [14, 20], monitoring and diagnostic systems [11, 16, 10, 8, 1, 15], to environmental measurements obtained by means of a dedicated sensor network [18, 23, 22, 6, 9, 21], to advanced education in the measurement field with the implementation of remote laboratories [4, 17, 2, 3].

The dark side of the moon is that these measurement systems are not only providing an effective solution to complex measurement problems, but pose also a number of new metrological issues when we try to answer the basic question of the measurement science: How accurate is the measurement result we get from these new measuring architectures?

This chapter, after shortly discussing the architecture of the modern Distributed Instruments, will focus on the major metrological problem encountered when using these systems.

2 The Distributed Measurement Systems

2.1 The Architecture

A modern measurement system, based on DSP techniques and structured according to a VI architecture, can be represented [13] by the block diagram shown in

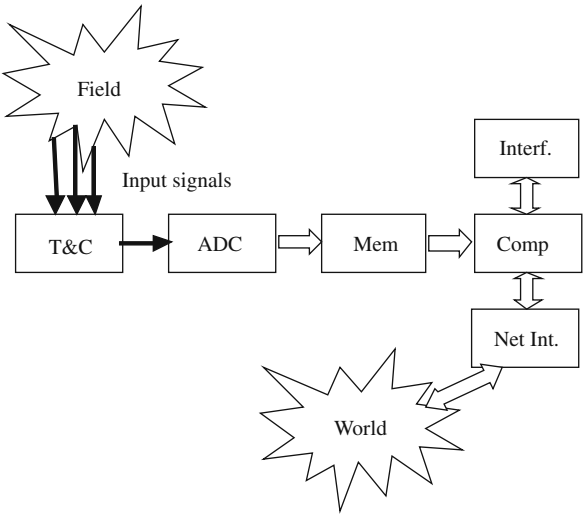


Fig. 33.1 Block diagram of a modern DSP-based instrument. T & C: transducer and signal conditioning; ADC: AD conversion unit; Mem: storage memory; Comp: DSP device; Interf: man–machine interface; Net Int: interface to the network

Fig. 33.1 and realizes a single unit of a Distributed Measurement System (DMS). The following main blocks can be identified.

- Transducer and signal conditioning (T & C) block adapts the input signals to the subsequent AD conversion block and pre-filters the input signals to avoid possible aliasing effects.
- The AD conversion unit (ADC) converts the input, continuous-time, analog signals into discrete-time signals, with the resolution and sampling rate required by the characteristic of the input signals to ensure a correct execution of the measurement algorithm.
- The storage memory (MEM), where the samples of the discrete-time signals are stored, after the AD conversion stage, for further processing.
- The computing unit (COMP) processes the stored samples of the discrete-time signals in order to execute the measurement algorithm and provide the required measurement result.

The four blocks shortly described in the above points define the architecture of a generic DSP-based instrument, within which any measurement process can be seen as one that extracts the desired information from the input signals by means of DSP techniques [13].

The block that specializes this general, DSP architecture into a VI architecture is the man–machine interface (block Interf. in Fig. 33.1) that provides an advanced, user-friendly graphical interface to the user.

In particular, it provides a virtual front panel of the realized instrument, by means of a graphical representation of the conventional controls (buttons, selectors, knobs, etc.) and indicators (digital displays, gauges, scope-like displays, leds, etc.).

It provides also an advanced graphical programming interface that allows the programmer to describe the measurement procedure by simply drawing a procedural diagram that defines data dependencies and operations to be performed. The nodes of the diagram represent high-level operations, such as DSP blocks, peripheral management, data presentation. Each node is represented by a dedicated icon, showing input and output connections for receiving and delivering operand and results. Directed paths to these connections represent the data dependencies among operations [13].

At last, the block that specialized this VI architecture into the node of a more complex DMS is the interface to the network (block Net Int. in Fig. 33.1). This block allows the single VI to be connected, trough a LAN or WAN (such as the Internet), to other VIs, thus realizing a network of sensing units that share their resources and cooperate to implement a measurement procedure.

It is worthwhile stressing these two last concepts of resource sharing and cooperative units, since they are the most innovative ones, when DMS are considered. They also make a clear distinction between the traditional concept of remote instruments, or remote sensing devices, and the more modern concept of DMS.

Indeed, remote instruments could be represented with the block diagram in Fig. 33.1, since they too showed an interface for the connection to a central remote system. However, this interface only allowed the remote instrument to send the measurement results to a central system and receive configuration data, and the communication procedure followed a master–slave architecture in a quite strict way. Moreover, in this architecture, each slave unit is configured as a traditional measurement device, where the whole measurement procedure is performed locally, and the only exploited resources are the local ones. From the operative point of view, these remote instruments can be seen as traditional instruments, with a remote front panel located at the central system site.

It is worth noting that, from a metrology point of view, these systems do not pose, in general, different problems from those posed by a traditional measurement system. In fact, they still are self-confined instruments, and the only de-localized part is the front panel.

On the other hand, each unit of a DMS can share its resources virtually with every other unit belonging to the same DMS. For instance, a unit can process the signal samples acquired by another unit simply by accessing to a shared portion of the other unit's memory or even requesting control of the other unit's ADC. Again, for instance, all units can share the same trigger, and start processing their input signals on the same trigger event; they can subsequently run comparison algorithms, for example, to determine how an event propagated in the area covered by the DMS.

According to the above considerations, the block diagram of a typical DMS is shown in Fig. 33.2. A supervising unit can be present to manage the whole system, store data, and, if necessary, modify the measurement algorithms on the measurement units, adapting them to the possible changes in the monitored system. The modern mobile agent technique can be effectively employed to achieve adaptive measurement systems [1, 15].

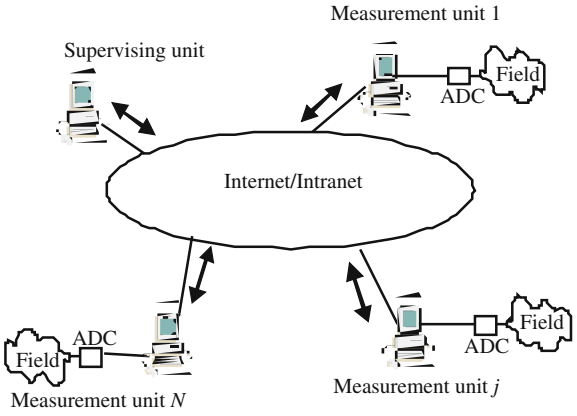


Fig. 33.2 Block diagram of a typical DMS including *N* measurement units and an optional supervising unit

2.2 The Metrology Problem

The main advantage of these architectures is that they can be implemented at a relatively low cost by interconnecting the single units through a public-domain network, such as the Internet. The wireless connections using the GPRS protocol are presently available almost everywhere, and faster connections using the WiFi IEEE 802.11 protocol are covering wider and wider areas, thus making the Internet the most convenient communication support for the Distributed Measurement Systems.

However, a major problem appears when a public-domain network is used: the time taken by each transmitted data packet to reach its destination is unknown and unpredictable, due to the unknown paths on which the routers can address the packet itself, and the data traffic on the network.

The main consequence of the limited knowledge of the transmission delays is that each single measurement unit cannot know the exact time in which the data coming from another unit have been acquired, and therefore cannot locate them on the same time line as its own acquired data.

From the metrology point of view, this can be seen as an unknown, possibly severe jitter contribution to uncertainty. Its estimation is possible, either by experimentally measuring the transmission time over a statistically significant number of events (type A evaluation of uncertainty, according to the recommendation of the Guide to the Expression of Uncertainty in Measurement (GUM) [5]) or by assuming it from a priori knowledge of the average network behavior (type B evaluation of uncertainty, according to the GUM recommendation [5]).

Both approaches, however, do not yield satisfactory results. In fact, the experimental one requires dedicated equipment and procedures to evaluate the transmission time, and requires to repeat the experiment over a significantly long period of time, to be sure that all possible traffic conditions are considered in the test. The confidence level of the obtained results is directly related to the number of tests and

the test duration. On the other hand, the estimation based on the a priori knowledge suffers, in general, of a low confidence and requires an overestimation of the final uncertainty.

The final result is that, in both cases, the estimated uncertainty might be higher than that potentially achievable with the DMS, provided that suitable actions are taken to better define the time line.

The most effective action is that of synchronizing the clocks of the single units of the DMS, thus defining a unique time line for all acquired data. Different solutions are possible, with different impacts on the final uncertainty and system cost. The next section is aimed at briefly discussing the most practical solutions.

3 Clock Synchronization

The main advantage of synchronizing the clocks of all units belonging to the same DMS is that a time stamp can be associated to each acquired data and transmitted along with the data itself. The receiving units can, therefore, successfully relate the incoming data to the self-acquired ones and to those coming by any other unit.

As far as measurement uncertainty is concerned, it is not related to the transmission delay any longer, but to the residual synchronization error. The transmission delay may only affect measurement integrity, in this case they are so high that the measurement procedure cannot be accomplished within the expected time. However, this issue is related more to a system failure, rather than to measurement uncertainty and is not considered in the following.

Different synchronization strategies with different residual errors and different costs are possible. The most used are the one based on the GPS synchronization and the one based on the NTP server synchronization.

3.1 GPS Synchronization

This method exploits the signals coming from the 24-satellite constellation used by the Global Positioning System (GPS) to allow one locating its own position on Earth by means of a simple receiver. The localization method is based on the measurement of the flight time of the time stamp broadcasted by at least three satellites.

Since this requires the transmission of a very accurate time signal, this signal can be used to synchronize the clocks of devices spread on a geographical area. GPS receivers are commercially available that allow to synchronize clocks every second, up to ± 50 ns with respect to the Universal Time Coordinated (UTC).

Since the short-term stability of the quartz clocks used by the single units of a DMS is generally much better, this means that the different units of a DMS can be all synchronized to the UTC, and the residual deviation can be supposed uniformly distributed in the ± 50 ns interval.

The GPS synchronization is presently the most accurate method for ensuring a uniform time reference to the data acquired by the different units of a DMS distributed on a very wide area. The DMS developed in [7] for the measurement of electric power quality indices is a good example of its effectiveness.

3.2 NTP Synchronization

The Network Time Protocol (NTP) has been originally developed in 1991 [19] to synchronize the clocks of computers connected to the same packet-switched, variable-latency network.

The advantage, over the GPS-synchronization method, is that it does not require any external unit, but only a dedicated routine for the connection to a dedicated NTP server.

The disadvantage is that the synchronization accuracy is much lower: the clocks can be synchronized, under normal operating conditions of the network, within ± 10 ms with respect to the UTC.

Nevertheless, this synchronization technique still defines a unique time line for all acquired data and provides an estimate of the residual synchronization error that can be employed in the estimation of the final measurement uncertainty. The decision on whether the final uncertainty is acceptable or not depends, of course, on the target uncertainty defined for the specific measurement application.

3.3 Measurement Algorithms

While clock synchronization is a strict requirement in the majority of measurement applications involving DMSs, the use of the most accurate GPS-synchronization method is not always strictly needed, depending on the measurement application and the adopted measurement algorithm.

It is quite obvious that a strict, accurate synchronization is needed whenever a signal sample is related to the corresponding sample of another signal, acquired in a different location. This is the case of power quality measurements based on synchronized harmonic phasors [7], where the phase shifts of voltages and currents measured in different nodes of the electric system have to be compared.

Similar problems are encountered when the direction of a transient disturbance traveling on the electric system has to be detected, in order to identify its origin. In this case, the exact time at which the disturbance appears in the different nodes must be evaluated and compared with that evaluated in the contiguous nodes to assess the direction. According to the velocity with which an electromagnetic wave propagates along an electric line and the typical distance between two nodes, the measured time differences, can be in the order of some microseconds, thus strictly requiring the synchronization accuracy provided by the GPS synchronization.

On the other hand, slower phenomena do not require to process only a few samples, simultaneously acquired. In many applications, average quantities, such as rms values, harmonic components, or long-period averages, are required to monitor the evolution of slowly variable phenomena.

In these cases, the synchronization errors do not affect the final measurement accuracy until they are negligible with respect to the averaging period. For example, if each unit of the DMS provides measurement results averaged over an interval of a few minutes (as is the case, for instance, of the averaging intervals considered by the electric energy meters for billing purposes), and the measured results coming from the different units must be time related to each other, the synchronization accuracy provided by the NTP-synchronization technique is quite adequate [12].

It can be concluded that the best trade-off between cost and accuracy can be achieved only by carefully considering how the synchronization error is reflected on the adopted measurement algorithm and may affect, consequently, the final measurement accuracy.

4 Conclusions

The modern Distributed Measurement Systems have been shortly described, showing that they can be considered the natural evolution of the remote measurement systems and potentially solve complex measurement problems.

On the other side, a warning has been issued about the risks of disregarding the new metrology issues posed by these new systems. The main one is the great impact that the lack of synchronization between the clocks of the different units of a DMS has on the final measurement result.

If disregarded, clock synchronization may lead to totally incorrect measurement results, thus thwarting the potential advantages of the realized DMS and making it quite useless.

The most common synchronization techniques have been shortly reported and discussed, with respect to the employed measurement algorithms.

It can be concluded that the Distributed Measurement Systems are a very promising tool for monitoring large, complex systems, such as large networks (electric, telecommunication, etc.), industrial systems with severe environmental restrictions, environment. The evaluation of their uncertainty is a critical task, but, if correctly performed, it is helpful in finding the best solution also from the economical point of view.

References

1. M. Albu, A. Ferrero, F. Mihai, and S. Salicone. Remote calibration using mobile, multiagent technology. *IEEE Transactions on Instrumentation and Measurement Magazine*, 54(1): 24–30, 2005.
2. G. Andria, A. Baccigalupi, M. Borsic, P. Carbone, P. Daponte, C. De Capua, A. Ferrero, D. Grimaldi, A. Liccardo, N. Locci, A.M.L. Lanzolla, D. Macii, C. Muscas, L. Peretto,

D. Petri, S. Rapuano, M. Riccio, S. Salicone, and F. Stefani. Remote Didactic Laboratory G. Savastano, the Italian experience for the e-learning at the technical universities in the field of the electrical and electronic measurements: Architecture and optimization of the communication performance based on thin client technology. *IEEE Transactions on Instrumentation and Measurement Magazine*, 56(4):1124–1134, 2007.

3. G. Andria, A. Baccigalupi, M. Borsic, P. Carbone, P. Daponte, C. De Capua, A. Ferrero, D. Grimaldi, A. Liccardo, N. Locci, A.M.L. Lanzolla, D. Macii, C. Muscas, L. Peretto, D. Petri, S. Rapuano, M. Riccio, S. Salicone, and F. Stefani. Remote Didactic Laboratory G. Savastano, the Italian experience for the e-learning at the technical universities in the field of the electrical and electronic measurements: Overview on didactic experiments. *IEEE Transactions on Instrumentation and Measurement Magazine*, 56(4):1135–1147, 2007.
4. L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, C. Offelli, M. Parvis, and V. Piuri. A Web-based distributed virtual educational laboratory. *IEEE Transactions on Instrumentation and Measurement Magazine*, 49(2):1471–1474, 2000.
5. BIPM, IEC, IFCC, ISO, IUPAC, OIML. *Guide to the Expression of Uncertainty in Measurement*, 1993.
6. A. Brandolini, G. D'Antona, M. Faifer, M. Lazzaroni, and R. Ottoboni. Low frequency magnetic flux density measurements based on navigation agents. In *Proceedings ISA/IEEE Sensors for Industry Conference SICON/04*, pp. 86–90, 2004.
7. A. Carta, N. Locci, and C. Muscas. GPS-based system for the measurement of synchronized harmonic phasors. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, pp. 1–5, 2007.
8. A. Carullo, M. Parvis, and A. Vallan. A traveling standard for the calibration of data acquisition boards. *IEEE Transactions on Instrumentation and Measurement Magazine*, 53(2): 557–560, 2004.
9. L. Cristaldi, M. Faifer, F. Grande, and R. Ottoboni. An improbe M2M platform for multisensors agent application. In *Proceedings of the ISA/IEEE Sensors for Industry Conference SICON/05*, pp. 79–83, 2005.
10. L. Cristaldi, A. Ferrero, A. Monti, F. Ponci, W. McKay, and R.A. Dougal. A virtual environment for remote testing of complex systems. *IEEE Transactions on Instrumentation and Measurement Magazine*, 54(1):123–133, 2005.
11. L. Cristaldi, A. Ferrero, A.Monti, and S. Salicone. A versatile monitoring system for AC motor drives. In *Proceedings of the 3rd IEEE International Symposium on Diagnostic for Electrical Machines, Power Electronics and Drives*, pp. 1–6, 2001.
12. L. Cristaldi, A. Ferrero, C. Muscas, S. Salicone, and R. Tinarelli. The impact of Internet transmission on the uncertainty in the Electric Power Quality estimation by means of a Distributed Measurement System. *IEEE Transactions on Instrumentation and Measurement Magazine*, 52(4):1073–1078, 2003.
13. L. Cristaldi, A. Ferrero, and V. Piuri. Programmable instruments, virtual instruments and distributed measurement systems: What is really useful, innovative and technically sound? *IEEE Instrumentation and Measurement Magazine*, 2(3):20–27, 1999.
14. L. Cristaldi, A. Ferrero, and S. Salicone. A distributed system for electric power quality measurement. *IEEE Transactions on Instrumentation and Measurement Magazine*, 51(4): 776–781, 2002.
15. De Capitani di S. Vimercate, A. Ferrero, and M. Lazzaroni. Mobile agent technology for re-mote measurements. *IEEE Transactions on Instrumentation and Measurement Magazine*, 55(5):1559–1565, 2006.
16. R.A. Dougal and A. Monti. The virtual test bed as a tool for rapid system engineering. In *Proceedings of the 1st Annual IEEE Systems Conference*, pp. 1–6, 2007.
17. A. Ferrero, S. Salicone, C. Bonora, and M. Parmigiani. ReMLab: A Java-based remote, didactic measurement laboratory. *IEEE Transactions on Instrumentation and Measurement Magazine*, 52(3):710–715, 2003.
18. A.M. Mielke, S.M. Brennan, M.C. Smith, D.C. Torney, A.B. Maccabe, and M.J.F. Karlin. Independent sensor networks. *IEEE Instrumentation and Measurement Magazine*, 8(2): 33–37, 2005.

19. D.L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.
20. C. Muscas, L. Peretto, S. Sulis, and R. Tinarelli. Investigation on multipoint measurement techniques for PQ monitoring. *IEEE Transactions on Instrumentation and Measurement Magazine*, 55(5):311–315, 2006.
21. F. Ponci, A. Deshmukh, L. Cristaldi, and R. Ottoboni. Interface for multi-agent platform systems. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, pp. 2226–2230, 2005.
22. D. Potter. Smart plug and play sensors. *IEEE Instrumentation and Measurement Magazine*, 5(1):28–30, 2002.
23. Md. A. Rahman, Md. S. Miah, W. Gueaieb, and A.E. Saddik. SENORA: A P2P service-oriented framework for collaborative multirobot sensor networks. *IEEE Sensors Journal*, 7(5):658–666, 2007.
24. H.J.W. Spoelder. Virtual instrumentation and virtual environment. *IEEE Instrumentation and Measurement Magazine*, 2(3):14–19, 1999.

Chapter 34
Recent Progresses of the Remote Didactic Laboratory LA.DI.RE "G. Savastano" Project

P. Daponte, D. Grimaldi, and S. Rapuano

Abstract The Remote Didactic Laboratory *Laboratorio Didattico Remoto*– LA.DI.RE. "G. Savastano" is an e-learning measurement laboratory supported by the Italian Ministry of University. It provides the students of electric and electronic measurement courses at the technical university with access to remote measurement laboratories, delivering different teaching activities related to measurement experiments. The core of the software architecture is the integration of the learning management system (LMS) with the remotely accessible measurement laboratories through web services and thin client paradigm, providing a new approach to remote experiments on measurement instrumentation. The chapter highlights the recent progresses in the last 2 years of the LA.DI.RE. "G. Savastano" project. In particular, the aspects taken into account concern (i) the availability of teaching experiments to cover different topics and aspects of the teaching program of instrumentation and measurement at the technical university, by taking into account the different countries and languages of the students, (ii) the reality of the experimental activity environment, by realizing a system for the real-time visualization of the instruments with the capability of adjusting the video-stream characteristics to the client bandwidth, and (iii) the quality assessment of the educational activities to the students.

Keywords e-Learning · Laboratory experiments · Measurement instrumentation

1 Introduction

Laboratory activity is an open challenge for online teaching applied to scientific domains because much practical training is absolutely essential to assure good knowledge transfer from teacher to students to educate good professionals. Therefore, the remote control of instrumentation for real experiments via the Internet is a topic of interest for many researchers [1, 4, 5, 6, 7, 8, 9, 11, 13, 15].

P. Daponte (✉)
Department of Engineering, University of Sannio, Corso Garibaldi 107, 82100 Benevento, Italy
e-mail: daponte@unisannio.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_34,

In particular, in teaching electric and electronic measurement topics, from academic courses to continuous training in industry, learners should achieve an accurate practical experience by working in real conditions and on real instruments. However, electric and electronic measurement laboratories, both public and private ones, due mainly to their costs, are not widespread, and this complicates the life-long learning of specialized technicians, particularly in the field of process control, quality control, and testing engineering.

The usefulness of e-learning technologies in teaching electric and electronic measurement subjects is still an open question and is one of the topics taken into account in the activities of the Working Group "E-tools for Education in Instrumentation & Measurement" [24].

Based on these fundamental aspects, the Laboratorio Didattico Remoto-LA.DI.RE. "G. Savastano" (Remote Didactic Laboratory), which is dedicated to the memory of Prof. G. Savastano, has been projected and realized.

This measurement laboratory provides the students of electric and electronic measurement courses with access to remote measurement laboratories, delivering different teaching activities related to measurement experiments.

The initial infrastructure is composed of the laboratories at the University of Sannio and at the University of Reggio Calabria "Mediterranea" under the patronage of the National Research Association on Electric and Electronic Measurement (GMEE) [26] and the collaboration of about 20 Italian universities and specialized instrumentation, e-learning, and publishing companies. The novelty aspects of the proposed solution were presented in [19], with respect to the majority of proposals presented in the literature [1, 4, 5, 6, 7, 8, 9, 12, 15, 17, 18, 23, 27]. They consist of the (1) integration of the advantages provided by the off-the-shelf learning management system (LMS), which is compliant with international standards for web services-based training; (2) implementation of the thin client paradigm providing a new approach to remote experiments on measurement instrumentation; and (3) monitoring of delivered teaching services.

The recent progresses in the last 2 years of the LA.DI.RE. "G. Savastano" project are devoted to explore new frontiers for the e-learning in the field of instrumentation and measurement by removing traditional, physical, geographical, and cultural barriers. Therefore, new, advanced, and innovative opportunities offered by the technology and collaboration and initiatives among universities of different countries are used. In particular, a first expansion opportunity is offered by the satellite infrastructure in the framework of the distance learning platform to be realized by the Italian Space Agency [14]. Another opportunity is offered by the currently developing common projects with universities of Croatia, Greece, Slovakia, and Ukraine [10]. As a result, the LA.DI.RE. "G. Savastano" project consists of broadening the laboratory utility for spreading the communication and the knowledge transfer among teachers and students of different countries and culture. By following the previous considerations, the recent practical and actual progresses involve (i) the availability of teaching experiments to cover different topics and aspects of the teaching program of instrumentation and measurement, by taking into account the different countries and languages of the students, (ii) the reality of the environment

created for laboratory activity, and (iii) the quality assessment of the educational activities.

The chapter is organized as follows. Initially, to make it self-containing, a brief overview of the software and hardware architecture of the LA.DI.RE. "G. Savastano" is reported. Successively, some laboratory experiments and hardware interfaces, including specific components delivered to the students to create their own test benches, are shown with the intent to highlight the versatility of the use and the innovative contribution to the teaching program. Moreover, the system is described permitting the real-time visualization of the instruments during the experimental session, with the capability of adjusting the video-stream characteristics to the available bandwidth on the client side. Finally, the technique is presented that is used to evaluate the quality of the system and of the provided services to the students.

2 Software and Hardware Architectures of the LA.DI.RE. "G.Savastano"

In order to allow the students to access the remote and geographically distributed teaching laboratory [26], the web-based multi-tier distributed architecture is implemented, centered on the LMS [2]. The proposed multi-tier architecture is composed of three tiers:

1. The *presentation* tier manages the experiment visualization on the client side. It is based on standard web browsers with no need for specific software components.
2. The *middle* tier manages the system logic on the server side.
3. The *storage* tier performs the data management, which is related, for example, to the management of the user profiles and the distributed management of the data of the available experiments at the different measurement laboratories.

The LMS is executed on a central server of the distributed laboratory, which is called laboratory portal. The LMS interfaces to the users through a web server that is hosted on the same machine.

The laboratory server (LS) is used to interface a real measurement laboratory with the rest of the distributed architecture. It delivers access and control to the laboratory measurement equipment through a service called bridge service. The LS is used for security purposes in order to monitor accesses to the measurement laboratory and to protect it against malicious accesses.

The measurement server (MS) is a server located in a measurement laboratory that enables interaction with one or more instruments, and is connected to a set of different electronic measurement instruments through an interface card.

The used virtual instruments (VIs) are stored in a database of the MS, where the LabVIEW environment is installed. No adjustment is necessary to include the VI in the virtual learning environment. Therefore, the wide number of existing VIs can be reused without requiring added work.

In order to overcome the well-known security weakness of Microsoft-based networks, each laboratory is protected by a Linux-based gateway machine.

3 Overview of the Didactic Experiments

The available experiments are organized to cover all teaching aspects in the field of electrical and electronic measurement.

In the following, only few of the available experiments are summarized with the intent to show the versatility and adaptability of each experiment to the innovative requirements [3].

3.1 Sampling and Windowing of Signals

The experiment allows the student to become familiar with both the signal sampling and windowing. Two different learning levels are available. In the first one, the student experiments with (1) the techniques for signal sampling and (2) the effects in the frequency domain of the different modalities of possible signal windowing techniques. In the second level, by simultaneously setting the characteristics of both the digital-signal generator and the digital oscilloscopes, the student experiments with (1) the techniques for signal sampling and (2) the effects of signal windowing.

3.2 Digital Oscilloscope Characterization

The experimental characterization of the Digital Storage Oscilloscope (DSO) is performed by means of tests according to the standard IEEE 1241-2000. The tests that the student can run are the following:

1. evaluation of the transition thresholds by the occurrence histogram;
2. compensation of the transition thresholds;
3. evaluation of the experimental transfer characteristic of the oscilloscope and comparison with the ideal one;
4. evaluation of the differential nonlinearity and Integral NonLinearity (INL);
5. evaluation of the rms noise in order to evaluate the signal-to-noise and distortion (SINAD) and the effective number of bits;
6. SINAD evaluation in both time and frequency domains.

The DSO available for test is the Tektronix TDS 210; the signal generator is the Agilent 33120A. Both instruments are connected to the PC through a General Purpose Interface Bus (GPIB) IEEE 488 standard bus.

3.3 Automated Power Consumption Measurement of Wireless Modules

In order to perform average power and energy measurements, a complete test bench was implemented. Such a test bench consists of two evaluation boards, including two Bluetooth modules Zeevo ZV3001Z, two digital multimeters (DMMs) Agilent 34401A, one dual-channel DSO Tektronix 3052B, and a virtual instrument (VI) running on a server PC.

Each evaluation board was explicitly developed to enable an easy and accurate measurement of the voltage and the current drawn by the Bluetooth module. For this purpose, each board is provided not only with the Device Under Test (DUT), the corresponding power-supply circuitry, and a printed (F-shaped) 2.4 GHz antenna, but also with a high-accuracy 1 Ω transducer to transform the current waveform drawn by the DUT into a differential voltage and an instrumentation amplifier to boost such a signal by a factor ten. Both evaluation boards are connected with the server PC through two RS232 serial ports, whereas the DMMs and the DSO are controlled, respectively, through a GPIB–USB interface and an Ethernet connection. In particular, the DMMs are used to measure independently and simultaneously the actual supply voltage of the DUT and the current drain, while the DSO enables a remote user to visualize the corresponding waveforms. Both measurement results and waveforms are collected and displayed in the front panel of the VI running on the server. The same VI also controls individually the Bluetooth modules by sending them different sequences of proprietary commands.

All instruments are software-triggered in order to perform current and voltage measurements while remaining in the same state for an adjustable amount of time. Eventually, the collected measurement results are combined according to the definitions of average power and energy, and the final values are displayed along with the corresponding standard uncertainty estimates [21].

From the educational point of view, the learning object associated with the proposed experiments guides a student toward the solution of the proposed measurement problem covering various important and complementary issues such as the following:

1. instrument selection criteria;
2. instrument-setup analysis and performance limitation;
3. analysis of loading and connection effects;
4. hints on printed-circuit-board design;
5. measurement-uncertainty estimation.

3.4 AC Power-Interference Cancellation in Electrocardiogram (ECG) Signals Using Adaptive Filters

The developed laboratory experiment has been purposely designed to show the effects of a sinusoidal interference having a frequency of 50 Hz on an ECG signal.

The objective is to show how a correctly performed numerical processing of a measurement signal can effectively improve the capabilities of estimating the required parameters. The implemented adaptive filter [20] performs a continuous estimation of the interference's amplitude and phase and uses this estimation to correct the input signal, as shown in Fig. 34.1. In particular, the experiment shows how filtering the ECG signal improves measurement of the ECG peak value, which gives information on the opening of cardiac valves. An important part of the laboratory activity is to explain how the uncertainty is to be estimated using only the standard methodology [21]. The used instrumentation consists of a processing system (a PC) equipped with a data-acquisition board (a National Instruments PCI-5911 High-Speed Digitizer) and an arbitrary waveform generator (an Agilent 33120A 15-MHz Arbitrary Waveform Generator). The waveform generator provides the synthesized cardiac signal that is corrupted by the ac power-line interference. The filtering action is performed in real time by the processing system that also implements the user interface.

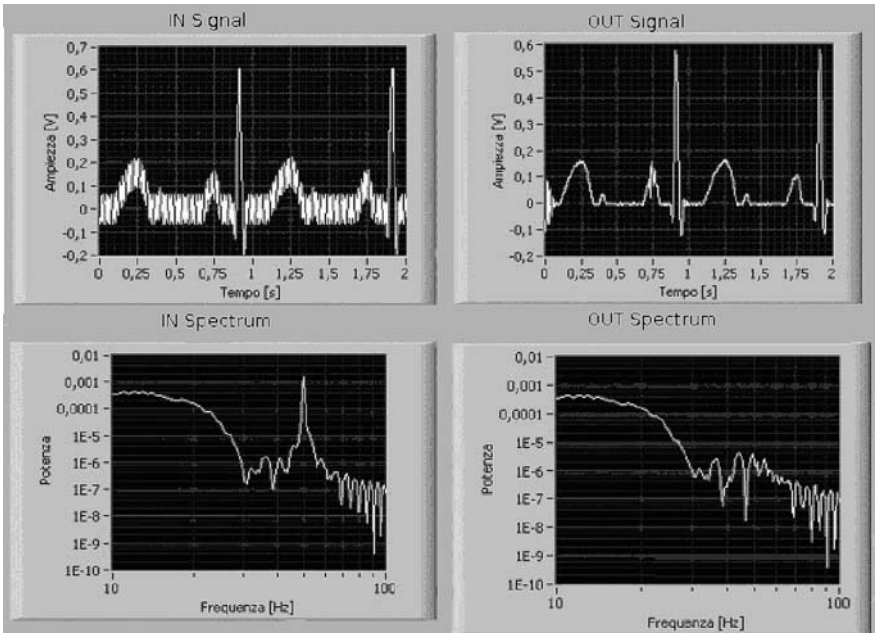


Fig. 34.1 ECG signal amplitude before and after filtering and the power spectrum around 50 Hz before and after filtering [3]

3.5 Basic Electrical Measurements

The readings of an analog (magnetoelectric) and a digital multimeter (DMM) are compared in order to investigate the behavior of the instruments for input signals featuring different frequencies and waveforms. The PC in the laboratory controls,

by means of an IEEE 488 interface, both the function generator Agilent 33120A and a 6 1/2 digits DMM Agilent 34401A. The student has, therefore, the possibility of changing remotely the main settings of both the signal source (waveform, frequency, offset) and the instruments (input quantity, range). Since the magnetoelectric device cannot be obviously controlled by any digital interface, its behavior (i.e., the full wave rectifier and the mechanical movement) is simulated and reproduced by a virtual index, Fig. 34.2. The realism of the experiment is improved by a webcam, which depicts the actual modifications presented in the display of the instruments.

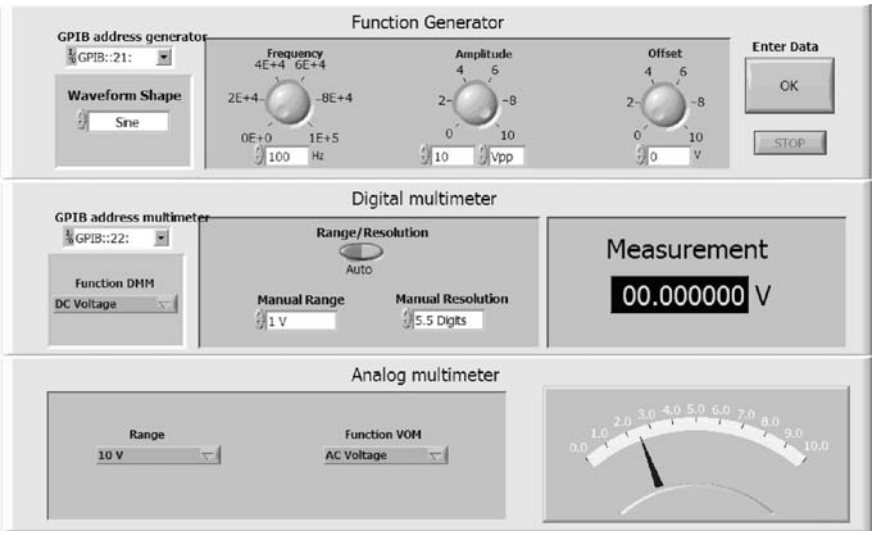


Fig. 34.2 Virtual instrument for analog and digital multimeter comparison [3]

3.6 Magnetic Measurements

This is the first experiment that (1) includes the interactive verification of the knowledge acquired by the students and (2) is executed in four different languages: Italian, English, Croatian, and Romanian. These new features will broaden the laboratory utility for spreading the communication and the knowledge transfer among teachers and students coming from countries where these languages are spoken. This approach follows the current European Union efforts devoted (1) to expanding the newest educational principles to partners and neighboring countries and (2) to create the European education and research area.

During the practical phase, the student will study hysteretic phenomena and how changes in magnetic circuit affect the magnetic induction (B)–magnetic field (H) curve.

The experiment is made with two different magnetic materials, a soft and a hard one. This is an important issue for understanding the wide range of currents to be applied to magnetically saturate different magnetic materials.

3.7 Uncertainty Characterization of Digital Instrumentation

The experiment allows the student to evaluate the overall uncertainty characteristics of a generic digital instrument by analyzing separately the different uncertainty contributions (described in the relevant user manuals) and their influence on measurement accuracy. To this aim, four generators have been implemented to represent the gain, offset, nonlinearity, and quantization errors, respectively, on the basis of the relevant information available directly from the instrument technical data. These input data could be selected by filling specific fields and/or windows in the VI panel shown in Fig. 34.3. For each generator (i.e., for each error-source contribution, like gain error or offset error), the student can select the probability distribution (uniform or Gaussian) and the standard deviation level, according to the accuracy characteristics of the digital instrument under test. The developed program calculates and visualizes the real-time histogram of the generated error signals. The student can also select the number of intervals used to generate the histogram.

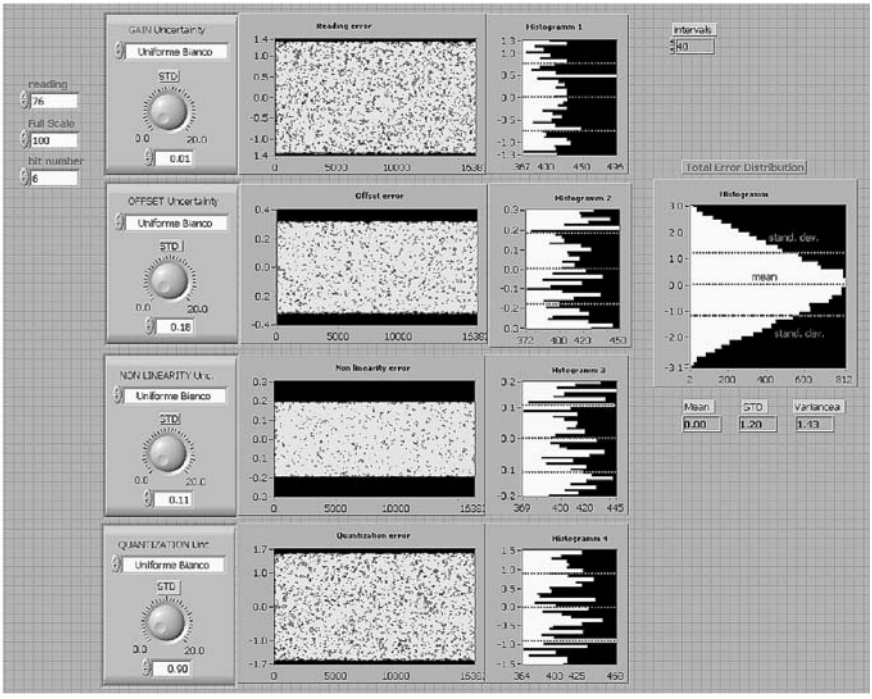


Fig. 34.3 Virtual instrument for uncertainty characterization experiment [3]

Finally, the probability distribution of the total instrument error is calculated by means of the convolution of the four input distributions, under the hypothesis (generally verified) that all the errors are independent and that the noise contribution is negligible with respect to all standard uncertainty components of instrumentation.

Then, the plot of the real-time histogram is generated, and the mean, standard deviation, and variance values are visualized. These values can be compared with those carried out simply by using the uncertainty information from the instrument technical data.

Then, he/she can study the influence of single-error contribution on the global random error. In particular, he can evaluate the statistical distribution (uniform, Gaussian, triangular, etc.) of the resulted global measurement error and all its characteristic parameters, according to the well-known *Guide to expression of Uncertainty in Measurement* (GUM) [21].

4 Measurement Bench for Remote Laboratory Activities

A set of hardware modules, emulating actual measurement instruments and a breadbox for circuit assembly, are provided on demand. The student can act on his measurement bench by assembling the circuit and by connecting the emulated instruments and power supply to the circuit as it actually should be done in the laboratory according to a selected exercise.

The correctness of both the circuit connections and the instrument wirings is dynamically and automatically verified by the system, which is interfaced with the PC through the serial port.

If the student performs all hardware operations correctly, the measurement on the actual system, which is located in a remote laboratory, will be performed and the results sent to the front panel of the student's PC.

The whole system consists of the server unit and of a given number of client units. The server unit is located in the remote laboratory, hosts the web server, and allows the actual measurement benches to be controlled. Each client unit is realized by means of suitable devices connected to the PC, which allow emulation of a real measurement bench.

Only when all the wirings are correctly performed and the circuit is correctly mounted on the breadboard, the PC accesses, via Internet, the server unit. The front panels of the remote instruments will appear on the screen of the student's PC, allowing interaction by setting the proper controls.

5 Real-Time Visualization of the Instruments During the Experimental Session

A method to manage the video-streaming data coming from different codecs has been proposed in [28]. Within that work, it has been proposed to acquire a video from each instrument of the measurement laboratory by means of a video acquisition device connected to the MS. A video bridge component is responsible for sending the instrumentation video received from the MSs to the requiring user. The proposed approach relies on the Real-Time Protocol (RTP) instead of the Remote

Desktop Protocol (RDP) used for remote access to the VIs. In fact, RDP is neither effective nor flexible for video transmission since it does not permit modification of parameters such as codec, video formats, and frame rate and involves a very high bandwidth occupation.

A specific system has been implemented to control the remote visualization of instrumentation integrated with the bridge service already operating in the LSs. The system is developed in Java to guarantee compatibility with the existing bridge service. Moreover, it has been possible to adopt a unified framework for integrating different video codecs, i.e., the Sun Java Media Framework (JMF) [22]. JMF is a framework to process time-based media within Java applets or applications.

It supports RTP and RTP Control Protocol and permits the handling of multiple video codecs and media formats through an easy-to-use plug-in-based architecture. By using JMF, it has been possible to add the remote monitoring of the instrument front panels to the existing user interfaces of the experiments without redesigning the whole LA.DI.RE. "G. Savastano" software architecture. The first architecture presents these software components:

1. *SendLab (Send Laboratory)*: This component is an application running on each MS. The choice of the video acquisition device was discussed in [28].
2. *BridgeUDP*: This application is the system core and performs a bridging on RTP flows coming from MSs, redirecting them to requiring clients. In LA.DI.RE. "G. Savastano," the bridge application runs on LS. The bridge implements the IP mapping, which replaces the IP address of LS with the client IP address (destination IP).
3. *RecLab (Receive Laboratory)*: This application allows the user to receive the RTP flow and to achieve the instrument display on the student's PC through a Java applet.

After the user request for a specific measurement experiment, the LMS verifies the experiment availability and communicates the connection data of the remote client to the LS, as described in [29].

This solution has limited flexibility because it does not allow the student control of the video capture parameters such as frame dimensions and frame rate. Moreover, due to the use of a USB Webcam, the encoding and transmission tasks have to be carried out on the MS, and the video-stream traffic is added to the VI data traffic.

Starting from the first analysis on the video bridge reported in [28], a new set of components has been developed to redirect videos and data coming from the MSs to the students.

The improved software architecture is formed by server and client components developed in Java that enable the delivery of instrument videos. To acquire the videos, an Axis 207 IP camera with an integrated MPEG-4 encoder has been used. In comparison with the first solution, the new one enables the control of the video capture parameters too. In particular, the new architecture presents the following software components:

1. *RTPProxy*: This component is the core of the system. In fact, it represents the LS interface interacting with the LMS to receive the authentication information. This component checks the authentication phase status and manages all user accesses.
2. *AppletClient*: This component is delivered to the student. It acquires connection permission from the LMS, sets up an encrypted connection with the RTPProxy module, allows the student setting up some camera parameters (codec compression factor, frame rate, frame sizes, bit rate) by means of the user interface, and enables MPEG-4 players.
3. *ManagerUsers*: This application controls the access permission, forwards the parameters to set up on the video device, estimates the available bit rate, and enables the forwarding permissions of the visualization request.
4. *TCPProxy*: This component sets the codec and camera parameters as they come from the ManagerUsers component.
5. *PortRedirector*: This application forwards the video stream to the clients and includes the BridgeUDP functions of the first architecture.

Then, the LS starts a direct communication with the client, delivering both the instrumentation access through the VI front panel and the video-stream transmission coming from the IP camera associated with the chosen experiment. In this case, the video stream has an independent part from the MS. No application runs on the MS. Before starting the experiment, the student can choose among different displaying options such as no visualization, photo displaying with different resolutions, visualization of 1 frame/s with different resolutions, visualization of 10 frames/s with different resolutions, and visualization of 25 frames/s with different resolutions. During the experiments, the student can observe the VI front panel and the instrument video in two floating windows, as shown in Fig. 34.4.

6 Testing the Quality of the System

In the following the first tests carried out for evaluating the quality of both the LA.DI.RE. "G. Savastano" system and the provided services will be illustrated.

Hands-on labs involve a physically real investigation process. Two characteristics distinguish hands-on from the remote lab: (i) all the equipment required to operate the laboratory is physically set up and (ii) the students who perform the laboratory experiment are physically present in the lab. Advocates argue that hands-on labs provide the students with real data and "unexpected clashes" – the disparity between theory and practical experiments that is essential for students in order to understand the role of experiments. Moreover, they highlight that this laboratory type assures socialization among the students.

Remote labs, on the contrary, are different from hands-on labs for the distance between the experiment and the experimenter. In fact, in remote labs experimenters obtain data controlling geographically detached equipment. In this way it is possible to extend the capability of a conventional laboratory. Furthermore, its flexibility increases the number of times and places a student can perform experiments and its

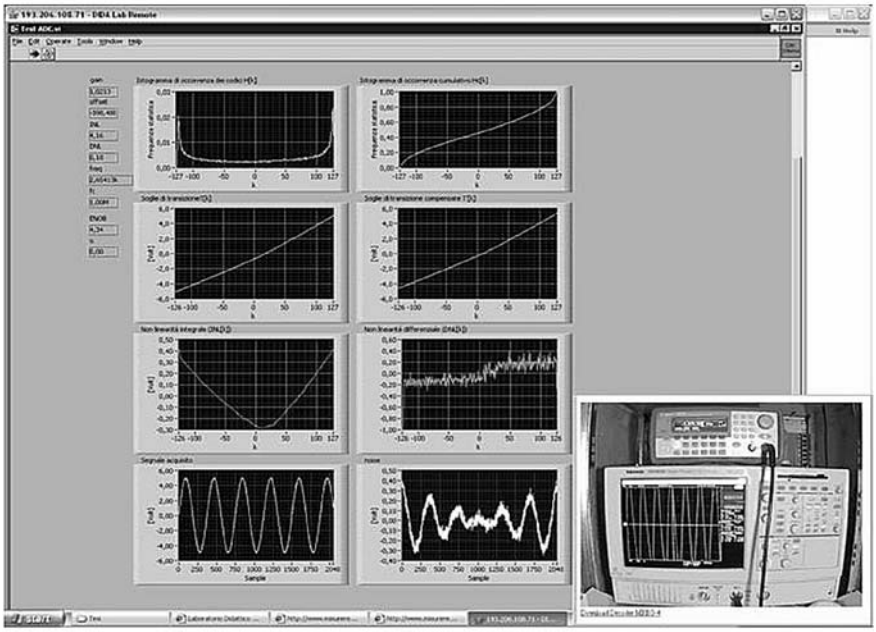


Fig. 34.4 Visualization and control of measurement instrumentation

availability is extended to more students. However, some analyses show that some students consider the simulated and remote labs the same thing, because they do not consider the remote lab as realistic.

The LA.DI.RE. "G. Savastano" system presents the advantages of distributed remote labs but assures also a good socialization for the students and extremely realistic aspect. In fact the system provides some specific "social services" as virtual classroom and chat and especially the real-time visualization of instrumentation front panels, thanks to the flexible use of IP cameras that consents the students to observe the real instrument changes caused by own actions on the user interface [28].

In order to evaluate the degree of effectiveness from the teaching point of view of a remote laboratory, the knowledge acquired by students in using and managing the laboratory instrumentation is verified. Moreover, the students' favor of a remote laboratory in spite of a traditional one is evaluated.

In general the service quality is a concept that aroused considerable interest and debate in the research literature because of the difficulties in both defining it and measuring it, with no overall consensus emerging on either. There are a number of different definitions about service quality. One that is commonly used defines service quality as the difference between user expectations on the service and his/her perceptions about it. The most common and used approach for measuring service quality is the SERVQUAL approach, that is, a widespread direct measuring system in academic and company environment [25]. The SERVQUAL methodology allows

the detection of the gap between the differences of the user expectations and the real perceptions. This is possible through a question set that allows obtaining a specific numeric value representing the user satisfaction about a given service. The users can assign a vote in a given range, called Servqual Score, both to their expectations and to the real perceptions in comparison with a target level for each aspect of the service. Then, this information can be processed by means of a statistical analysis to find the aspects of the service representing its "power areas" and those representing the "improvement areas."

According to the chosen methodology a questionnaire to test in detail the quality of the system and of the provided services has been carried out.

The realized questionnaire presents two important parts: the part concerning the quality of the system and the part concerning the quality of the provided services. The first part includes some important topics as the Internet navigation, the usability, the access procedure (authentication phase), the bandwidth problems, the helpdesk, the experienced delays, the overall system functionalities. The second part, on the contrary, includes topics as the course duration, the graphics, the clearness of the contents (theoretical and practical contents), and the completeness of the contents.

This questionnaire was sent to all LA.DI.RE. "G. Savastano" users in the period October 2005 to September 2006 (Fig. 34.5). This procedure gave the possibility to test the quality of the system and to find the possible improvements of the provided services in conformity with the user satisfaction [16].

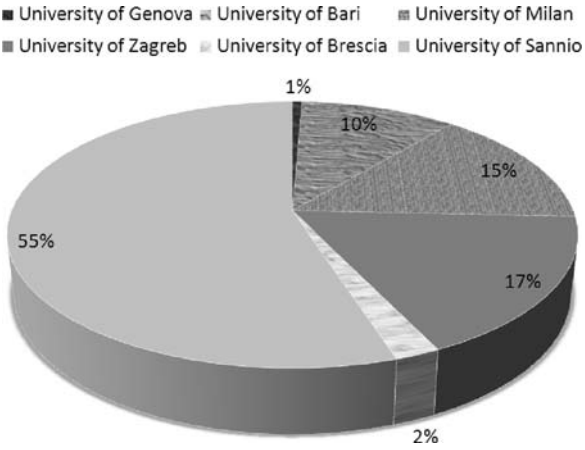


Fig. 34.5 User accesses to the LA.DI.RE. "G. Savastano" in the period October 2005–September 2006

7 Conclusions

The chapter provided an overview on the recent progresses in the last 2 years of the LA.DI.RE. "G. Savastano" project.

In particular, some laboratory experiments and hardware interfaces, including specific components delivered to the students to create their own test benches, have been shown. These teaching experiments concern measurement characterization of instruments and communication systems, measurement devices for remote laboratory, basic electrical measurements, magnetic measurements, electromagnetic interference measurements, and signal processing for measurement applications. The number of the experiments is continuously increasing according to the contributions and the requirements of the users.

Moreover, the system was described permitting the real-time visualization of the instruments during the experimental session. Finally, the technique was presented that is used to evaluate the quality of system and of the provided services to the students.

The ongoing activity is devoted to extend the frontier of the e-learning to the mobile e-learning. Indeed, mobile learning can be considered today the complementary activity to both e-learning and traditional learning. On the basis of this consideration, the ongoing research is addressed to explore the use of the mobile devices in the teaching of electrical and electronic measurement and instrumentation.

Therefore, efforts are devoted to extend the functionalities of the LA.DI.RE. "G. Savastano" to mobile devices. The priority goal is to create a software platform which allows the students to visualize, control, and create experiments by means of mobile devices.

References

1. M. Albu, K. Holbert, G. Heydt, S. Grigorescu, and V. Trusca. Embedding remote experimentation in power engineering education. *IEEE Transactions on Power Systems*, 19(1): 144–151, 2004.
2. G. Andria, A. Baccigalupi, M. Borsic, P. Carbone, P. Daponte, C. De Capua, A. Ferrero, D. Grimaldi, A. Liccardo, N. Locci, A.M.L. Lanzolla, D. Macii, C. Muscas, L. Peretto, D. Petri, S. Rapuano, M. Riccio, S. Salicone, and F. Stefani. Remote Didactic Laboratory G. Savastano, the Italian experience for e-learning at the technical universities in the field of electrical and electronic measurements: Architecture and delivered services. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference IMTC 2006*, pp. 998–1002, Sorrento, Italy, 24–27 Apr. 2006.
3. G. Andria, A. Baccigalupi, M. Borsic, P. Carbone, P. Daponte, C. De Capua, A. Ferrero, D. Grimaldi, A. Liccardo, N. Locci, A.M.L. Lanzolla, D. Macii, C. Muscas, L. Peretto, D. Petri, S. Rapuano, M. Riccio, S. Salicone, and F. Stefani. Remote Didactic Laboratory G. Savastano, the Italian experience for e-learning at the technical universities in the field of electrical and electronic measurements: Overview on didactic experiments. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference IMTC 2006*, pp. 1537–1542, Sorrento, Italy, 24–27 Apr. 2006.
4. P. Arpaia, A. Baccigalupi, F. Cennamo, and P. Daponte. A distributed measurement laboratory on geographic network. In *Proceedings of the IMEKO 8th International Symposium on New Measurement and Calibration Methods of Electrical Quantities and Instruments*, pp. 294–297, Budapest, Hungary, 1996.

5. P. Arpaia, A. Baccigalupi, F. Cennamo, and P. Daponte. A measurement laboratory on geographic network for remote test experiments. *IEEE Transactions on Instrumentation and Measuring*, 49(5):992–997, 2000.
6. A. Bagnasco and A. Scapolla. A grid of remote laboratory for teaching electronics. In *Proceedings of the 2nd International Work on e-Learn and Grid Technologies: A Fundamental Challenge for Europe*, Paris, France, 2003. http://ewic.bcs.org/ conferences/ 2003/2ndlege/index.htm
7. M. Bagnasco, A. Chirico, and M. Scapolla. XML technologies to design didactical distributed measurement laboratories. In *Proceedings of the 19th IEEE IMTC*, vol. 1, pp. 651–655, Anchorage AK, USA, 2002.
8. L. Benetazzo and M. Bertocco. A distributed training laboratory. In *Proceedings of the Eden Annual Conference on Open and Distance Learning in Eurpe and Beyond*, pp. 409–414, Granada, Spain, 2002.
9. L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, C. Offelli, M. Parvis, and V. Piuri. A Web based, distributed virtual educational laboratory. *IEEE Transactions on Instrumentation and Measuring*, 49(2):349–356, 2000.
10. M. Borsic, D. Cmuk, P. Daponte, C. De Capua, D. Grimaldi, T. Kilic, T. Mutapčić, and M. Riccio. Italian-Croatian Remote Laboratory distributed on geographical network. In *Proceedings of the IEEE Softcom'06*, pp. 357–362, Split-Dubrovnik (Croatia), Sep. 29–Oct.1 2006.
11. G. Canfora, P. Daponte, and S. Rapuano. Remotely accessible laboratory for electronic measurement teaching. *Computer Standards and Interfaces*, 26(6):489–499, 2004.
12. M. Cobby, D. Nicol, T.S. Durrani, and W.A. Sandham. Teaching electronic engineering via the World Wide Web. *IEEE Colloquium on Computer Based Learning in Electronic Education*, pp. 7/1–7/11, 1995.
13. P. Daponte, C. De Capua, and A. Liccardo. A technique for remote management of instrumentation based on web services. In *Proceedings of the IMEKO-TC4 13th International Symposium on Measurements for Research and Industry Applications*, pp. 687–692, Athens, Greece, 2004.
14. P. Daponte, A. Graziani, S. Rapuano. *Final report on Progetto preliminare di teleducazione*. Agenzia Spaziale Italiana, 2004. http://progetto-teleducazione.cres.it
15. P. Daponte, D. Grimaldi, and M. Marinov. Real-time measurement and control of an industrial system over a standard network: Implementation of a prototype for educational purposes. *IEEE Transactions on Instrumentation and Measuring*, 51(5):962–969, 2002.
16. P. Daponte, S. Rapuano, M. Riccio, and F. Zoino. Remote didactic laboratory in electronic measurements: quality of system testing. In *Proceedings of the Instrumentation and Measurement Technology Conference, IMTC 2007*, pp. 1–6, Warsaw (Poland), 1–3 May 2007.
17. Educators Corner. http://www.educatorscorner.com/experiments/index.shtml
18. D. Grimaldi, S. Rapuano, and T. Laopoulos. Aspects of traditional versus Virtual Laboratory for education in instrumentation and measurement. In *Proceedings of IEEE IMTC 2005, Instrumentation and Measurement Technology Conference*, pp. 1233–1238, Ottawa, Canada, 17–19 May 2005.
19. D. Grimaldi, S. Rapuano, and T. Laopoulos. Exploring the capability of web-based measurement systems for distance learning. In F. Davoli, S. Palazzo, and S. Zappatore, editors, *Distributed Cooperative Laboratories: Networking, Instrumentation and Measurements*, pp. 373–393, 2006. ISBN 0-387-29811-8.
20. M.L. Halstrom and W.J. Tompkins. Digital filters for real-time ECG signal processing using microprocessors. *IEEE Transactions on Biomedical Engineering*, BME-32:708–713, 1987.
21. ISO ENV 13005:1999.
22. Java Media Framework. http://java.sun.com/products/java-media/jmf/
23. Laboratory of Signal Processing and Measurement Information. Engineering Faculty of University of Sannio. http://lesim1.ing.unisannio.it /english/labrem_eng.htm
24. LEIM-WG. http://www.deis.unical.it/deis1.0/portale/ricerca/leim/wg/
25. Manage. http://www.12manage.com

26. Misureremote. http://www.misureremote.unisannio.it
27. G.C. Orsak and D.M. Etter. Connecting the engineer to the 21st century through virtual teaching. *IEEE Transactions on Education*, 39(2):165–172, 1996.
28. N. Ranaldo, S. Rapuano, M. Riccio, and F. Zoino. Remote control and video capturing of electronic instrumentation for distance learning. *IEEE Transactions on Instrumentation and Measuring*, 56(4):1419–1428, Aug. 2007.
29. S. Rapuano and F. Zoino. A learning management system including laboratory experiments on measurement instrumentation. *IEEE Transactions on Instrumentation and Measuring*, 55(5):1757–1766, Oct. 2006.

Chapter 35
A Software Architecture for the m-Learning in Instrumentation and Measurement

P. Daponte, D. Grimaldi, and S. Rapuano

Abstract Mobile-learning establishes a new frontier for research of e-learning in electrical and electronic measurement and instrumentation. Indeed, the reduced functionalities of the mobile devices must be integrated with the typologies of the services delivered to the student. By referring to the Remote Didactic Laboratory Laboratorio Didattico Remoto – LA.DI.RE. "G. Savastano" the delivered services taken into account are experiment visualization, experiment control, and experiment creation. In order to enable the user of the mobile device to view, control, and create the experiment application, several software architectures have been considered. Finally, the description of the proposed solution is given in the framework of the LA.DI.RE. "G. Savastano".

Keywords m-Learning · e-Learning · Virtual laboratory · Mobile device

1 Introduction

E-learning took learning away from the classroom or campus; mobile-learning (m-learning) is taking learning away anytime and anywhere. While e-learning is an alternative to classroom learning, m-learning is the complementary activity to both e-learning and traditional learning [5]. m-learning expects that the user would like to interact with educational resources while away from their normal place of learning – classroom or computer. It is gaining appeal among younger generations who have grown up using portable video game devices and wireless technology. In this sense, m-learning appeals not only to those who need learning portable but to those who have grown up with a cognitive disposition toward using mobile devices (MDs) whether or not they have the need for true portability in their learning.

P. Daponte (✉)
Department of Engineering, University of Sannio, Corso Garibaldi 107, 82100 Benevento, Italy
e-mail: daponte@unisannio.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_35,

The m-learning has been recognized by the European Commission, which founded the m-learning project [1].

The m-learning has been around for longer than e-learning, with the paperback book and other portable resources, but technology is what shapes its usage today. Technology now allows to carry vast resources in the pockets and to access them wherever one finds it convenient by means of a series of portable devices such as hand-held PCs and smartphones. While the opportunities provided from the use of portable devices for m-learning are new, the challenges are quite old as a consequence of their small screens, limited processing power, and input capabilities. These challenges mean that adapting existing e-learning services and didactic contents to m-learning is not a trivial task.

The aim of this chapter is not to introduce m-learning as a novel and popular way of learning that can substitute the older types of learning, but to show how it can become an important component of blended learning in the field of electric and electronic measurement. With today's prevalence of high technology in many Countries, *blended learning* often refers specifically to the combined adoption of e-learning or m-learning and classical education methods (classroom courses, etc.). In the chapter some aspects of the m-learning are addressed concerning the education in the electrical and electronic measurement and instrumentation. This engineering field involves certain peculiarities which will be solved in practical way, so that optimized blended learning process in the electrical and electronic measurement and instrumentation curriculum becomes feasible.

Indeed, practical experience is a crucial step in effective transfer of knowledge to students in electrical engineering education. But the need for experimentation is often overcome by practical problems in creating, maintaining, and using an efficient laboratory in undergraduate curriculum [2].

It is highlighted in [6] that the Remote Didactic Laboratory Laboratorio Didattico Remoto – *LA.DI.RE. "G. Savastano"* delivers multiple e-learning resources to the learners by means of a complete learning management system (LMS) and enables learners to execute geographically distributed experiments on real equipment based on a thin client approach. Following the newest approaches to the e-learning optimization and extension, the *LA.DI.RE. "G. Savastano"* includes testing of the effectiveness, implementation of the newest standards, realization of feedback aided experiments in multiple languages.

In order to extend the functionalities of the *LA.DI.RE. "G. Savastano"* to MDs, the primary target is to create the software platform which will allow students to display, control, and create experiments from such devices. This last aspect makes the major difference from the common m-learning systems.

In the following, the main characteristics of the LA.DI.RE. delivered services are discussed. These aspects are fundamental to design the proper architecture for the m-learning. Then, the technical characteristics of the MDs used as references for the research are summarized. Finally, the software architecture to deliver m-learning courses as a new feature of the *LA.DI.RE. "G. Savastano"* is described.

2 Services Delivered by Means of LA.DI.RE. "G. Savastano"

The Remote Didactic Laboratory *LA.DI.RE. "G. Savastano"* is a virtual learning environment devoted to the teaching of electric and electronic measurement that integrates an off-the-shelf LMS and a geographically distributed laboratory, accessible from the web by using a simple browser. The distributed laboratory is accessed through the LMS executed on a central server that delivers such functionality to users by means of a thin client-based software architecture [4, 6], and virtual instruments (VIs), controlling the instrumentation.

This platform delivers the typical functionalities of a common LMS, including user authentication, management, and tracking of learning process at user level. Moreover, it provides several innovative functionalities encapsulating in specific Learning Objects (LOs) the remote control of measurement instrumentation. This objective has been achieved by developing an additional module for the LMS Inform@ from Didagroup. The module ensures the integration of VIs written in LabVIEW or in other programming languages in the LMS as LOs, thus enabling remote users to transparently get the control of a real measurement instrument and to display the measurement results within the normal learning activities. Currently, all the VIs have been developed in LabVIEW.

The remote laboratory is distributed on geographical scale since the measurement instruments are physically located in laboratories belonging to different universities. At present, there are four laboratories involved. Two of them have been realized and operate at the University of Sannio in Benevento and at the "Mediterranea" University of Reggio Calabria in Italy. Two more laboratories are going to be realized at the University of Zagreb (Croatia) and at the Technical University of Kosice (Slovak Republic). The web address is `www.misureremote.unisannio.it`.

The access to the measurement instruments is handled by a scheduling system which, transparently through specific scheduling policies, connects the user to a specific physical laboratory in which the required measurement instruments are available.

Different user profiles are managed by the system: student, teacher, and administrator [6].

The services delivered by the remote measurement laboratory module to the student are mainly the following:

- *Synchronous virtual laboratory* – this service allows the student to follow online laboratory activity held by the teacher. The student can see on his/her display the desktop of the server used by the teacher to control the measurement instruments involved in the experiment. The experiment is carried out by operating on the front panel of the LabVIEW VI controlling all the involved instrumentation. Of course, the students should be logged in the system during the scheduled lab session. The data stream from the physical laboratory to the students can be sent in multicast mode. No automated scheduling policy is foreseen for such kind of activity.

- *Experiment visualization* – this service allows the student to observe the automatic execution of the experiment to take practice with the experiment procedure. This kind of service can be delivered to the students at each time of the day and all the times they need it without supervision.
- *Experiment control* – this service allows the student to perform an experiment controlling remotely one or more instruments and, in some cases, observing them by means of a camera. The student can choose a specific experiment in a set of predefined ones and he/she can run it only if the required measurement instruments are currently available.
- *Experiment creation* – this service allows the student to create remotely an experiment by interacting directly with specialized software executed on the servers used to control the measurement instruments. This feature enables the adoption of project-based learning (PBL) as didactic model. Under the supervision of the teacher, the students can develop a specific project producing, in an individual or collaborative manner, the VI to control a set of real instruments.

The services delivered to the teacher are related to the remote handling of the available experiments (remote creation, modification, and removal of experiments, etc.).

Finally, the administrator is responsible for the correct working of the overall distributed system and of handling the user profiles.

Only the last three asynchronous services are taken into account for the m-learning: experiment visualization, experiment control, and experiment creation.

3 Technical Characteristics of Mobile Devices

The limitations of the technical characteristics of the MDs limiting the possible solutions and giving specific directions to the research and development of the software architecture for m-learning can be grouped in hardware, software, and bandwidth ones.

3.1 Hardware Limitations

The biggest hardware limits of MDs are the small displays and navigation control problems for MDs without touch screen. The reduced display size without any standard format requires an adaptive run-time environment and specific Graphical User Interfaces (GUIs). On the other hand, the inner limits to navigation control due to the hardware navigation tools of MDs require new approaches to the experiment control and creation. Limited processor power is also a strong drawback because applications must be very "light," consuming memory as little as possible in order to reduce the response time of the applications running on the device.

The wide variety of the MDs can be split into two categories. The first one is represented by so-called smart phones, while the other one is represented by the Personal Digital Assistant (PDA). A relevant difference between the two groups is that a lot of smart phones do not have touch screens or any pointing devices, while the PDA usually has, which is an important issue. Moreover, PDAs have bigger

memory resources and processor power. Such a difference has been faced in Java Mobile Edition (J2ME) by considering different configuration specifications [7].

Taking in account such differences, two main limitations of MDs have been considered during the research work, having the highest impact on the services provided by LA.DI.RE. "G. Savastano": the software and bandwidth limitations.

3.2 Software Limitations

In order to access the LA.DI.RE. "G. Savastano" system, the basic software needed on the client side is the operating system (OS), a Java-compatible browser, and a Java virtual machine (JVM) installed.

In traditional and laptop PCs, the basic software configuration is almost the same even if the hardware is different, including a limited number of combinations of three OSs (Windows, MAC, and Linux), three browsers (Internet Explorer, Mozilla Firefox, and Opera) and one JVM. Usually, a new version of a software package is compatible with the preceding ones.

MDs, instead, come with a wide variety of installed software from different producers and with different versions that are sources of huge interoperability problems for developers. Starting from the OS, involving strict requirements for creating standalone or distributed applications, the supported programming languages differ greatly. Moreover, even the same OS (like Windows CE), throughout its versions, supports only some versions of the same programming language. The Java™Platform, Micro Edition (Java ME), which is one of the most ubiquitous application development platform for MDs, varies greatly in versions and in terms of built-in packages from device to device. The same problem exists concerning the browser capabilities of supporting JavaScript and other web-based languages. The ubiquitous browser Internet Explorer Mobile (IE Mobile) or Opera Mobile, for example, supports only a limited set of functionalities of JavaScript, Document Object Model, and XML.

3.3 Bandwidth Limitations

A wireless LAN is not always available to the potential user, and for sure it does not fulfill "anytime and anywhere" requirements, so the system should be able to work over UMTS, EDGE, or GPRS connections. The latter has high latency and restricted bandwidth, which is a big drawback for applications or services requiring to be always connected, like those controlling the instrumentation in real time.

4 Software Architecture for the m-Learning

In the following the possibility to use the MDs for the m-learning in electrical and electronic measurement and instrumentation is explored by referring to the delivered services of the *LA.DI.RE. "G. Savastano"*.

In particular, the functionalities of the MDs must integrate with the three asynchronous services delivered to the student by the remote measurement laboratory: experiment visualization, experiment control, and experiment creation.

In order to enable the user to view and control an experiment VI, several methods have been considered and tested. But, only a limited number of them could be used with the existing system. On the contrary, only one method is able to permit the user to create a new experiment. These methods are described in the following.

4.1 Experiment Visualization

Remote experiment visualizations are normally a static GUI where only data presentation is dynamic (graphs, diagrams, numerical results, etc.). Thus, every optimized solution will separate data from its presentation. In other words, it is necessary to reduce the traffic as much as possible to enable effective and meaningful data visualization of the experiment. This specification limits the possible solutions because the separation of data processing and presentation usually requires proprietary protocols, and requires that the experiment creator be able to write the experiment with a certain level of understanding of the communication protocols. One of the main consequences is that a huge number of already developed experiments should be redesigned and realized from scratch or cannot be integrated in the system.

LabVIEW is one of today's most recognized and used programming tools for the development of the measurement software. Some of the best solutions are proprietary solutions of the NITM. The first one consists in realizing two VIs, one on the client side and another on the server side communicating via the TCP/IP protocol of the proprietary *Datasocket* technology. The second one relies on LabVIEW web server that gives the possibility of viewing the front panel of a VI running on the server machine, while for accessing and controlling it the client should have LabVIEW run-time environment installed.

Several solutions have also been proposed implementing a specific client-server couple using object oriented programming languages, instead of using LabVIEW. National Instruments provides a Lab VIEW web browser client and server-side software. These modules, however, are better adapted for factory automation behind a firewall than the open Internet. This solution, like similar specific client-server structures, has the following advantages: (i) high bandwidth efficiency, (ii) light clients that do not require specific plug-ins, and (iii) high portability, when open technologies are used on the client side. Its main disadvantage is the limited reusability of already developed experiments and the necessity of developing new versions of existing VIs.

By means of the approach proposed in [6], those problems were successfully solved. The thin client paradigm splits the presentation layer between client and server. The presentation logic runs on the server, while the thin client has the only task of showing the GUI that often reproduces a window of the application running on the server. This approach gives the possibility of providing remote access to any application, written in whichever language, running locally on the server and

requiring only an Internet browser on the client side. Moreover, being born for the MDs, the thin client paradigm is inherently suited for m-learning applications.

Many implementations of the thin client paradigm have been proposed by researchers and deployed commercially. The thin client platforms that have been evaluated are: Remote Desktop Protocol (RDP), Independent Computing Architecture (ICA), X-Window, and Virtual Network Computing (VNC). Those platforms were evaluated in [4]. The RDP protocol was chosen, for its low cost with respect to ICA, and higher efficiency with respect to X-Window and VNC. RDP is a proprietary protocol of Microsoft, and allows direct connection to the Windows server. It supports different mechanisms to reduce the amount of transmitted data (compression, cashing of bitmaps in RAM), which is very important for applications that use a large amount of bitmaps. A thin-client approach that was used for visualization of remote experiments in PC browsers has been used as a basis for delivering similar content to MDs.

In order to realize a novel client–server architecture making possible to access the experiment visualization service the development work started from the same client formerly designed and realized for the LA.DI.RE. "G. Savastano" system and using web services [8], Asynchronous JavaScript, and XML. The web-based application has been developed by using AJAX technology. Ajax is a browser technology that involves the use of existing web standards and technologies (XML/XHTML, DOM, CSS, JavaScript, XHR – XMLHttpRequest) to create more responsive web applications that reduce bandwidth usage by avoiding full page refreshes and providing a more "desktop application-like" user experience. Applications created by Ajax are generally classed in the category of Rich Internet Applications (RIA).

The currently employed system is based on a modified ProperJavaRDP client [4] which is used to connect to a server delivering the experiment and running Windows terminal services. The client grabs the image of a VI front panel and, by means of XMLHTTPRequest, handles it dynamically to a "lightweight" web page that provides the front panel presentation on the MD screen. Since the image dimensions are very small on MDs, some cashing algorithms would be just consuming processor power of the sever disabling it to be highly *scalable*, while handheld devices would lose their limited resources on cashing and compression, with small benefit in bandwidth usage. Therefore, no caching has been adopted in the application.

4.2 Experiment Control

The control of the experiment is based on a Javascript, which "catches" mouse coordinates and key pressed from the device keyboard and transfers them to an intermediate web server, realized for such task, that passes the data through the ProperJavaRDP client to the RDP server. Web pages are also dynamically loading with JavaScript reducing the start-up time of the experiments. Adaptive resolution of the dynamic web page which will enable optimal image size for various MDs is still under construction.

Possible alternatives have been tested in order to achieve the best accessibility and usability of the experiments. Java mobile Edition has been abandoned because the VI that comes with JVM is generally a kilobyte size virtual machine [7]. Therefore, it is unable to handle demanding issues like image handling, presentation, and socket communication with the server. Sun abandoned the development of usable JVM for PDA, so currently it exists only as a commercial product form IBM (Workplace Client Technology, Micro Edition). Another possible solution was to develop application in C#. Lack of this solution was that it would work only on Microsoft mobile platforms.

4.3 Experiment Creation

The experiment creation by means of a MD is based on a new design approach of the innovative VI proposed in [3]. This approach permits to overcome the difficulties arising from the limited amount of memory in a MD, while enabling the user in the dynamic configuration of the measurement procedure by operating on the MD. In this manner the user can customize the measurement procedure by operating on the MD, once the innovative VI has been downloaded on demand. This approach is based on the organization of the innovative VI in both autonomous and self containing entities on the basis of the characteristics of the measurement instrument (MI). The communication among the modules is permitted by using a virtual bus implemented into the MD.

5 Design of the New Experiment Creation Module

The traditional approach to design the VI is based on the implementation of all the functionalities of the MI. All the functionalities are always available by means of the GUI, and the user employs only those required by the measurement procedure. The result of this approach is the complete mapping of the MI into the VI (Fig. 35.1a). Alternatively, only the functionalities of the MI required by the measurement procedure can be implemented by selecting the corresponding cluster of commands, as shown in Fig. 35.1b. In this case the VI is created and customized on the basis of the complete measurement procedure. The selected functionalities are available always by means of the customized GUI of the measurement procedure. The advantage consists in the reduction of the memory size. The inconvenient consists in re-building the VI if new functionalities of the MI are needed. In the case numerous MIs are employed, the VI implementing the measurement procedure uses the required functionalities of each MI by selecting the corresponding cluster of commands. Also in this case the selected functionalities of the MIs are always available by means of the customized GUI of the measurement procedure.

The implementation of the measurement procedure in only one VI can be advantageous during the design. Indeed, this approach speeds up the realization of the VI. On the contrary, the VI must be changed if new functionalities of the MI are needed.

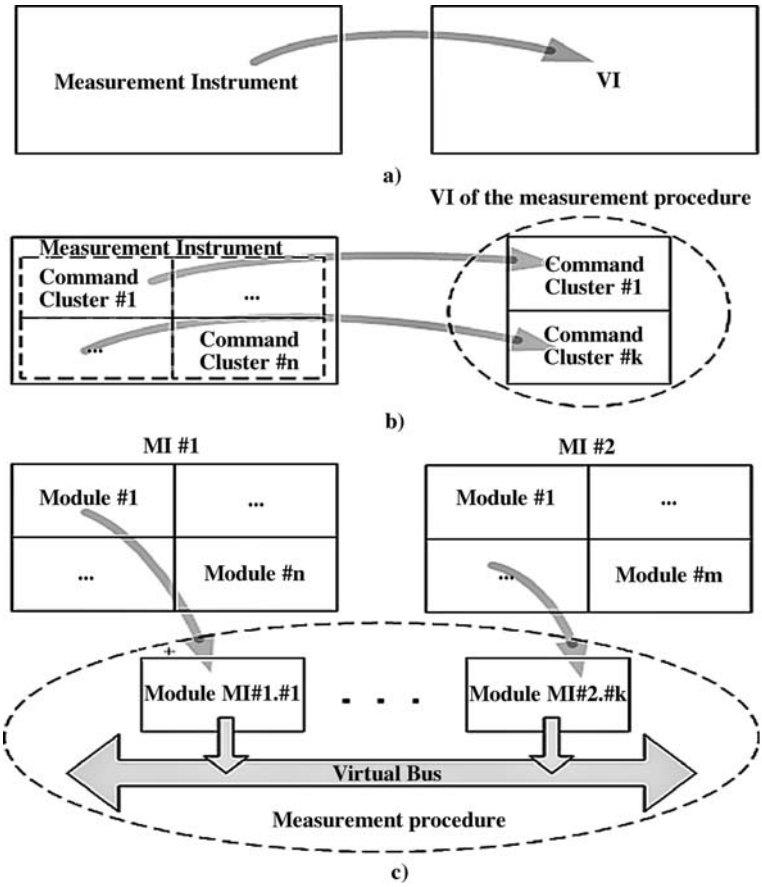


Fig. 35.1 Traditional and innovative criteria to build the VI of the measurement procedure

The new design approach proposed in [3] is based on splitting up the traditional VI of the MI in several self-contained modules. The modules are able to exchange data by using the virtual bus in order to execute the measurement procedure. The criteria to be taken into account and the steps to be followed to define and implement each module are the following:

Step 1. To analyze the commands of the MI under consideration and define the different logical functionalities;
Step 2. To select all the commands to be used to implement each functionality in autonomous and self containing module;
Step 3. To organize each module to receive the commands from the GUI;
Step 4. The data should be exchanged by the virtual bus, only;
Step 5. Each module must execute the procedure without exchanging commands or data with other modules;

Step 6. Each module can be written in a different language and communicate with the other modules by using specific files;

Step 7. Each module produces a specific file in a custom XML format in order to save the info and to be easily read by different modules written in different languages.

In order to highlight the advantages of this approach, Fig. 35.2 shows the organization of three measurement procedures created on a PDA by splitting the VI controlling a Digital Storage Oscilloscope (DSO) in self-containing modules, according to the steps of the design approach. In this case the user downloads on demand by the server only the modules employed in the measurement procedure. Therefore, the occupancy of the PDA memory can be reduced to the minimum. Each module can be written in a different language and communicates with another one by using the virtual bus, as depicted in Fig. 35.2. The independence among the modules permits the user to dynamically configure the measurement procedure on demand.

The configuration of the measurement procedure is performed on the PDA by organizing the ordered list of modules by using the specific browser.

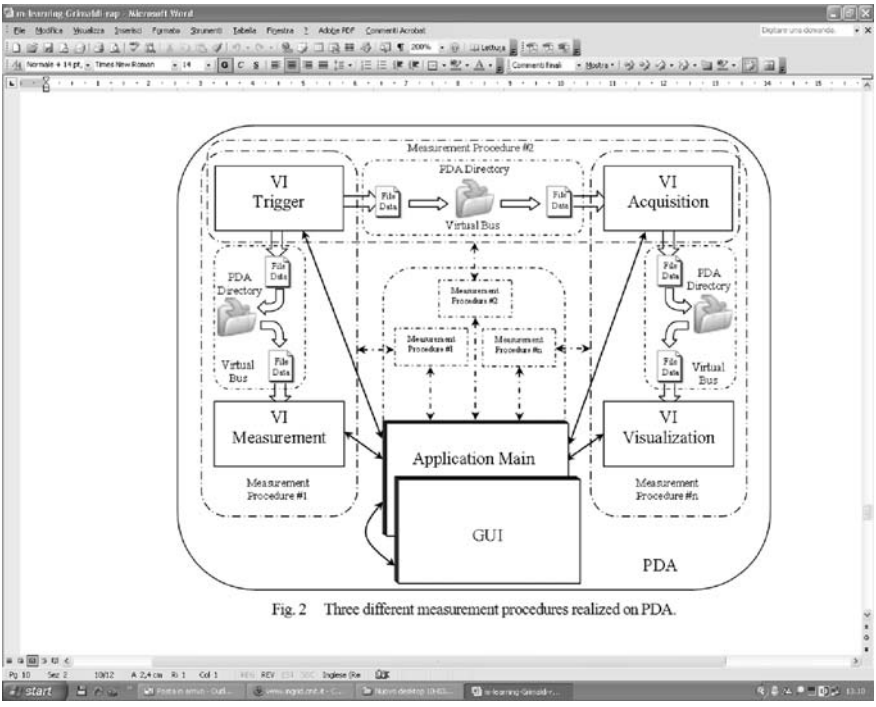


Fig. 2 Three different measurement procedures realized on PDA.

Fig. 35.2 Three different measurement procedures realized on PDA

6 An Example of Experiment Creation

For security reasons, in order to grant access to the system, the user has to provide a login and password (Fig. 35.3a). After the logon, the system will show a list of the applications already available to be executed on the PDA (Fig. 35.3b). If the required application is not in this list, the user can search and, eventually, download it from a measurement server.

Fig. 35.3 Experiment creation on PDA. From left to right: (**a**) login, (**b**) available applications, (**c**) modules of each application, and (**d**) list of available modules

By selecting and opening the application in the list, the system will show all the modules composing the application (Fig. 35.3c). The user can run each module individually or can set up a new measurement procedure selecting the several modules in the list (Fig. 35.3d).

The column "Ready" in the module list of Fig. 35.3c will specify if the module has been already downloaded from the server and is ready to run. At the end of the module list, the measurement application composed by the user is set.

In the case the application is not present on the PDA, the user (i) connects to the system (Fig. 35.4a), (ii) selects the VI category (Fig. 35.4b) or enters a keyword, and (iii) starts the search. Once the necessary application is found, the user can download each module separately or the complete VI (Fig. 35.4c). Of course, the user can download the procedure only if he/she has the correct authorization. As the download is completed, the application will be added to the list of the ready-to-run applications on the PDA (Figs. 35.3b and 35.4a).

Figure 35.5 shows the GUI of each downloaded module. In particular, Fig. 35.5a refers to the trigger module, Fig. 35.5b to the measurement module, and Fig. 35.5c to the acquisition module.

Fig. 35.4 Experiment creation on PDA. From left to right: (**a**) available applications, (**b**) search application, and (**c**) modules of each application

Fig. 35.5 GUI of each downloaded module. From left to right: (**a**) trigger module, (**b**) measurement module, and (**c**) acquisition module

7 Conclusions

Mobile-learning can be considered today the complementary activity to both e-learning and traditional learning. On the basis of this consideration, the research has been addressed to explore the use of the MDs in the teaching of electrical

and electronic measurement and instrumentation. By referring to the delivered services to the students from the Remote Didactic Laboratory Laboratorio Didattico Remoto – *LA.DI.RE. "G. Savastano"* different solutions have been proposed based on several software architectures and on the characteristics of the MDs. In particular, the functionalities of the MDs must integrate with the three asynchronous typologies of the services delivered to the student by the remote measurement laboratory: experiment visualization, experiment control, and experiment creation. In order to enable the user to view and control an experiment VI, the thin client approach was used for visualization of remote experiments on the MDs. In order to enable the user to create the experiment by the MD, the adoption of the VI design approach described in [3] has been proposed. This approach is based on the organization of an innovative VI split in autonomous and self-containing entities on the basis of the characteristics of the measurement instrument.

References

1. J. Attewell and C.S. Smith. Mobile learning and social inclusion: Focusing on learners and learning. *Learning with Mobile Device*, 2004. www.LSDA.org.uk
2. D. Cmuk, M. Borsic, and F. Zoino. Remote versus Classical Laboratory in Electronic Measurements teaching – effectiveness testing. In *Proceedings of the IMEKO XVIII World Congress*, Rio de Janeiro, 2006.
3. D. Grimaldi and F. Lamonaca. Dynamical configuration of measurement procedures on PDA by using measurement application repository server. In *Proceedings of IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Dortmund, Germany, 6–8 Sep. 2007.
4. D. Grimaldi, S. Rapuano, and T. Laopoulos. Exploring the capability of web-based measurement systems for distance learning. In F. Davoli, S. Palazzo, and S. Zappatore, editors, *Distributed Cooperative Laboratories: Networking, Instrumentation and Measurements*, pp. 373–393. Springer, 2006.
5. A. Holzinger, A. Nischelwitzer, and M. Meisenberger. *Lifelong-Learning Support by Mlearning: Example Scenarios*. e-Learn magazine, 2005. http://www.elearnmag.org/subpage.cfm?section=research\&article=6-1
6. S. Rapuano and F. Zoino. A learning management system including laboratory experiments on measurement instrumentation. *IEEE Transactions on Instrumentation and Measurement*, 55(5):1757–1766, Oct. 2006.
7. K. Topley. *J2ME in a Nutshell*. O'Reilly, Mar. 2002.
8. Web Services. http://www.w3.org/2002/ws/

Part VII
Sensor Networks for Measurement

Chapter 36
Performance of Linear Field Reconstruction Techniques with Noise and Correlated Field Spectrum

A. Nordio, G. Alfano, and C.F. Chiasserini

Abstract We consider a wireless sensor network that is deployed over an area of interest for environmental monitoring. As often in the practice, the sensor nodes are randomly distributed on the area and their measurements are noisy. Furthermore, the sensors measure a multidimensional physical field (signal), with correlated spectrum, which can be approximated as bandlimited. A central controller, the sink node, is in charge of reconstructing the field from the sensor measurements, which represent an irregular sampling of the signal. We assume that the sink uses a linear reconstructing technique, and we take as performance metric of the reconstruction quality the mean square error (MSE) of the estimate. We then carry out an asymptotic analysis as the number of sensors and the number of the field harmonics go to infinity, while their ratio is kept constant. In particular, we approximate the MSE on the reconstructed field as a function of the eigenvalues of the matrix representing the sampling system. We validate our approximation against numerical results, for some of the most common spectrum correlation models.

Keywords Signal reconstruction · Irregular sampling · Sensor networks · Random matrix theory

1 Introduction

The deployment of wireless sensor networks for environmental monitoring applications requires to carry on the sampling and the reconstruction of a physical field, whose spectrum can be approximated as bandlimited. In addition, the field spectrum often is correlated with the level of correlation depending on the application.

Other systems in signal processing, such as radars, require the estimation of a signal with correlated spectrum. The peculiarity of sensor networks, however, is that they represent an irregular sampling system, where the sample coordinates are known or can be estimated. Indeed, sensors are typically randomly deployed

A. Nordio (✉)
Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129, Torino, Italy
e-mail: alessandro.nordio@polito.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_36,

over the area of interest. Even in the case where a regular, spatial deployment is desired, in the practice, the sensors cannot be perfectly placed due to obstacles or terrain irregularities [3]. Sensors then deliver their measurements, along with their coordinates, to a central controller, the so-called *sink node*, which is in charge of reconstructing the physical field from the set of received, irregular samples.

From the above description, it should be clear that one of the most fundamental issues in sensor networks is the assessment of the impact of the sample spatial randomness on the reconstructed field quality. As often done, we take as reconstruction quality metric the mean square error (MSE) on the reconstructed signal. Up to now, in the literature this performance metric has been studied considering the impairments due to the random sensor locations or due to quasi-equally spaced sensors [7], or even assuming the information processing to take place in a cluster-based network [6], but always assuming that the sensed field spectrum is uncorrelated. Here, instead, we make the more realistic case where the signal spectrum is correlated and we study the system performance when linear techniques are employed.

We highlight that we do not deal with issues related to information transport and consider that all data are correctly received at the sink node.

Quality analysis of the reconstructed field, as stressed in [8], requires the availability of an expression for the eigenvalue distribution of the matrix representing the reconstruction system. In [9], such a distribution has been proven to tend to the Marčenko–Pastur law [5], as the field dimension d grows to infinity. However, its behavior is far from being understood as nonidealities, like spectrum correlation, have to be taken into account. Based on these observations, in this work we provide an approximation to the MSE on the correlated field, which is reconstructed at the sink node. We do so by exploiting some results on random matrices.

The remainder of the chapter is organized as follows. Section 2 introduces some definitions and the system model under study. Section 3 includes a brief review of the mathematical tools needed in our derivations, while Section 4 presents our analysis as well as some numerical results. Some directions for future work conclude the chapter.

2 Preliminaries

In the following, we give some background on linear reconstruction techniques and summarize some results from previous work on irregular sampling and signal reconstruction.

2.1 Irregular Sampling and Reconstruction of Multidimensional, Band-Limited Signals

We follow the work in [8] and consider a d-dimensional field ($d \geq 1$), $s(\mathbf{x})$, sensed by r nodes. We assume that $s(\mathbf{x})$ can be approximated through a weighted sum of $2M + 1$ harmonics per dimension, i.e., by $(2M + 1)^d$ coefficients of its multidimensional Fourier expansion:

$$s(\mathbf{x}) = \frac{1}{\sqrt{(2M+1)^d}} \sum_{\boldsymbol{\ell}} a_{\boldsymbol{\ell}} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell}^{\mathsf{T}} \mathbf{x}}$$

where $\boldsymbol{\ell} = [\ell_1, \ldots, \ell_d]^{\mathsf{T}}$, $\ell_m = -M, \ldots, M$, $\mathbf{x} = [x_1, \ldots, x_d]^{\mathsf{T}}$, $x_m \in [0,1)$, and $a_{\boldsymbol{\ell}}$ are the coefficients of the expansion. We assume that the r sensors are randomly located in a normalized hypercube $[0,1)^d$. Their positions are denoted by $\mathbf{x}_q \in [0,1)^d$ for $q = 1, \ldots, r$, and the corresponding field values are denoted by $s(\mathbf{x}_q)$.

An important parameter useful for our analysis is β defined as [8]

$$\beta = \frac{(2M+1)^d}{r}$$

i.e., it is the ratio between the number of harmonics used for the field reconstruction and the number of the sensors sampling the field. In the case of regular sampling, perfect reconstruction is achieved for $\beta = [0,1)$, and for $\beta = 1$ we have the limit fixed by the Nyquist theorem for the perfect recovery of the signal.

Following [2], we write the vector $\mathbf{s} = [s(\mathbf{x}_1), \ldots, s(\mathbf{x}_q)]^{\mathsf{T}}$ of the field values at positions $\mathbf{x}_1, \ldots, \mathbf{x}_q$ as a function of the spectrum:

$$\mathbf{s} = \mathbf{G}^{\dagger} \mathbf{a}$$

where the $(2M+1)^d \times r$ matrix $\mathbf{G}$ is defined as

$$\mathbf{G}^{\dagger}_{\nu(\boldsymbol{\ell}),q} = \frac{1}{\sqrt{(2M+1)^d}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\ell}^{\mathsf{T}} \mathbf{x}_q}$$

The function

$$\nu(\boldsymbol{\ell}) = \sum_{m=1}^{d} (2M+1)^{m-1} \ell_m$$

$-\frac{(2M+1)^d - 1}{2} \leq \nu(\boldsymbol{\ell}) \leq +\frac{(2M+1)^d - 1}{2}$ maps the vector $\boldsymbol{\ell}$ onto a scalar index and $\mathbf{a}$ is a vector of size $(2M+1)^d$, whose $\nu(\boldsymbol{\ell})$-th entry is given by $a_{\boldsymbol{\ell}}$. The relation between samples and field spectrum in presence of noise can be written as

$$\mathbf{p} = \mathbf{s} + \mathbf{n} = \mathbf{G}^{\dagger} \mathbf{a} + \mathbf{n} \tag{36.1}$$

where the noise is represented by the r-size, zero-mean random vector $\mathbf{n}$, with covariance matrix $\mathbb{E}[\mathbf{n}\mathbf{n}^{\dagger}] = \sigma_n^2 \mathbf{I}_{2M+1}$. We then define the signal-to-noise ratio on the measure as

$$\mathrm{SNR}_m = \frac{\sigma_a^2}{\sigma_n^2} = \frac{1}{\alpha}$$

2.2 Previous Results on Reconstruction Quality

Let the reconstructed signal be

$$\hat{s}(\mathbf{x}) = \frac{1}{\sqrt{(2M+1)^d}} \sum_{\boldsymbol{\ell}} \hat{a}_{\boldsymbol{\ell}} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell}^{\mathsf{T}} \mathbf{x}}$$

Then reconstruction MSE is defined as [9]

$$\mathrm{MSE} = \mathop{\mathbb{E}}_{\mathbf{a},\mathbf{n},\mathbf{x}} \left[\int_{[0,1)^d} \left| \hat{s}(\mathbf{x}) - s(\mathbf{x}) \right|^2 \mathrm{d}\mathbf{x} \right] = \frac{\mathop{\mathbb{E}}_{\mathbf{a},\mathbf{n},\mathbf{x}} \left[\|\hat{\mathbf{a}} - \mathbf{a}\|^2 \right]}{(2M+1)^d}$$

where the operator $\mathbb{E}[\,\cdot\,]$ averages with respect to the subscripted random vectors and $\hat{\mathbf{a}}$ is an estimate of the Fourier coefficients $\mathbf{a}$. In order to get $\hat{\mathbf{a}}$ we employ a suitable linear filter $\mathbf{B}$ (an $r \times (2M+1)$ matrix) such that the estimate of the field spectrum is given by the linear operation

$$\hat{\mathbf{a}} = \mathbf{B}^{\dagger} \mathbf{p}$$

In particular, we take the linear filter providing the best performance in terms of MSE, that is, the linear minimum MSE (LMMSE) filter. Thus, we have [4]

$$\mathbf{B} = (\mathbf{R} + \alpha \mathbf{C}^{-1})^{-1} \mathbf{G}$$

where the matrix $\mathbf{R} = \mathbf{G}\mathbf{G}^{\dagger}$ is Toeplitz and Hermitian and the covariance matrix of the vector $\mathbf{a}$ is given by

$$\mathbb{E}[\mathbf{a}\mathbf{a}^{\dagger}] = \sigma_a^2 \mathbf{C}$$

Moreover, σ_a^2 is suitably chosen in order to have $\mathsf{Tr}\{\mathbf{C}\} = (2M+1)^d$, where $\mathsf{Tr}\{\cdot\}$ is the matrix trace operator.

Using the above definitions, in the case of the LMMSE we obtain [8]

$$\mathrm{MSE} = \frac{\alpha \sigma_a^2}{(2M+1)^d} \mathsf{Tr} \left\{ \mathbb{E} \left[\left(\mathbf{R} + \alpha \mathbf{C}^{-1} \right)^{-1} \right] \right\} \tag{36.2}$$

The number of harmonics and the number of samples being typically quite large, we are interested in an asymptotic analysis of the reconstruction quality. Thus, we let the number of harmonics per dimension, $2M+1$, and the number of samples r grow to infinity, while the ratio $\beta = (2M+1)^d / r$ is kept constant; in the practice we observed the validity of asymptotic analysis, even for small values of M and r [8].

We therefore consider as performance metric the *asymptotic per sample average* MSE normalized to σ_a^2:

$$\mathrm{MSE}_\infty = \lim_{\substack{M,r\to+\infty \\ \beta}} \frac{\mathrm{MSE}}{\sigma_a^2} \tag{36.3}$$

3 Mathematical Background

Here, we quickly review some material on random matrix theory, which will be used in the remaining of this work.

3.1 Large Random Matrices and Asymptotic Spectrum Characterization

Given an $N \times N$ Hermitian matrix $\mathbf{A}$, the empirical cumulative distribution function of the eigenvalues (also referred to as the empirical spectral distribution (ESD)) of $\mathbf{A}$ is defined as

$$\mathsf{F}_{\mathbf{A}}^N(\lambda) = \frac{1}{N}\sum_{i=1}^{N} 1\{\lambda_i(\mathbf{A}) \le \lambda\}$$

where $\lambda_1(\mathbf{A}), \ldots, \lambda_N(\mathbf{A})$ are the eigenvalues of $\mathbf{A}$ and $1\{\cdot\}$ is the indicator function. If $\mathsf{F}_{\mathbf{A}}^N(\,\cdot\,)$ converges as $N \to \infty$, then the corresponding limit (asymptotic ESD, AESD) is denoted by $\mathsf{F}_{\mathbf{A}}(\,\cdot\,)$. The corresponding asymptotic probability density function is denoted by $f_{\mathbf{A}}(\,\cdot\,)$.

Random matrices have been a part of advanced multivariate statistical analysis since the end of the 1920s with the work of Wishart [14] on fixed-size matrices with Gaussian entries. The first asymptotic results on the limiting spectrum of large random matrices were obtained by Wigner in the 1950s in a series of papers motivated by nuclear physics. Since then, research on the limiting spectral analysis of large-dimensional random matrices has continued to attract interest in probability, statistics, and physics.

Most among the earliest results in random matrix theory pertain to the eigenvalues of square matrices with independent entries. However, key problems in wireless communications involve the singular values of rectangular matrices [12], and, even if those matrices have independent entries, their Grammians, whose eigenvalues are of interest, do not have independent entries.

The asymptotic theory of singular values of rectangular matrices has concentrated on the case where the matrix aspect ratio (ratio between the number of columns and that of rows) converges to a constant as the size of the matrix grows. The first success in the quest for the limiting empirical singular value distribution of rectangular random matrices is due to Marčenko and Pastur in 1967, from which several generalizations have been produced in more recent years.

The matrix in [8], while not requiring its elements to be i.i.d., ends up with a Marčenko–Pastur distributed Grammian. For more general classes of random matrices, however, not always the AESD is available. Rather, indirect characterizations making use of the Stieltjes transform, which uniquely determines the distribution function, have been derived in several cases [12, Ch. 2].

A particularly appealing application of the Stieltjes transform approach is the characterization of the eigenvalues of the sum of two random matrices, under some assumptions on their entries distributions (or moments at least). In this context, the so-called *freeness* (i.e., the noncommutative analog of independence between two random variables) among pairs of random matrices is largely exploited. To exemplify, let us recall that, in general, one is not allowed to infer about the asymptotic spectrum of a sum of two random matrices, starting from the individual spectra, but for the case of independent diagonal matrices. When the random matrices are asymptotically free [13], the asymptotic spectrum of the sum can also be obtained from the individual asymptotic spectra.

We will build our conjecture on the behavior of the MSE in irregular sampling when the spectrum components of the sensed signal are not independent, relying on existing results on the Stieltjes transform of free random matrices.

3.2 Random Matrix Transforms and Freeness

We provide here some further definitions involving random matrices and recall a milestone theorem to be used in Section 4.

Definition 36.1 Given a square random matrix $\mathbf{A}$ of size N, define

$$\phi(\mathbf{A}) = \lim_{N\to\infty} \frac{1}{N} \mathsf{Tr}\left\{\mathbb{E}\left[\mathbf{A}\right]\right\} \tag{36.4}$$

The expected asymptotic p th moment of $\mathbf{A}$ is given by $\phi(\mathbf{A}^p)$ and $\phi(\mathbf{I}) = 1$.

Definition 36.2 Two Hermitian random matrices $\mathbf{A}$ and $\mathbf{B}$ are *asymptotically free*, if $\forall \ell$ and for all polynomials $p_i(\,\cdot\,)$ and $q_i(\,\cdot\,)$, with $1 \le i \le \ell$ such that

$$\phi(p_i(\mathbf{A})) = \phi(q_i(\mathbf{B})) = 0$$

we get

$$\phi(p_1(\mathbf{A})q_1(\mathbf{B}) \ldots p_\ell(\mathbf{A})q_\ell(\mathbf{B})) = 0$$

Definition 36.3 Let us consider an analytic function $g(\,\cdot\,)$ in $\mathbb{R}^+$. Let $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\dagger$ be a random positive definite Hermitian $N \times N$ matrix, where $\mathbf{U}$ is the eigenvectors matrix of $\mathbf{A}$ and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{A}$. Then, by using the result for symmetric matrices in [1, Ch. 6] combined with the result in [10, p. 481], we have

$$\phi(g(\mathbf{A})) = \mathbb{E}\left[g(\lambda)\right] \tag{36.5}$$

where the random variable λ is distributed as the asymptotic eigenvalues of $\mathbf{A}$.

Definition 36.4 The η-transform of the random matrix $\mathbf{A}$ is defined as

$$\eta_{\mathbf{A}}(\gamma) = \mathbb{E}\left[\frac{1}{1+\gamma\lambda}\right] \tag{36.6}$$

where γ is a scalar and λ is a random variable distributed as the asymptotic spectrum of $\mathbf{A}$.

Definition 36.5 The Stieltjes transform of the random matrix $\mathbf{A}$ is defined as

$$\mathcal{S}_{\mathbf{A}}(z) = \mathbb{E}\left[\frac{1}{\lambda - z}\right]$$

where z is a complex variable.

We can then explicit the relationship between the two transforms as

$$\eta_{\mathbf{A}}(\gamma) = \frac{\mathcal{S}_{\mathbf{A}}\left(-\frac{1}{\gamma}\right)}{\gamma} \tag{36.7}$$

The Stieltjes transform is involved in the following theorem which will be of particular interest in our study.

Theorem 36.1 [12, Th. 2.38] *Let* $\mathbf{G}$ *be a* $(2M+1)^d \times r$ *matrix whose entries are i.i.d. complex random variables with zero mean and variance* $\frac{1}{2M+1}$. *Let* $\mathbf{W}_0$ *be a* $(2M+1)^d \times (2M+1)^d$ *Hermitian complex random matrix with empirical eigenvalue distribution converging almost surely to a nonrandom distribution whose Stieltjes transform is* $\mathcal{S}_{\mathbf{W}_0}$. *If* $\mathbf{G}$ *and* $\mathbf{W}_0$ *are independent, the empirical eigenvalue distribution of*

$$\mathbf{W} = \mathbf{G}\mathbf{G}^{\dagger} + \mathbf{W}_0$$

converges, as r and $(2M+1)^d \to \infty$, *with* $\frac{(2M+1)^d}{r} = \beta$, *almost surely to a nonrandom limiting distribution whose Stieltjes transform* $\mathcal{S}_{\mathbf{W}}(z)$ *satisfies*

$$\mathcal{S}_{\mathbf{W}}(z) = \mathcal{S}_{\mathbf{W}_0}\left(z - \frac{1}{\beta(1+\mathcal{S}_{\mathbf{W}}(z))}\right)$$

4 Main Results and Applications

Let us consider the sensor networks introduced in Section 2 and the signal model in (36.1). By applying the linear reconstruction technique described in Section 2.2, the

MSE provided by the LMMSE filter is given by (36.2). Asymptotically, from the definitions of Section 3.2 and from the expression of the asymptotic MSE in (36.3), we write

$$\begin{aligned}
\mathrm{MSE}_\infty &= \lim_{\substack{M,r\to+\infty \\ \beta}} \frac{\alpha}{(2M+1)^d} \mathsf{Tr}\left\{\mathbb{E}\left[\left(\mathbf{R}+\alpha\mathbf{C}^{-1}\right)^{-1}\right]\right\} \\
&\overset{(a)}{=} \alpha\,\phi\left(\left(\mathbf{R}+\alpha\mathbf{C}^{-1}\right)^{-1}\right) \\
&\overset{(b)}{=} \phi\left((\mathbf{W}/\alpha+\mathbf{I})^{-1}\right) \\
&\overset{(c)}{=} \mathbb{E}\left[\frac{1}{1+\lambda_{\mathbf{W}}/\alpha}\right] \\
&\overset{(d)}{=} \eta_{\mathbf{W}}\left(\frac{1}{\alpha}\right)
\end{aligned}$$

where in (a) we used (36.4), in (b) we defined $\mathbf{W} = \mathbf{R} + \mathbf{W}_0$ and $\mathbf{W}_0 = \alpha(\mathbf{C}^{-1} - \mathbf{I})$, in (c) we used (36.5) with $g(\lambda) = (1+\lambda/\alpha)^{-1}$, and in (d) we employed the definition of the η transform (36.6). Clearly, when $\mathbf{C} = \mathbf{I}$ we have $\mathbf{W}_0 = \mathbf{0}$, $\mathbf{W} = \mathbf{R}$, and $\mathrm{MSE}_\infty = \eta_{\mathbf{R}}(1/\alpha)$.

As is evident from the above expression, to compute the asymptotic MSE, we would need the asymptotic eigenvalue distribution of the matrix $\mathbf{W}$, which is still unknown. We therefore need to find an alternative way to compute the MSE_∞. Our idea would be to apply Theorem 36.1 and write the asymptotic MSE by using the Stieltjes transform of $\mathbf{W}_0$, but the i.i.d. constraint on the entries of matrix $\mathbf{G}$ is not fulfilled.

In [5], it has been shown that, as the field dimension d tends to infinity, the spectrum moments of its Grammian tend to those of the Marčenko–Pastur distribution. There is no mathematical justification that could allow to consider as equivalent (with respect to matrix transforms evaluation) matrices which have the same asymptotic spectra, since their eigenvectors would always come into play, invalidating any rigorous proof. Nevertheless, pushed by the nice spectral behavior of the matrix studied in [8], we performed an extensive numerical investigation and formulated the following conjecture.

Conjecture 36.1 Let $\mathbf{G}$ be an $(2M+1)^d \times r$ matrix whose Grammian asymptotic spectrum follows the Marčenko–Pastur law. Let $\mathbf{W}_0$ be an $(2M+1)^d \times (2M+1)^d$ Hermitian complex random matrix with empirical eigenvalue distribution converging almost surely to a nonrandom distribution whose Stieltjes transform is $\mathcal{S}_{\mathbf{W}_0}$. Let

$$\mathbf{W} = \mathbf{G}\mathbf{G}^\dagger + \mathbf{W}_0$$

If $\mathbf{G}$ and $\mathbf{W}_0$ are independent, the Stieltjes transform $\mathcal{S}_{\mathbf{W}}(z)$ converges, as r and $(2M+1)^d \to \infty$, with $\frac{(2M+1)^d}{r} = \beta$, to the function

$$\mathcal{S}^f_{\mathbf{W}}(z) = \mathcal{S}_{\mathbf{W}_0}\left(z - \frac{1}{\beta(1+\mathcal{S}^f_{\mathbf{W}}(z))}\right)$$

where the superscript ^{f} is reminiscent of the fact that the result would have been rigorous if the two random matrices $\mathbf{G}$ and $\mathbf{W}_0$ were free.

From the relationship between η and Stieltjes transforms in (36.7) and by using the above conjecture, we write

$$\mathrm{MSE}_\infty = \eta^f_{\mathbf{W}}\left(\frac{1}{\alpha}\right) = \alpha\mathcal{S}_{\mathbf{W}_0}\left(-\alpha - \frac{\alpha}{\beta(\alpha+\eta^f_{\mathbf{W}}(1/\alpha))}\right) \tag{36.8}$$

Given α, β, and the expression of $\mathcal{S}_{\mathbf{W}_0}$, (36.8) can be solved numerically, thus providing an effective method to study the reconstruction quality of correlated signals. We provide some examples below.

4.1 Examples of Applications

Non-flat spectrum: We assume $\mathbf{C}$ to be a diagonal matrix with C elements (out of $(2M+1)^d$) set to λ_1 and $(2M+1)^d - C$ terms set to λ_2. Because of the normalization constraint $\mathsf{Tr}\{\mathbf{C}\} = (2M+1)^d$ we have

$$\frac{C}{(2M+1)^d}\lambda_1 + \frac{(2M+1)^d - C}{(2M+1)^d}\lambda_2 = 1 \tag{36.9}$$

For simplicity, we define $c = \frac{C}{(2M+1)^d}$ so that (36.9) becomes

$$c\lambda_1 + (1-c)\lambda_2 = 1$$

Since $\mathbf{W}_0 = \alpha(\mathbf{C}^{-1} - \mathbf{I})$, asymptotically, the probability density function of $f_{\mathbf{W}_0}$ is given by

$$f_{\mathbf{W}_0}(\lambda) = c\delta(\lambda - \tilde{\lambda}_1) + (1-c)\delta(\lambda - \tilde{\lambda}_2)$$

where $\tilde{\lambda}_{1,2} = \alpha\left(1/\lambda_{1,2} - 1\right)$. Therefore, we obtain

$$\begin{aligned}\mathcal{S}_{\mathbf{W}_0}(z) &= \mathbb{E}\left[\frac{1}{\lambda - z}\right] \\ &= \int \frac{1}{\lambda - z} f_{\mathbf{W}_0}(\lambda)\,\mathrm{d}\lambda \\ &= \frac{c}{\tilde{\lambda}_1 - z} + \frac{1-c}{\tilde{\lambda}_2 - z} \\ &= \frac{c}{\alpha(\frac{1}{\lambda_1} - 1) - z} + \frac{1-c}{\alpha c \frac{\lambda_1 - 1}{1 - c\lambda_1} - z} \\ &= \frac{c\lambda_1}{\alpha(1-\lambda_1) - z\lambda_1} + \frac{(1-c)(1-c\lambda_1)}{\alpha c(\lambda_1 - 1) - z(1-c\lambda_1)} \end{aligned} \tag{36.10}$$

In Fig. 36.1, we show the field reconstruction MSE as a function of the SNR_m, for varying values of d and β, for $c = 1/16$ and $\lambda_1 = 15$. The lines without markers refer to the results obtained by numerical evaluation of the analytic expression (36.8), when $S_{\mathbf{W}_0}$ is given by (36.9). The lines with markers show the MSE obtained by computing MSE/σ_a^2 (see (36.2)) and averaging over several realizations of the matrix $\mathbf{R}$. Notice that for $\beta = 0.8$, the analytic approximation tightens as d increases and is very accurate for $d = 3$, especially for low to moderate values of SNR_m. For $\beta = 0.2$, the approximation is very tight already for $d = 1$. Indeed, as d increases the moments of the asymptotic eigenvalue distribution of $\mathbf{R}$ tend to the Marčenko–Pastur law [9] and the conjecture of Section 4 is satisfied.

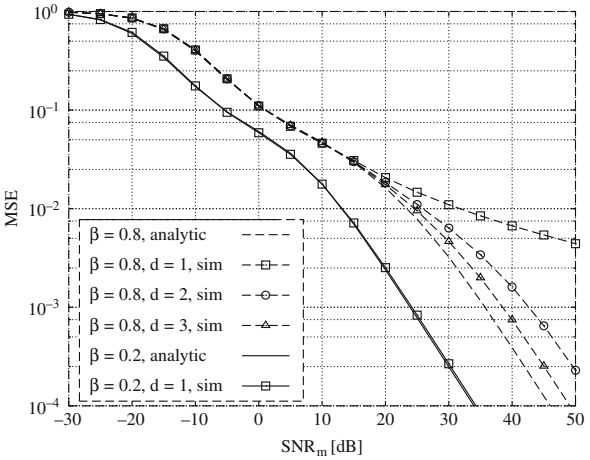


Fig. 36.1 MSE of the reconstructed field for varying values of d and β. Comparison between the numerically evaluated MSE and the approximated analytical expression derived using the Stieltjes transform method (36.9)

Constant correlation model: The constant correlation model (see, e.g., [11, Example IV]), usually exploited to model correlation between closely spaced antennas, represents a sort of worst-case correlation. Indeed, as opposite to the previous

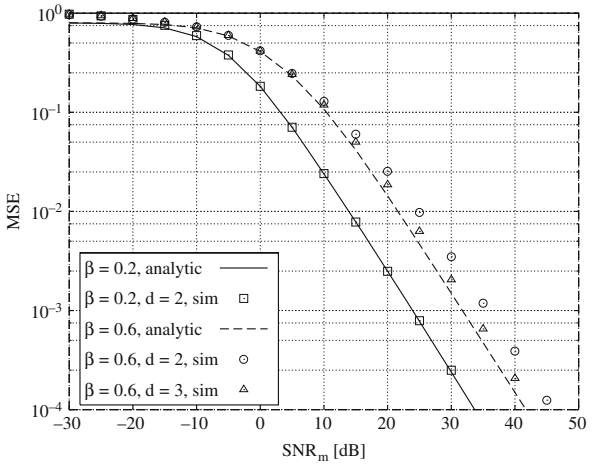


Fig. 36.2 MSE of the reconstructed field for $\rho = 0.2$ and varying values of d and β. The numerically evaluated value of the asymptotic MSE is compared against the approximated analytical expression derived using the Stieltjes transform (36.9)

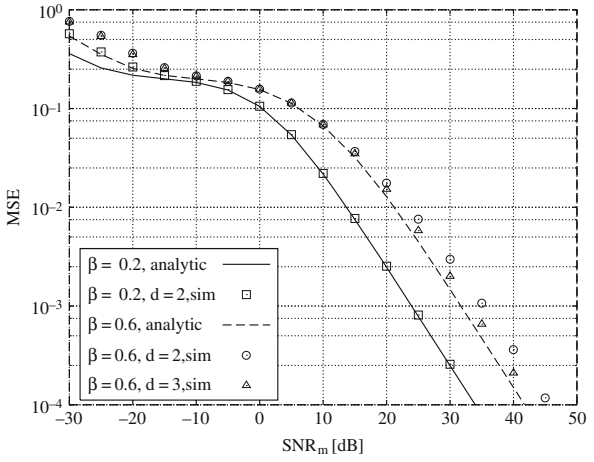


Fig. 36.3 MSE of the reconstructed field for $\rho = 0.8$ and varying values of d and β. The numerically evaluated value of the asymptotic MSE is compared against the approximated analytical expression derived using the Stieltjes transform (36.9)

example, all the off-diagonal entries of $\mathbf{C}$ are equal to an average value of the correlation coefficient. Specifically, being the constant correlation model in force, we consider a value $\rho \in [0,1)$ and assume $\mathbf{C}_{i,j} = \rho$, $i \neq j$. For the diagonal elements, we have $\mathbf{C}_{i,i} = 1$. This matrix has then only two distinct eigenvalues, $(1-\rho+(2M+1)^d\rho)$ with multiplicity 1 and $(1-\rho)$ with multiplicity $(2M+1)^d-1$. We can proceed similarly to what we have done in the previous example and compute the asymptotic MSE.

Figures 36.2 and 36.3 show the MSE of the reconstructed field for $\rho = 0.2$ and $\rho = 0.8$, respectively. Different values of d and β are considered. Looking at the plots, we note that even in the case of a non-diagonal correlation matrix, there is an excellent agreement between the proposed approximate solution (solid and dashed curves) and the numerical values of the asymptotic MSE (markers). Indeed, although for $\rho = 0.8$ the approximation is loose for very low values of the SNR_m, only values of SNR_m greater than 0 dB are of practical interest. Finally, we observe that the gap between analytical and simulation results is very limited even for large values of β, i.e., as the ratio of the number of signal harmonics to the number of available samples increases.

5 Conclusions

We considered a wireless sensor network sampling a multidimensional field with irregularly spaced samples, and we investigated the MSE of the reconstruction when the coefficients of the field spectrum are correlated. In particular, we performed an asymptotic analysis of the MSE which required concepts from the random matrix theory field. We then formulated a conjecture which provides a tight approximation of the asymptotic MSE. We showed the validity of our conjecture through some examples of correlated signals.

Acknowledgments This work was supported partially by the Italian Ministry for University and Research through the PRIN CARTOON 2006 project and partially by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM++ (contract n. 216715).

References

1. R. Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, 2nd edition, 1970.
2. H.G. Feichtinger, K. Gröchenig, and T. Strohmer. Efficient numerical methods in nonuniform sampling theory. *Numerische Mathematik*, 69:423–440, 1995.
3. D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin. Coping with irregular spatio-temporal sampling in sensor networks. *ACM SIGCOMM*, pp. 125–130, Jan. 2004.
4. S.M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1993.
5. V.A. Marčenko and L.A. Pastur. Distribution of eigenvalues for some sets of random matrices. *USSR Sbornik*, 1:457–483, 1967.
6. A. Nordio, C.F. Chiasserini, and A. Muscariello. Signal compression and reconstruction in clustered sensor networks. In *IEEE ICC*, Beijing, China, May 2008.
7. A. Nordio, C.F. Chiasserini, and E. Viterbo. The impact of quasi-equally spaced sensor layouts on field reconstruction. In *International Symposium on Information, Processing in Sensor Networks (IPSN 2007)*, Cambridge, MA, Apr. 2007.
8. A. Nordio, C.F. Chiasserini, and E. Viterbo. Performance of linear field reconstruction techniques with noise and uncertain sensor locations. *IEEE Transactions on Signal Processing*, 56(8):3535–3547, Aug. 2008.

9. A. Nordio, C.F. Chiasserini, and E. Viterbo. Reconstruction of multidimensional signals from irregular noisy samples. *IEEE Transactions on Signal Processing*, 56(9):4247–4285, Sep. 2008.
10. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes*. Cambridge University Press, 2nd edition, 1997.
11. H. Shin and J.H. Lee. Capacity of multiple-antenna fading channels: Spatial fading correlation, double scattering, and keyhole. *IEEE Transactions on Information Theory*, 49,10(10): 2636–2647, Oct. 2003.
12. A. Tulino and S. Verdú. Random matrices and wireless communications. *Foundations and Trends in Communications and Information Theory*, 1(1), Jul. 2004.
13. D. Voiculescu. Limit laws for random matrices and free products. *Inventiones Mathematicae*, 104(1):201–220, 1991.
14. J. Wishart. The generalized product moment distribution in sample from a normal multivariate population. *Biometrika*, 20:32–52, 1928.

Chapter 37
Hybrid Zigbee–RFID Networks for Energy Saving and Lifetime Maximization

P. Medagliani, G. Ferrari, and M. Marastoni

Abstract Zigbee is the standard of choice in wireless personal area networks with low power consumption, such as, sensor and control wireless networks. Since nodes may be positioned in non-easily accessible places, a high energy efficiency is required in order to maximize the network lifetime and minimize the maintenance costs. A simple and straightforward solution to maximize the network lifetime consists in turning off all nodes which are not needed, e.g., when the node spatial density is higher than that required to satisfy the sensing requirements. We propose an innovative radio-switched Zigbee network, where remote sensor nodes are selectively turned off. More precisely, the radio control is based on the use of radio frequency identification (RFID) technology, leading to a hybrid Zigbee/RFID architecture. In other words, we consider two logically overlapped networks, RFID and Zigbee. The RFID network turns on/off the nodes of the Zigbee network through a power-off algorithm, referred to as *deep sleep* algorithm, designed to equalize the residual energy in each Zigbee node. In fact, the RFID controller (i.e., the reader) cyclically switches off the Zigbee nodes with low amounts of residual energy. By building upon the proposed RFID-controlled Zigbee networks, we focus on applications which require a minimum *local* spatial density of observations. This is of interest, for instance, in distributed monitoring applications, where one needs to monitor the largest possible area in a homogeneous way. In this case, we introduce a *virtual spatial grid* over the monitored region, and we apply the deep sleep algorithm cell by cell of the grid, requiring that at most one node per cell is active at a time. The proposed hybrid Zigbee/RFID networks are analyzed through Opnet-based simulations.

Keywords ZigBee · RFID · Hybrid networks · Energy efficiency · Wireless sensor networks

P. Medagliani (✉)
Wireless Ad-hoc and Sensor Networks (WASN) Laboratory, Department of Information Engineering, University of Parma, I-43100, Parma, Italy
e-mail: paolo.medagliani@unipr.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_37,

1 Introduction

Wireless sensor networks are an interesting research topic, both in military [1, 2, 4] and is civilian scenarios [7, 23]. In particular, remote/environmental monitoring, surveillance of reserved areas, etc., are important fields of application of wireless sensor networking techniques. These applications often require very low power consumption and low-cost hardware [18].

One of the newest standards for wireless networking with low transmission rate and high energy efficiency has been proposed by the Zigbee Alliance [28, 14]. An experimental analysis of Zigbee wireless sensor networks (WSNs), taking into account the impact of the most important system parameters (e.g., the received signal strength indication (RSSI), throughput, network transmission rate, and delay) is presented in [14, 17, 11]. One of the most interesting research direction for WSNs is the design of network architectures with high energy efficiency. In [5], the authors analyze different approaches and possible optimization strategies in order to reduce the power consumption of IEEE 802.15.4 networks. In [22], instead, a mechanism for shutting down the radio frequency interface of wireless sensors, in order to reduce power consumption, is presented.

On the other side, radio frequency identification (RFID) devices are also receiving more and more attention, by both industrial and scientific communities. In particular, they can be used for luggage identification in airports, biological materials identification in hospitals, monitoring of post parcels, tracking of livestocks, efficient monitoring of objects in supply chains, etc. [12, 19, 25, 26]. One of the newest areas of interest for the RFID technology is pervasive computing, typically carried out integrating different technologies, such as RFID devices and WSNs [20]. In [27, 24], the authors propose and evaluate three different system architectures in order to combine WSNs with RFID systems.

We first introduce a logical scheme for the integration of Zigbee and RFID networks to maximize the battery lifetime of the nodes in the former network. Then, we analyze the performance of this integrated network in terms of energy consumption, considering a simple deep sleep algorithm, such that nodes are cyclically turned off once their battery energy becomes lower than a threshold level, which is adaptively adjusted during network evolution. Finally, in order to monitor the network surface uniformly, we introduce a virtual spatial grid and apply the deep sleep algorithm cell by cell of this grid.

The structure of this work is the following. In Section 2, a short overview on the Zigbee standard is provided, whereas in Section 3 we describe the RFID technology. In Section 4.1, we present the selective wake-up idea upon which our model is based, and in Section 4.2 the Opnet simulator structure of the integrated Zigbee–RFID network is presented. In Section 4.3, the implementation of the deep sleep algorithm is considered and in Section 4.4 the concept of virtual spatial grid is presented. In Section 5, the performance of hybrid Zigbee–RFID networks is analyzed. Finally, Section 6 contains concluding remarks.

2 Zigbee Standard Overview

The Zigbee standard is suited for the family of low-rate wireless personal area networks (LR-WPANs), allowing network creation, management, and data transmission over a wireless channel with the highest possible energy savings. Three different types of nodes are foreseen by the Zigbee standard: (i) *coordinator*, (ii) *router*, and (iii) *end device*. In the absence of a direct communication link, the router is employed to relay the packets toward the correct destination. The coordinator, in addition to relaying the packets, can also create the network, exchange the parameters used by the other nodes to communicate (e.g., a network identifier (ID), a synchronization frame), and send network management commands. The router and coordinator are referred to as *full function devices* (FFDs), i.e., they can implement all the functions required by the Zigbee standard in order to set up and maintain communications. The end devices, which are also referred to as *reduced function devices* (RFDs), can only collect data from sensors, insert these values into proper packets, and send them to destination nodes.

The Zigbee standard is based, at the first two layers of the ISO/OSI stack, on the IEEE 802.15.4 standard [3], which employs a *non-persistent* carrier sense multiple access with collision avoidance (CSMA/CA) medium access control (MAC) protocol and operates in the 2.4 GHz band (similarly to the IEEE 802.11 standard [15]). In addition, the IEEE 802.15.4 standard provides an optional ACK message to confirm the correct delivery of a packet. We remark that the medium access mechanism in Zigbee wireless networks makes use of a *back-off algorithm* to reduce the number of packet collisions. A node, before transmitting a new packet, waits for a period whose length is randomly chosen within an interval defined during the network start-up phase. After this period of time has elapsed, the node tries to send its packet: if it detects a collision, it doubles the previously chosen interval and waits; if the channel is free, instead, it transmits its packet. This procedure is repeated for five times, after which the waiting interval is kept fixed to its maximum value. This back-off algorithm makes it likely, in the considered scenarios (low traffic load), that a node will eventually manage to transmit its packet.

The Zigbee standard provides a technique which improves the synchronization between RFDs. This technique is based on the subdivision of the time into *slots* and the transmission of a periodic signal, referred to as *beacon*, from the coordinator of the network. In particular, the standard defines a specific frame structure, referred to as *superframe*, shown in Fig. 37.1. This frame is divided into two main zones: (i) active and (ii) passive. The active zone is divided, in turn, into two periods: (i) the contention access period (CAP) and (ii) the contention free period (CFP). In the CAP, the RFDs access the channel according to the CSMA/CA protocol, whereas in the CFP only the RFDs that have previously reserved a guaranteed time slot (GTS) can access the channel, thus preventing the collisions. Finally, in the passive zone all the RFDs switch into *sleep* state in order to improve the energy saving.[1] According to the Zigbee notation, the time interval between two subsequent beacons is referred

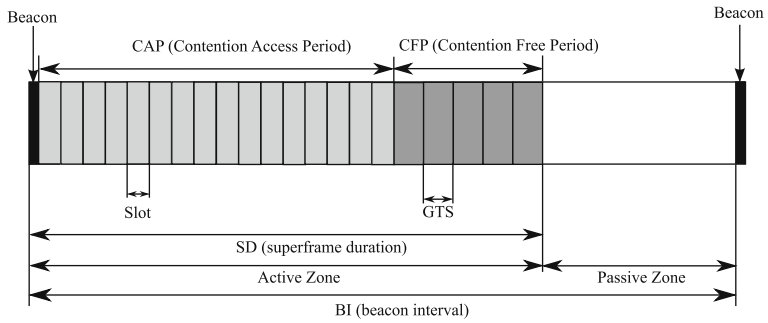


Fig. 37.1 Structure of the superframe used by the Zigbee standard

to as beacon order (BO), whereas the duration of the active frame is referred to as superframe duration (SD).

3 RFID Technology

The RFID technology specifies two different devices: (i) the *reader* and (ii) the *transponder* or *tag*. The tag is an electronic device that stores data useful for identification and is placed on the object to be tracked. On the opposite, the reader is the device used to interrogate the tags. The tags can be classified into three main categories, according to the power source they are equipped with, as (i) *active*, (ii) *semi-passive*, and (iii) *passive* [12]. The active tags, in order to respond to the interrogation of the reader, use their own internal batteries for processing operations and signal transmission. The semi-passive tags, instead, use their batteries only to power the internal processor and not to broadcast the return signal. Finally, the passive tags, being not equipped with batteries, exploit the energy of the received signal to respond to the reader (*backscattering* technique). Obviously, the tags with longer lifetime (virtually infinite) are the passive ones, whereas the active tags have the longest transmission range. Once the reader has interrogated a tag and this has replied correctly without incurring into collisions with other tags (proper anti-collision mechanisms are used [6]), the reader sends an *acknowledgment* message to the tag to confirm correct message reception.

The functionalities provided by the RFID technology are mainly related to error-free communication between reader and tags (bidirectional), identification and communication with multiple tags, selection of a target set of tags, lock of data stored in the tags, and writing and overwriting operations in the tags. The communication from the reader to the tags, according to the backscattering technique, is realized through amplitude modulation of the interrogation signal. On the other hand, after the transmission of the interrogation signal, the reader emits a constant power signal which allows the tags to respond to the former signal through a load modulation

technique [12]. The reader receives data from the tags as a variation of the reflection of the constant power signal.

When the reader interrogates a set of tags, these may respond immediately, leading to a collision. One of the most important standards for the RFID technology is the ISO/IEC 18000-6 standard [13]. This standard defines a reference model for identification systems which operate in the range of ultra high frequencies (UHF). In particular, this standard regulates RFID systems in the frequency band around 868 MHz in Europe. The 18000-6 standard defines two different transmission types: (i) *type A* and (ii) *type B*. Both transmission types make use of a rate equal to 40 kbps and use a binary phase modulation. The difference between these two transmission types resides in the medium access control (MAC) protocol. If a collision is detected by the reader, the Type A tags retransmit the interrogation request according to the *Aloha* MAC protocol [9]. According to this protocol, a tag transmits as soon as a packet is generated and it assumes that a collision has happened if an acknowledgment message is not received after a given period of time. Type B tags, instead, use the binary tree protocol (BTP) [6] as anti-collision mechanism. The idea of the BTP is that, upon a collision, the set of tags is divided into two subsets: one set tries to retransmit their messages immediately, whereas the other set waits till a new interrogation request is generated by the reader. Eventually, only one tag will retransmit, and its identity (or any other stored information) will be successfully acquired by the reader. This recursive procedure is then repeated, under the constraint that the already censed tags do not respond to the interrogation signal.

4 Hybrid Zigbee–RFID Networks

4.1 System Model

Our system model consists of a network where N nodes are deployed to monitor a particular phenomenon of interest. Under the assumption that a minimum spatial density of observations is required, i.e., a minimum number $N_{\min}$ of RFDs needs to be used to monitor a given surface (with area A) of interest, and that the number N of deployed RFDs is larger than $N_{\min}$, our approach consists in implementing a *selective wake-up* of the RFDs. More precisely, our strategy consists in selectively turning on and off the RFDs in order to equalize the energy consumption of the nodes in the network. In order to maximize the network lifetime, as soon as the residual energy of an active RFD becomes lower than a fixed threshold E_{th}, the RFD is switched into the sleep state. At the same time, one of the remaining $N - N_{\min}$ RFDs (previously switched off) is woken up, so that an overall minimum spatial density of observations of the phenomenon of interest is guaranteed. This procedure is referred to as *deep sleep algorithm*. An illustrative example of the status, at a given time, of a network where the deep sleep algorithm is used, is shown in Fig. 37.2. We point out that the RFDs are only switched into the sleep state, instead of being

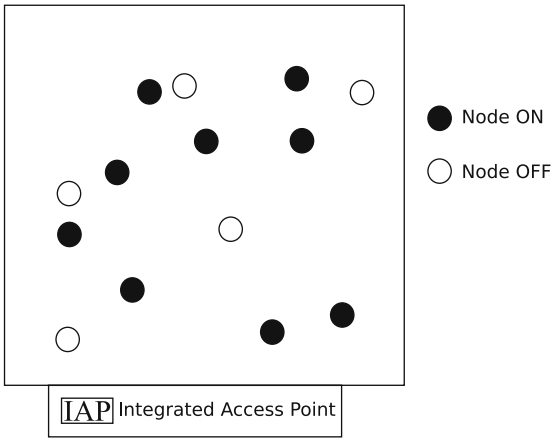


Fig. 37.2 Network scenario with $N = 14$ nodes, out of which $N_{\min} = 9$ are active and $N - N_{\min} = 5$ have been turned off by the deep sleep algorithm

turned off, in order to prevent them from losing synchronization with the coordinator and the other RFDs in the network. In fact, according to the Zigbee standard, the network joining operations performed when an RFD is turned on to introduce a delay longer than the length of the wake-up operations, since the RFD must wait for a new beacon in order to synchronize with the network before starting its transmission. On the opposite, the RFDs in the sleep state switch to the active state only when a beacon transmission is scheduled and subsequently, if not required, return into the sleep state without losing their synchronization with the network. The goal of the deep sleep algorithm is energy equalization among the RFDs, so that the residual energies of the RFDs are balanced. When an RFD is in the sleep state, in fact, its energy consumption is four orders of magnitude lower than in the active state [8, 16].

4.2 Opnet Simulator Structure

Our hybrid Zigbee–RFID model is based on an IEEE 802.15.4 Opnet model developed at the University of Porto, Portugal [21], and on an RFID Opnet model developed at the University of Parma, Italy [10]. The IEEE 802.15.4 model is beaconed in order to synchronize the nodes in the network and save energy. In addition, this model contains a battery module which is used to evaluate the energy consumption of the devices.[2] On the other side, the Opnet RFID model implements the ISO/IEC 18000-6 standard, considering faded wireless channels. Since the Zigbee and RFID models are disjoint, we have developed a hybrid Opnet Zigbee–RFID model where the subcomponents communicate and cooperate.[3] Obviously, the beaconing mechanism is modified in order to take into account also the RFDs which are not transmitting during a frame. Note that in our system the BTP (embedded in

the ISO/IEC 18000-6 standard) is not used, since the RFID tags are used only as switches for the (Zigbee) RFDs and do not need to be identified. In Fig. 37.3a, the logical scheme of the integrated Zigbee–RFID network is shown. More precisely, the RFID network lays on top of the Zigbee network. In our hybrid system, the information is transferred through the IEEE 802.15.4 (logical) network which is, in turn, controlled by the RFID network. In Fig. 37.3b, instead, we show the integrated devices used in our network: the *integrated AP*, which is obtained from the combination of an RFID reader and a Zigbee coordinator and *integrated node*, which is obtained from the combination of an RFID tag and a Zigbee RFD. We point out that the integrated node is battery powered, whereas the integrated AP is supposed to be connected to the electrical system, so that battery exhaustion is not an issue for the latter, but only for the former.

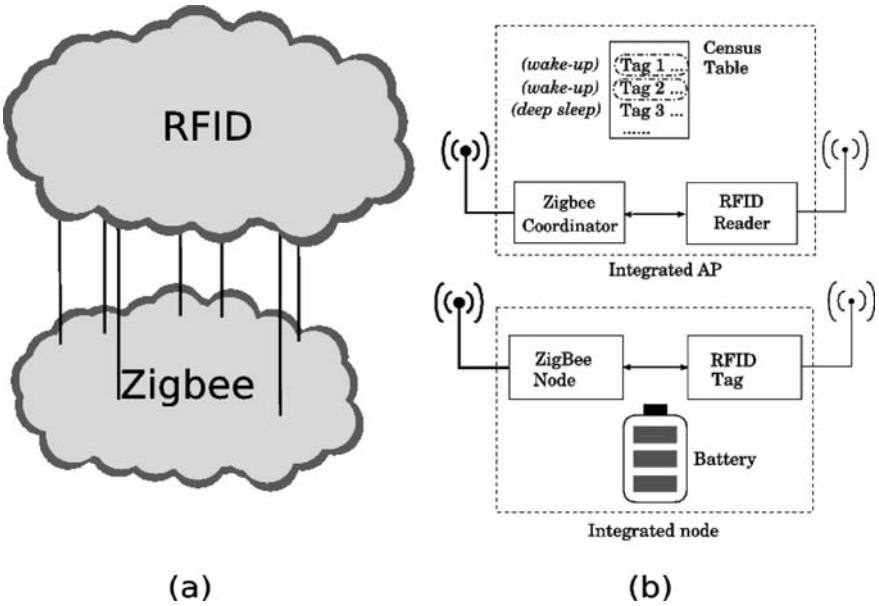


Fig. 37.3 Logical scheme of the integrated network: (**a**) the Zigbee and RFID networks are combined together; (**b**) the integrated nodes and AP are obtained from the integration of Zigbee and RFID devices

4.3 Deep Sleep Algorithm

We now describe the basic functionalities of a hybrid Zigbee–RFID network. When an RFD is in the sleep state and is selected by the coordinator, the associated RFID tag receives the signaling message from the reader (through the logical RFID network) and switches on its RFD, which enters into the active state and starts communicating (through the logical Zigbee network). Similarly, when an RFD is active and

its residual energy becomes lower than a threshold value, the RFD communicates it to the Zigbee coordinator. The coordinator informs its associated RFID reader, which, by sending a proper message to the tag, forces the selected RFD into the sleep state. Since the deep sleep algorithm is managed by the coordinator of the Zigbee network, the active RFDs embed their residual energies inside the data packets sent to the coordinator. In this way, the coordinator is constantly aware of the residual energy of each active RFD. On the other hand, since an RFD in the sleep state does not transmit any packet, the coordinator also uses an estimation algorithm to predict the energy consumed by an RFD during the sleep state.

We now describe the main steps of the deep sleep algorithm through illustrative diagrams. Figure 37.4 refers to the operations involved when the network is started. All the RFDs are equipped with identical batteries and we refer to the initial energy

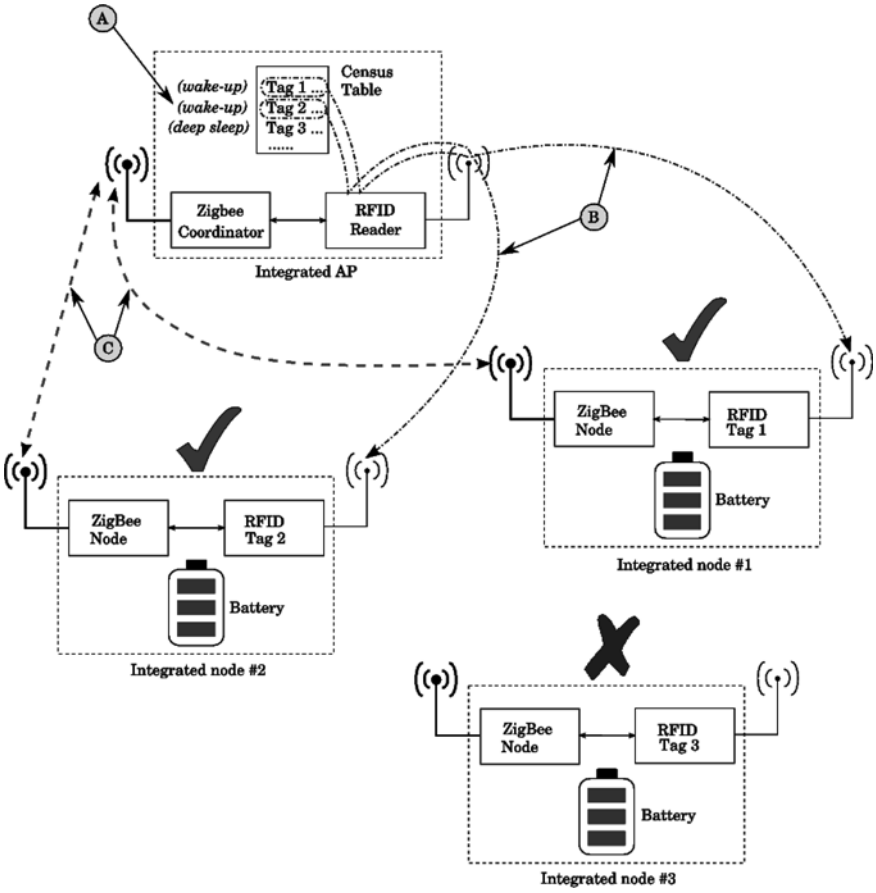


Fig. 37.4 Initial steps of the deep sleep algorithm for selective wake-up of remote nodes: (A) $N_{\min}$ out of N RFDs are selected by the AP; (B) the selected RFDs are notified through the associated RFID tags; (C) the activated RFDs start communicating

level as 100%. At the network startup, only $N_{\min}$ RFDs are turned on, whereas the remaining $N - N_{\min}$ nodes are turned into the sleep state. The integrated AP creates a *census table*, where information about the position, the residual energy, and the status of each RFD is stored. In our simulations, we assume that the position of each RFD is preliminarily available at the integrated AP, whereas the entry of the census table relative to the residual energy of an RFD is updated as soon as a packet (containing this information) is received from the node. If an RFD is in the sleep state, its residual energy value is estimated by the AP before scheduling the transmission of a new beacon.

As soon as an RFD is turned on, its residual energy E_r reduces and, consequently, its value in the census table is updated. The evolution of the network according to the deep sleep algorithm is illustrated in Fig. 37.5. When the energy of an RFD

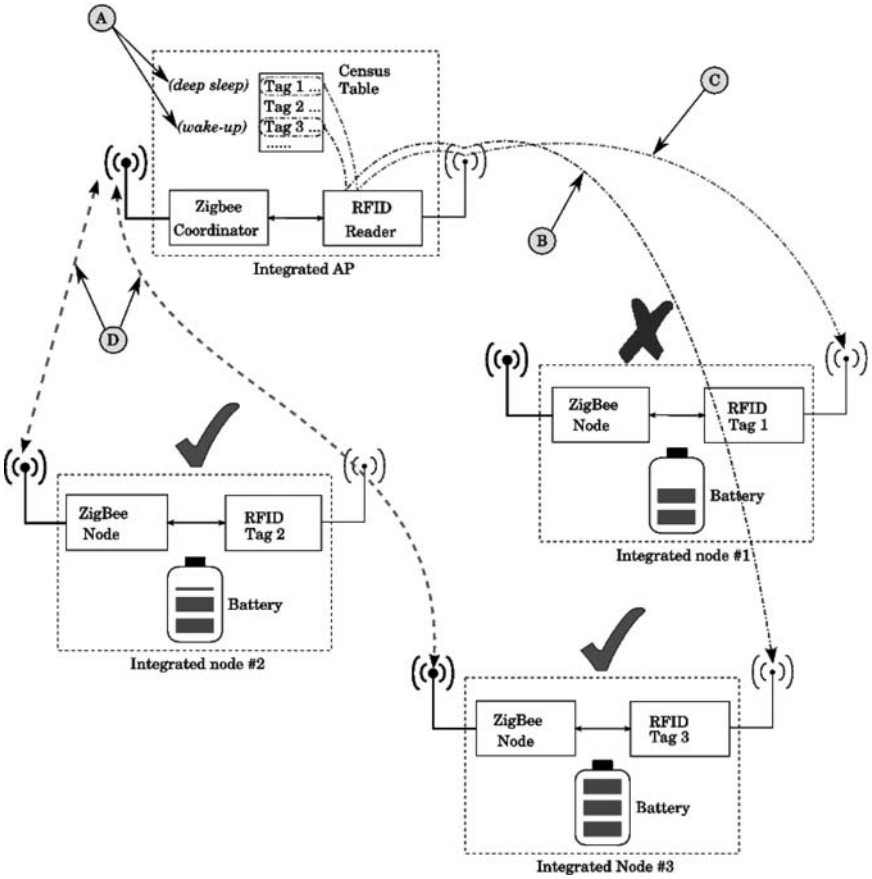


Fig. 37.5 Steps of the replacement of an active node with residual energy under threshold and a sleeping node with higher residual energy: (A) the AP identifies an RFD with low residual energy; (B) a sleeping node with high residual energy is turned on; (C) the selected active RFD with low residual energy is turned off; (D) the set of active nodes communicate

becomes lower than an initial threshold $E_{\text{th}}^{(0)}$, the integrated AP forces the RFD into the sleep state through its associated RFID tag. At the same time, the integrated AP forces one of the sleeping nodes with higher residual energy to wake up. This procedure is repeated until there are no more RFDs with residual energies higher than $E_{\text{th}}^{(0)}$. At this point, the threshold value of the residual energy is decreased of a pre-defined energy step E_{s}. More precisely, the new energy threshold is set to $E_{\text{th}}^{(1)} = E_{\text{th}}^{(0)} - E_{\text{s}}$. This procedure is repeated until the energy threshold is so low that data communications in the Zigbee logical network are no longer possible.[4] At this point, as soon as the minimum spatial density of observations is no longer guaranteed, the network is declared dead and the simulation stops. More details on the energy consumption at the nodes will be given in Section 5.1.

In general, N integrated nodes could be deployed randomly over a given surface. As an illustrative scenario, in Fig. 37.6, $N = 27$ nodes are deployed randomly over a square surface. The integrated AP is placed in the center of the monitored surface, whereas the $N = 27$ integrated nodes are deployed according to a 2D Poisson distribution.

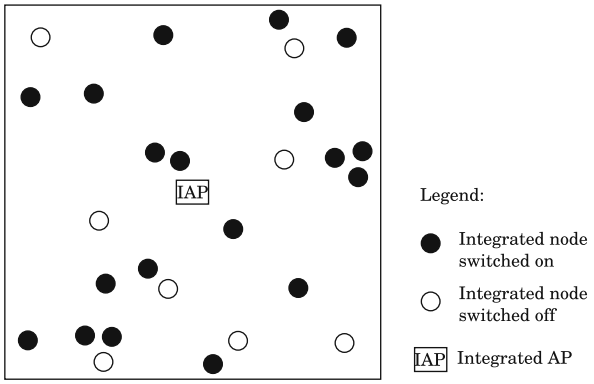


Fig. 37.6 Network scenario with $N = 27$ nodes, out of which $N_{\text{min}} = 19$ are active and $N - N_{\text{min}} = 8$ have been turned off by the deep sleep algorithm. The monitored surface is not divided into cells

4.4 Deep Sleep Algorithm with Virtual Spatial Grid

While the deep sleep algorithm takes into account the *overall* observation spatial density (considering the total number of nodes deployed over the entire network surface), in several applications it might be of interest to monitor *homogeneously* the network surface, i.e., to guarantee an approximately constant *local* observation spatial density across the surface. In order to do this, we follow an approach based on the use of a *virtual spatial grid* over the network.

If the monitored area is virtually partitioned into observation cells (with a local minimum required node spatial density per cell), then the "standard" deep sleep

algorithm can no longer be applied, since it could lead to turn off all nodes of the same cell. However, there might be scenarios where particularly critical phenomena need to be observed with high accuracy over the entire monitored surface. In Fig. 37.7, an illustrative example of the network in Fig. 37.6 with an overlaid virtual spatial grid is shown. As one can see, different cells may contain different numbers of nodes. In the following, we will denote the cells where there is only one integrated node as *secondary*, whereas the cells with more than one integrated node will be denoted as *primary*. In the presence of a virtual spatial grid, the deep sleep algorithm is applied in each primary cell, where the observations are redundant. In the secondary cells, no deep sleep algorithm is applied, since the integrated nodes cannot be turned off. The cells with no node are not relevant for monitoring purposes.

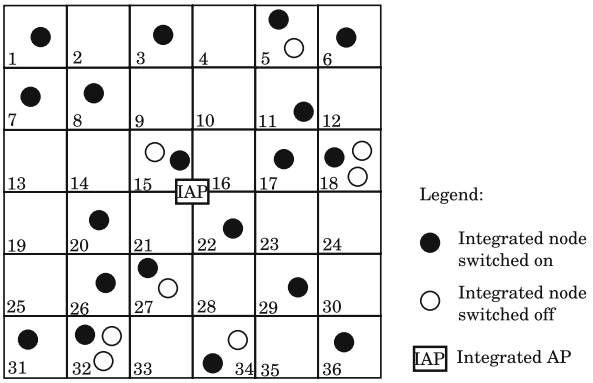


Fig. 37.7 Network scenario with $N = 27$ nodes, out of which $N_{\min} = 19$ are active and $N - N_{\min} = 8$ have been turned off by the deep sleep algorithm. The monitored surface is divided into cells and only one node per cell is active, according to the spatial grid procedure

The network is declared dead as soon as all the RFDs of one of the primary cells die. This criterion comes from the assumption that primary cells are assumed to be associated with critical zones of the phenomenon under observation, whereas the secondary cells may be associated with less critical zones. Thus, when a primary cell contains no more RFD, we assume that the observations are no longer reliable and the network is declared dead. We remark that our approach is valid also in scenarios where $N > N_{\min}$, but it is more effective in scenarios where $N \gg N_{\min}$.

5 Performance Analysis

The simulations have been carried out considering networks with $N = 40$ nodes deployed over a square surface with 6 m long sides. In all cases, the RFDs send packets directly to the integrated AP, i.e., all network realizations have star topologies. The packet interarrival time T_{int} is either fixed to 0.25 s or Poisson distributed

with average value equal to 0.25 s. The considered packet length is $L = 200$ bit/pck. We assume that the network is dead when the energy level of all RFDs becomes 80% of the initial value. Each performance simulation result is the average of the results obtained considering six realizations of the network topology (according to a 2D Poisson distribution).

Before starting the simulation-based performance analysis of hybrid Zigbee–RFID networks using the deep sleep algorithm, we provide the reader with some details about the energy consumed by each integrated node. The average residual energy[5] at a generic instant t can be expressed as follows:

$$E_{\mathrm{r}}(t) = E_{\mathrm{i}} - E_{\mathrm{c}}(t) \tag{37.1}$$

where E_{i} is the initial energy of a node and $E_{\mathrm{c}}(t)$ is the energy consumed at the instant t. In particular,

$$E_{\mathrm{c}}(t) = E_{\mathrm{c\_TX}}(t) + E_{\mathrm{c\_RX}}(t) + E_{\mathrm{c\_idle}}(t) + E_{\mathrm{c\_sleep}}(t)$$

where

$$\begin{aligned}
E_{\mathrm{c_TX}}(t) &= \lambda \cdot E_{\mathrm{c_TX_1_pck}} \cdot t = K_{\mathrm{TX}} \cdot t \\
E_{\mathrm{c_RX}}(t) &= \lambda \cdot (N-1) \cdot E_{\mathrm{c_RX_1_pck}} \cdot t = K_{\mathrm{RX}} \cdot t \\
E_{\mathrm{c_sleep}}(t) &\simeq E_{\mathrm{c_sleep_1_s}} \cdot \frac{\Delta t_{\mathrm{passivezone}}}{\Delta t_{\mathrm{superframe}}} \cdot t = K_{\mathrm{sleep}} \cdot t \\
E_{\mathrm{c_idle}}(t) &\simeq E_{\mathrm{c_idle_1_s}} \cdot \frac{\Delta t_{\mathrm{activezone}}}{\Delta t_{\mathrm{superframe}}} \\
&\quad \cdot \left[t - \Delta t_{\mathrm{TX_1_pck}} \cdot \lambda \cdot t - \Delta t_{\mathrm{RX_1_pck}} \cdot \lambda \cdot (N-1) \cdot t \right] \\
&= K_{\mathrm{idle}} \cdot t
\end{aligned}$$

λ is the packet transmission rate[6]; $E_{\mathrm{c\_TX\_1\_pck}}$ and $E_{\mathrm{c\_RX\_1\_pck}}$ are the energies consumed per packet transmission and reception acts, respectively; $E_{\mathrm{c\_sleep\_1\_s}}$ and $E_{\mathrm{c\_idle\_1\_s}}$ are the energies consumed during one second of persistence in the sleep and idle states, respectively; and $\Delta t_{\mathrm{activezone}}$, $\Delta t_{\mathrm{passivezone}}$, and $\Delta t_{\mathrm{superframe}}$ are the durations of the active zone, the passive zone, and the superframe, respectively. Therefore, we can approximate $E_{\mathrm{c}}(t)$ as

$$E_{\mathrm{c}}(t) \simeq \left[K_{\mathrm{TX}} + K_{\mathrm{RX}} + K_{\mathrm{sleep}} + K_{\mathrm{idle}} \right] \cdot t = K_{\mathrm{tot}} \cdot t \tag{37.2}$$

Using (37.2) into (37.1), the residual energy at time t can be rewritten as

$$E_{\mathrm{r}}(t) \simeq E_{\mathrm{i}} - K_{\mathrm{tot}} \cdot t \tag{37.3}$$

The final expression (37.3) for the residual energy is obtained under the assumption of no use of the deep sleep algorithm. However, if the deep sleep algorithm

is used, only the values of K_{sleep} and K_{idle} change, but the linear dependence of E_{r} from t still holds, as will be shown in the following.

5.1 Deep Sleep Algorithm

In Fig. 37.8, we evaluate the impact of the energy step E_{s} on the average (over the integrated nodes) residual energy in three different scenarios, considering $N = 40$ nodes in all cases: (i) with $N_{\min} = 40$ (i.e., the deep sleep algorithm is not used), (ii) with $N_{\min} = 30$, and (iii) with $N_{\min} = 25$. The packet generation rate is constant where packet interarrival time equal to 0.25 s is used. Observing the curves related to the scenarios with the deep sleep algorithm, it can be noted that the performance for a given value of $N_{\min}$ does not depend on E_{s}. This is to be expected, provided that the energy step is sufficiently smaller than the initial energy. In fact, when the deep sleep algorithm is used, $N_{\min}$ nodes are active at a time, and the value of E_{s} determines only the rate at which nodes get activated and deactivated. In particular, the following considerations can be carried out.

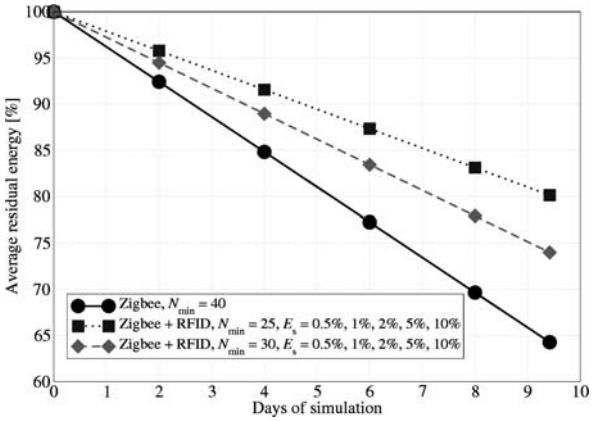


Fig. 37.8 Average residual energy as a function of the days of simulation, in scenario with $N = 40$ nodes. Various values (25, 30, 40) of $N_{\min}$ are considered for the application of the deep sleep algorithm. In the case with $N_{\min} < N$, considering different values of the energy step does not change the performance

- If E_{s} is very small, then the energy levels of the nodes are equalized in a very fine way. However, this implies that nodes will cycle between on and off states very often, i.e., the integrated AP will spend energy in sending control messages and in the processing required to track the network topology changes.
- On the other hand, if E_{s} is not too small, then energy equalization is "rougher" but the integrated AP will spend less energy in network management operations. Since energy consumption is typically not an issue for the integrated AP, "finer" energy equalization should be chosen.

To summarize, the network behavior of a hybrid Zigbee–RFID network using the deep sleep algorithm is very similar to a "standard" Zigbee network with $N_{\min}$ RFDs, regardless of the value of E_s.

In Fig. 37.9, the average residual energy is shown as a function of the number of days of simulation, for various values of the number $N_{\min}$ of active RFDs. We have considered only $E_s = 1\,\%$ because, as shown in Fig. 37.8, the average residual energy is independent of E_s. Observing the curves in Fig. 37.9, one can conclude that

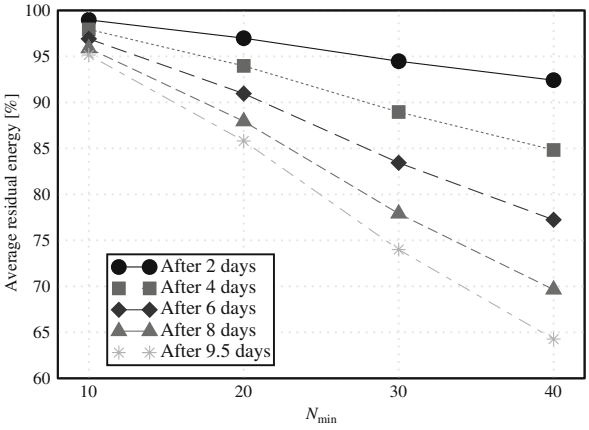


Fig. 37.9 Average residual energy, as a function of $N_{\min}$, considering various values for the number of days of simulation. In all cases, $N = 40$

- the smaller $N_{\min}$, the higher the energy saving, regardless of the simulation duration;
- the performance of the network without the deep sleep algorithm (i.e., the scenario with $N_{\min} = 40$) is the worst.

According to the deep sleep algorithm, in fact, when $N_{\min}$ is small, there is a large number of RFDs in the sleep state, and the energy consumption (averaged over all RFDs in the network) is, therefore, low. On the opposite, when $N_{\min}$ is large or, as a limiting case, the deep sleep algorithm is not used ($N = N_{\min}$), there is a large number of RFDs active in the network, and the energy consumption is high.

In Fig. 37.10, the average residual energy is shown as a function of the number of days of simulations, considering constant and Poisson packet generation distributions. As mentioned at the beginning of this section, the packet generation rate is $\lambda = 4$ pck/s – this is the *average* packet generation rate with Poisson distribution. As one can see from results in Fig. 37.10, the average residual energy is a linearly decreasing function of the time *regardless* of the packet generation distribution. However, for a given value of $N_{\min}$, it can be observed that a Poisson-distributed packet generation leads to a performance degradation with respect to the case with

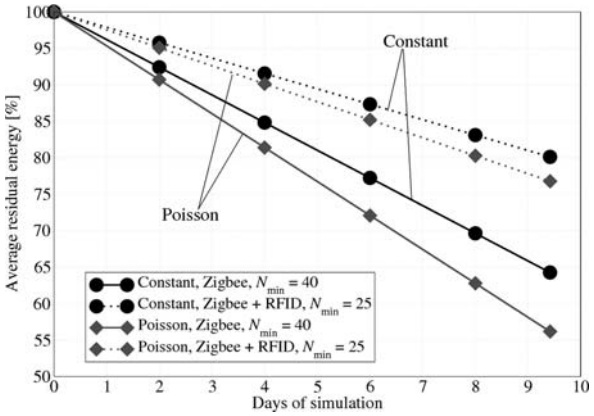


Fig. 37.10 Average residual energy, as a function of the number of days of simulation, for various packet generation distributions: (i) constant and (ii) Poisson. In all cases, $N = 40$, whereas $N_{\min}$ is set to either 25 or 40

constant packet generation, and this degradation is more pronounced the larger is $N_{\min}$. This is due to the higher traffic generated with the Poisson distribution. Each active node receives all the packets transmitted in the network and processes only those with a destination address equal to its own address. On the other hand, when a node is in the sleep state, no packet is received and energy is preserved.

Finally, we point out that, also with Poisson-distributed packet generation, different values of the energy step E_s do not influence the observed performance.

5.2 Impact of the Virtual Spatial Grid

In this section, we present performance results in scenarios where a virtual spatial grid is considered and the deep sleep algorithm is applied locally, as described in Section 4.4. The same network topology of Section 5.1 is considered and the monitored surface is divided into thirty-six 1 m^2 cells. We remark that, according to our algorithm, in the secondary cells the nodes are turned on according to the traditional superframe structure. In the primary cells, where the deep sleep algorithm is applied cell by cell, instead, there is at least one active node per cell, whereas the other nodes of the cell are cyclically switched into the sleep state.

We have considered different network configurations and we have compared the results in the presence of the virtual spatial grid with those of scenarios without it. For different types of networks, we have considered the same network topology realizations, in order to make the comparison fair.

In Fig. 37.11, we present the average number of active cells guaranteed by the two different node management mechanisms. On average, using the different topologies considered in our simulation framework, the virtual spatial grid procedure guarantees the coverage of 24 cells, whereas a scenario without virtual spatial

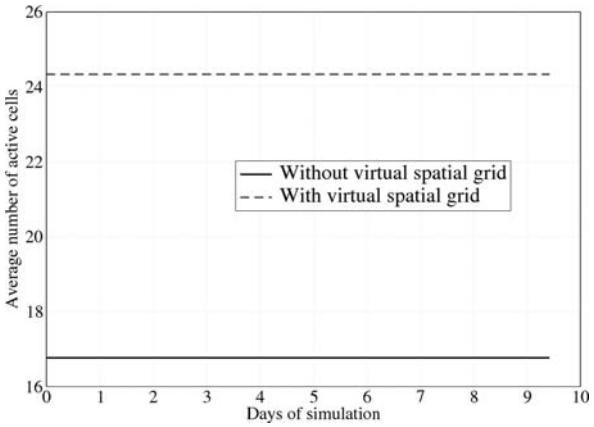


Fig. 37.11 Average number of active cells with and without the virtual spatial grid. In all cases, the deep sleep algorithm is used ($N_{\min} = 25$) and $N = 40$

grid procedure leads to an average of 17 active cells. As one can see, the average number of active cells remains constant over the simulation: this is due to the fact that in both cases the battery energy is not depleted.

In Fig. 37.12 the average energy consumption performance is shown as a function of the number of simulation days, considering scenarios without the deep sleep algorithm ($N_{\min} = 40$), with the deep sleep algorithm without virtual spatial grid

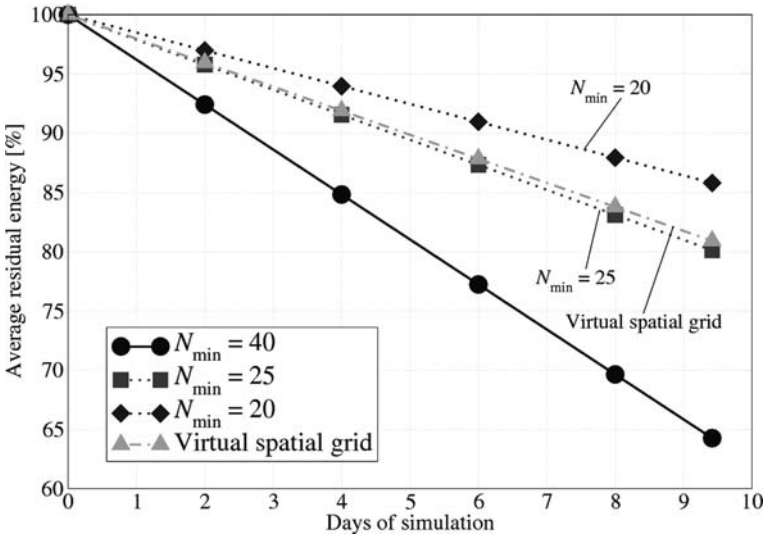


Fig. 37.12 Average residual energy with energy step equal to 1% of the initial energy level. The number of nodes is $N = 40$. The deep sleep algorithm is applied considering $N_{\min} = 20$, $N_{\min} = 25$, respectively. The performance with the virtual spatial grid is also shown

($N_{\min} = 20$ or $N_{\min} = 25$), and with the virtual spatial grid. The curves presented in this figure are obtained in a scenario with $E_s = 1\%$, because we have experienced that, as in the case with deep sleep algorithm, the energy step has no impact on the performance of networks with the virtual spatial grid. The energy consumption of a traditional Zigbee network is larger than that of networks with deep sleep algorithm both with and without virtual spatial grid. In particular, in scenarios where the virtual spatial grid is applied, on average there are 24 active nodes, and the average residual energy trend is similar to that of the network with deep sleep algorithm and without virtual spatial grid with $N_{\min} = 25$.

6 Conclusions

We have proposed an approach to integrate Zigbee and RFID networks in order to create an energy-efficient network which allows to selectively turn on and off the remote nodes. Therefore, we have proposed a deep sleep algorithm which selects the nodes to be activated according to their residual energy, so that the energy consumption of the nodes in the network is equalized. Finally, we have introduced a procedure which selects the integrated nodes to be activated according to not only their residual energy but also their spatial positions. This selection is based on the introduction of a virtual spatial grid over the network surface and *local* application of the deep sleep algorithm. For the configurations without virtual spatial grid, we have evaluated through the Opnet simulator the average residual energy performance as a function of $N_{\min}$, E_s, highlighting the energy saving guaranteed by the hybrid Zigbee–RFID network. For the configurations with virtual spatial grid, instead, we have analyzed both the area effectively monitored by the sensor network and the energy consumption. In this case, the virtual spatial grid not only provides the same area coverage than traditional Zigbee networks but also allows to extend the network lifetime. This solution, which should be applied in scenarios where local observation spatial density is relevant, could be adopted in order to create a very energy-efficient wireless sensor network based on totally passive components with addressing capabilities.

Notes

1. We remark that the energy consumption in the active periods is three orders of magnitude higher than that in the passive periods.
2. We remark that the energy consumption values considered in this model are typical of MICAz devices [8].
3. We point out that the realization of an experimental prototype physically connecting a Zigbee RFD with an RFID tag is currently under study.
4. Denoting by n_{fin} the final updating step, $E_{\text{th}}^{(n_{\text{fin}})} = E_{\text{th}}^{(0)} - n_{\text{fin}} \cdot E_s$, is too low to support communication in the Zigbee network. The value of n_{fin} depends on the specific parameters of the Zigbee network.

5. The residual energy $E_r(t)$ is the exact, not only the average, residual energy in the case of Poisson-distributed packet generation.
6. In the case of constant packet interarrival time T_{int}, $\lambda = 1/T_{int} = 4$ pck/s, whereas in the case of Poisson-distributed packet generation, $\lambda = 4$ pck/s is the *average* packet transmission rate.

References

1. R. Abileah and D. Lewis. Monitoring high-seas fisheries with long-range passive acoustic sensors. In *Proceedings of International Conference on OCEANS'96: Prospects for the 21st Century*, vol. 1, pp. 378–382, Fort Lauderdale, FL, USA, Sep. 1996.
2. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, 40:102–114, 2002.
3. IEEE 802.15.4 Std: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Computer Society Press*, pp. 1–679, Oct. 2003.
4. S. Barberis, E. Gaiani, B. Melis, and G. Romano. Performance evaluation in a large environment for the AWACS system. In *Proceedings of International Conference on Universal Personal Communications (ICUPC'98)*, vol. 1, pp. 721–725, Florence, Italy, Oct. 1998.
5. B. Bougard, F. Catthoor, D.C. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'05)*, pp. 196–201, Munich, Germany, Mar. 2005.
6. J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25(5):505–515, 1979.
7. C. Chong and S.P. Kumar. Sensor networks: Evolution, opportunities and challenges. *IEEE Press*, 98(8):1247–1256, Aug. 2003.
8. Inc. Crossbow Technology. *MICAz OEM Module 2.4 GHz*. San Jose, CA, USA, 2005. Datasheet, website: `http://www.xbow.com/Products/productdetails.aspx?sid=191`
9. R. Gallagher and D. Bertsekas. *Data Networks*. Prentice-Hall, Upper Saddle River, NJ, USA, 1992.
10. G. Ferrari, F. Cappelletti, and R. Raheli. A simple performance analysis of RFID networks with binary tree collision arbitration. *International Journal on Sensor Networks*, 4(3): 61–75, 2008.
11. G. Ferrari, P. Medagliani, S. Di Piazza, and M. Martaló. Wireless sensor networks: Performance analysis in indoor scenarios. *EURASIP Journal of Wireless Communications and Networking*, Special Issue On Mobile Man (Mobile Multi Hop Adhoc Networks) from theory to Reality, vol 2007, Article ID 81864, 14 pages, 2007. DOI: 10.1155/2007/81864.
12. K. Finkenzeller. *RFID Handbook – Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, New York, USA, 2003.
13. ISO/IEC 18000-6: Information technology radio frequency identification for item management Part 6: Parameters for air interface communications at 860 MHz to 960 MHz. *International Organization for Standardization (ISO)*, pp. 1–134, 2004.
14. O. Hyncica, P. Kacz, P. Fiedler, Z. Bradac, P. Kucera, and R. Vrba. The Zigbee experience. In *Proceedings of International Symposium on Communications, Control and Signal Processing (ISCCSP'06)*, Marrakech, Morocco, Mar. 2006.
15. IEEE 802.11 Std: Wireless LAN Medium Access Control (MAC) a Physical Layer (PHY) specifications. *IEEE Computer Society Press*, pp. 1–459, Jun. 1997.
16. P. Jurcik and A. Koubaa. The IEEE 802.15.4 OPNET Simulation Model: Reference Guide v2.0. Technical report, IPP HURRAY! Research Group, Polytechnic Institute of Porto (ISEP-IPP), Porto, Portugal, May 2007. website: `http://www.open-zb.net/research.php`

17. J.S. Lee. An experiment on performance study of IEEE 802.15.4 wireless networks. In *Proceedings of the 10th Conference on Emerging Technologies and Factory Automation (EFTA'05)*, vol. 2, pp. 451–458, Catania, Italy, 2005.
18. M. Madou. *Fundamentals of Microfabrication*. CRC Press, 1997.
19. B. Nath, F. Reynolds, and R.Want. RFID Technology and Applications. *IEEE Pervasive Computing*, 5(1):22–24, Mar. 2006.
20. L.M. Ni, Y. Liu, Y.C. Lau, and A.P. Patil. LANDMARC: Indoor location sensing using active RFID. *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference*, pp. 407–415, Mar. 2003.
21. IPP-HURRAY! Research Group Polytechnic Institute of Porto. `http://www.hurray.isep.ipp.pt:8080/opnet.html`
22. I. Ramachandran, A.K. Das, and S. Roy. Analysis of the contention access period of IEEE 802.15.4 MAC. *ACM Trans on Sensor Networks (TOSN)*, 3(1):1–29, Mar. 2007.
23. S.N. Simic and S. Sastry. Distributed environmental monitoring using random sensor networks. In *Proceedings of 2nd International Workshop on Information Processing in Sensor Networks*, pp. 582–592, Palo Alto, CA, USA, Apr. 2003.
24. J. Sung, T. Sanchez Lopez, and D. Kim. The EPC sensor network for RFID and WSN integration infrastructure. *Pervasive Computing and Communications Workshops, 2007*. (PerCom Workshops '07). *Fifth Annual IEEE International Conference*, pp. 618–621, Mar. 2007.
25. H. Vogt. Efficient object identification with passive RFID Tags. In *Proceedings of the 1st International Conference on Pervasive Computing* (PERVASIVE'03), pp. 98–113, Denver, CO, USA, Jul. 2002.
26. R. Weinstein. RFID: A Technical Overview and its Application to the Enterprise. *IT Professional*, 7(3):27–33, May 2005.
27. L. Zhang and Z. Wang. Integration of RFID into wireless sensor networks: Architectures, opportunities and challenging problems. In *Proceedings of 5th International Conference on Grid and Cooperative Computing Workshops (GCCW'06)*, pp. 463–469, Changsha, Hunan, China, Oct. 2006.
28. Zigbee Alliance Website. `http://www.zigbee.org`

Chapter 38
A Service-Oriented Wireless Sensor Network for Power Metering

A. Bagnasco, P. Buschiazzo, L. Carlino, and A.M. Scapolla

Abstract Wireless sensor networks hold the promise of innovative applications in the area of monitoring and control. This chapter presents a distributed architecture to control a grid of Wireless sensor networks. Each Wireless sensor network is managed by a gateway device that uses a web service interface to provide basic functionalities for delivering data collected by the sensors. The sensor nodes have been designed with features of low cost, easiness to use, and offer specific support to industrial applications. The chapter examines how these nodes can be used in a real world scenario, an automatic meter reading system, where there is the need to monitor energy consumption of several buildings in a wide commercial area.

Keywords Wireless sensor networks · Web services · ZigBee · Embedded systems · Energy monitoring

1 Introduction

In the last years Wireless sensor networks (WSN) have been studied by many researchers and new communication standards have been developed to fit power constraints. IEEE 802.15.4 [8] and ZigBee [18] are among these. Commercial systems and hardware devices, in particular low-cost processors, miniature sensors, and low-power radio modules are becoming available. Many companies, like Texas Instruments, Atmel, and Ember have developed Systems-on-Chip (SoC) that reduce the cost and the power consumption of the network nodes, whose batteries should last several years. They are standard compliant and the interoperability of products by different vendors is guaranteed [15].

In a typical situation, a WSN is composed of a large number of sensor nodes which are deployed in a region where there is the need to monitor a physical phenomenon. If the region is large, inaccessible, or dangerous, this deployment could be

A. Bagnasco (✉)
DIBE – University of Genoa, 16145 Genova, Italy
e-mail: andrea.bagnasco@unige.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_38,

random and the WSN nodes have to self-organize and to collect data automatically. Since the networks can be composed of hundreds or thousands of nodes, many algorithms and protocols [10, 17], available in the literature, target network issues, like distributed routing, congestion control, and reduced power consumption. WSNs for environmental monitoring and process estimation are investigated in [5], with attention to the trade-off between quality of service in process estimation and network life-time.

Potentially WSNs support a broad spectrum of monitoring applications like habitat monitoring, object tracking, military surveillance, industrial plant controlling, fire detection, and many others. The absence of physical cables reduces the impact of the systems and their installation costs.

One of the emerging application fields is automatic energy meter reading. As the European, Asian, and North American electricity grids come under increasing demand, the focus on curtailing peak load asks to building owners and building automation stakeholders to participate in load shedding strategies. Besides this, electricity suppliers want to improve their services and enhance internal operation management by means of a real-time energy consumption indication from the meters. This chapter presents the service oriented implementation of a WSN platform for monitoring power meters. The next sections describe the architecture of an automatic meter reading (AMR) system for power metering and detail the hardware structure of a single node of the WSN.

2 System Architecture

A WSN is typically composed of a set of low-power sensing nodes, deployed in the environment and managed by a more powerful node, called base-station or gateway. The gateway brings the acquired data to the upper level of the communication infrastructure, so it is often equipped with GSM/GPRS/UMTS, Ethernet or Wi-Fi connectivity.

In the real world the number of WSNs for data acquisition, and thus the number of the gateways, tends to be large. Moreover, we have to deal with different kinds of gateways, provided by different vendors or integrators. It is important to find a standard and flexible way of communication to collect data from these gateways.

This chapter proposes the adoption of a Service Oriented Architecture (SOA) [9], a loosely coupled model, where clients ask services to the gateway servers, without a specific knowledge about them. The service interfaces, described by WSDL (Web Service Description Language), hide the specific hardware and software architecture used in the WSN.

The implementation of web services on limited-resources devices is not trivial [6, 12], but it is feasible at gateway level because these devices, the WSN access points, have more resources, in term of energy, computational power, and storage, than the simple sensor nodes.

We have studied a two-layer architecture, where the lower layer provides basic functionalities for the exchange of data between the gateway and the sensor nodes

using the ZigBee protocol and the upper layer aggregates and export data via the web service technology (*see figure 38.1*). From the WSN point of view the gateway is the network coordinator, while from the application point of view, it is a data service provider. This approach allows aggregating many WSNs and to build grid-based infrastructures [4].

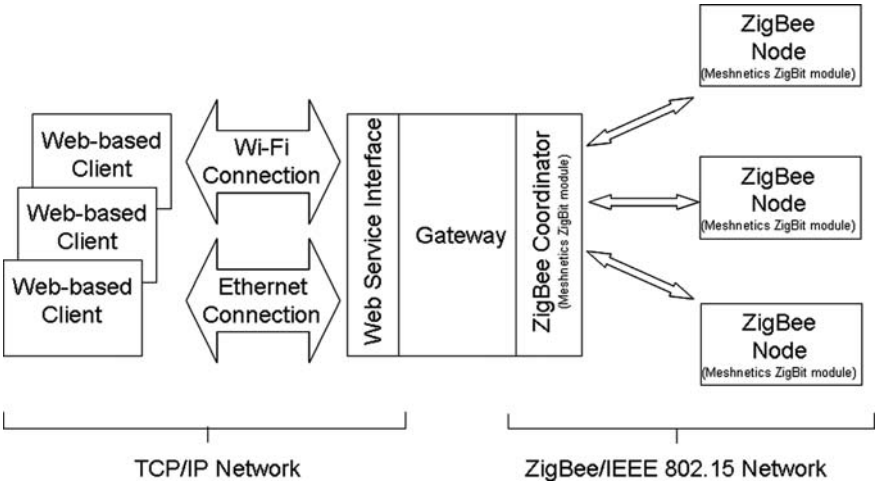


Fig. 38.1 The system architecture

3 The Gateway

The gateway collects and stores the information acquired by the WSN nodes in a local memory, delivers data to the upper control layer, and forwards configuration commands to the end nodes. This multi-threaded job requires a complete operative system. The automatic meter reading system, presented in this chapter, uses gateways that are implemented on the Fox Board [1] platform. It is a ready to use Linux embedded board, based on an Axis ETRAX 100LX microprocessor [3]. The ETRAX 100LX processor has a 100 MIPS 32 bit RISC architecture with a 32 MB RAM and 8 MB of flash memory. The Fox Board offers several types of communication channels: up to four serial ports, two USB 1.1 interfaces, and a 10/100 Ethernet interface. Fox Board also supports MMC/SD memory cards to increase the persistent storage.

Despite the original GNU/Linux distribution of Fox Board has a native web server, the adoption of the more sophisticated foXServe distribution was preferred. It contains an embedded version of the Apache Web Server and uses the SQLite library for high-level data storage methods. The SQLite library simplifies the implementation of the database of the data acquired from the WSN, and the reading procedures running in the Fox Board can use common SQL queries.

The gateway manages all the nodes of the network through a radio module connected to an RS-232 port of the Fox Board. It stores two kinds of information: data acquired from sensors and events, like alarms. The WSN sends three kinds of events to the gateway: local events, simple notifications, and critical notifications. Local events are typically associated to well-known occurrences, so they are directly managed by the gateway. Notifications are forwarded to the server. The critical ones usually represent alarms that need human intervention, so it is necessary to notify them immediately.

All the web services deployed on the gateway have been realized using the gSOAP toolkit [16]. The web services realized with gSOAP are compact and fast, so they can be deployed in embedded systems such as the Fox Board. gSOAP allows an easy development of web services by C/C++ language and provides a transparent SOAP Application Program Interface (API) that maps C/C++ native data types to XML schemas. As an example, *GatewayService*, the main web service interface deployed on the gateway, provides several methods to communicate with the sensor nodes using XML encoded data: the *SendCommand()* method allows client applications to send commands to sensor nodes; the *SetValue()* and *GetValue()* methods can be used to send and retrieve data from the nodes; finally, the *GetMetadata()* method can be used to retrieve additional information on the sensor node. All the commands, data, and metadata used by the gateway are associated with a specific XML schema and are automatically managed by the gSOAP toolkit.

4 The WSN Node

Engineers have designed many sensor nodes (commonly called "motes") for WSNs; some relevant examples are Mica [7], MicaZ, Telos [13] from University of Berkeley, Imote, developed by Intel Research and EyesIFX by Infineon. These nodes and some other sensor boards with limited capabilities are often used as prototypes for research activities. They are good development platforms for the research community. There are many studies on new network protocols which employ these devices, but they are seldom used in real-world applications, because they are expensive and sometimes use proprietary communication protocols, making difficult to achieve a complete interoperability between different devices. These considerations suggested designing a WSN node which could be totally customized to the application under development. The System-on-Chip approach has been adopted instead of conceiving a specific radio module and a separate computational unit, because it is more cost- and size effective. During the preliminary stage of the design, different commercial products have been considered: Telegesis ETRX2, Xbee series 2, and Meshnetics ZigBit modules [11]. The final choice was ZigBit, since it is totally ZigBee compliant, while ETRX2 and Xbee series 2 use modified packet structures. ZigBit can be used as both a separate IEEE 802.15.4 radio modem and a complete system-on-chip. The ZigBit chip includes an ATmega1281 microcontroller and an AT86RF230 [2] radio from Atmel. The microcontroller has 8 kB RAM, 4 kB EEPROM, and 128 kB of program memory to host both the ZigBee protocol stack and the user

program. The entire protocol stack is built on the TinyOS operating system and Meshnetics provides API functions to manage the reception and the transmission of data through the network. This way, application developers can build ZigBee-based wireless solutions, dedicated to their applications, without a deep knowledge of TinyOS and the nesC language. The same approach has been used, for example, by Texas Instruments with the release of API in the Z-Stack and by Ember in the EmberZNet ZigBee protocol stack.

The architecture of the node consists of a central unit which contains the Meshnetics ZigBit module, a Flash data memory, a switching regulator, and some other components; and one or more external sensor boards to acquire and condition analog signals and sensor data. This modular design guarantees a general purpose system. It allows to select the hardware required by the target application, avoiding unnecessary sensors or components. This way the power consumption and the dimension of the node are reduced at the minimum.

Since industrial applications often use serial communication (like the Modbus protocol) and differential form of signaling (like RS422/RS485) we have developed a specific hardware module to manage these communications. In this board there is a low-power half-duplex RS422/RS485 transceiver that, instead of using termination resistors, uses a Schottky-diode termination. Unlike other termination types, Schottky diodes do not attempt to match the line impedance, but they simply clamp the overshoots and undershoots caused by reflections. Voltage excursions are limited to the positive rail plus a Schottky diode drop in one direction and to ground less the same voltage drop in the other direction. Schottky-diode terminations waste little power, because they conduct only in presence of overshoots and undershoots. On the other hand, standard resistor terminations, with or without failsafe bias resistors, draw power continuously.

5 Application Scenario

The WSN node, described in the previous section, has been designed and tested for the application of power monitoring at the Porto Antico area of Genova. This is a wide commercial and recreational area planned by Renzo Piano for the Columbus Exhibition in 1992 in Genova (Italy). Power load and energy consumption of the utilities in the area must be monitored through a number of energy meters, located in different places. A control room in the headquarter building supervises all these devices.

The area is very large (130,000 m^2) and there are many sparse buildings. WSNs fit well communication needs, in terms of distance and reliability. Furthermore, since power meters are already on, wireless nodes avoid the installation of additional cables or electrical equipments inside the cabins.

The nodes connected to the wall sockets work as routers, which manage the other battery powered nodes inside the room. The tiered multi-hop structure of ZigBee-based networks is well suited for this scenario. The network layer of the ZigBee protocol stack uses the Ad-hoc On-demand Distance Vector (AODV) routing protocol

[14], which enables a multi-hop data transmission across the network; so data packets can cross different routers to reach the control room. Tests have been done on the collision avoidance of IEEE 802.15.4, to be sure that nodes could communicate even in presence of Wi-Fi access points or other electromagnetic interferences in the area.

In the cabins there are two kinds of electricity meters: the Circutor CVM-BD-RED-C420 and the Vemer Energy-400 PWR. The former is used inside the high power cabins and it is accessible through a RS485 interface by using Modbus, which is a commonly used industrial communication protocol. The latter is a simpler energy meter that has an opto-isolated line, which provides a pulse every unit of KWh consumed.

The sensor devices of the CVM meters are equipped with an additional board to convert UART signals into RS485 differential signal levels. By using the Modbus protocol we can access also internal registers of the meter that gives information about the peak load and the active/reactive power consumption. This information is useful for a real-time analysis of the system. To reduce the power consumption of the nodes, a periodic polling of the CVM meters is scheduled and the nodes stay in sleep mode for the rest of the time.

The sensor devices, which monitor Vemer Energy-400 m, use the interrupt lines of the microcontroller to wake them up when a pulse is received. The devices are in sleep mode most of the time to save their energy and periodically send information using the CSMA/CA technique defined in the IEEE 802.15.4 standard.

Each cabin is equipped with a WSN that acquires data from the meters. The gateway of each WSN exposes services that can be used by end users to collect data from power stations.

To preserve battery life and to improve reliability, every node remains in a sleep state for most of the time and wakes up periodically to send data. So, if the gateway does not receive any packet from a specific node for a predefined amount of time, it alerts the management staff that there is a problem in the network, and there is the need to verify if a node or a communication link is broken.

Every WSN node saves data in its internal EEPROM memory, and if the radio transmission to the coordinator is successful (this means that the remote node receives an acknowledge packet), the EEPROM location is cleared. Instead, if the transmission is unsuccessful, the node goes on incrementing the value of the measured pulses, until the communication link is restored. By using this approach, it is guaranteed that the system does not lose any pulses from the energy meters.

Since every nodes wakes up periodically to send data to the gateway, there is the need to define its optimum duty cycle, to preserve battery life, and to guarantee a good resolution of the system.

The current consumption of the node is 20 mA during transmission and reception of data (at 0 dBm output transmission power), and it is reduced to about 10–15 mA when the radio is turned off and the MCU is active (depending on multiple factors). When the node is in sleep state, the current consumption is less than 50 μA. For example, if WSN nodes wake up for 1 s and sleep for 30 s, with two standard AA alkaline batteries (the capacity of a single AA battery is 2,500 mAh), every node

consumes about 19.4442 mA every 31 s. This means that, depending on the number of pulses received, the batteries will last for about 332 days in optimal working conditions (few loss packets, 0 dBm transmission power, good communication link, ideal batteries, etc.).

An improvement could be obtained if the node wakes up only for half a second (which is a sufficient time to send data to the coordinator); in this situation the batteries life is almost doubled. The following graph shows the estimated battery life of a single node, depending on its active and sleep periods.

As you can see from the graph of figure 38.2, if the active period is long, then increasing its sleep period does not extend battery life. Instead, when the active period is short, controlling the sleep period duration gives an improvement, even of 100 days, on the life of the node.

According to this, the energy monitoring system can use an adaptive strategy and change the sleep period of every node, depending on the energy consumption of the meters connected to them. So, if the electricity consumption of a utility is little, the system can change the duty cycle of the node, to save batteries. This adaptive control can be done from the management station, that can produce statistics of the current consumption of the different utilities, and then, depending on this analysis, it can send messages to the nodes to change their duty cycle.

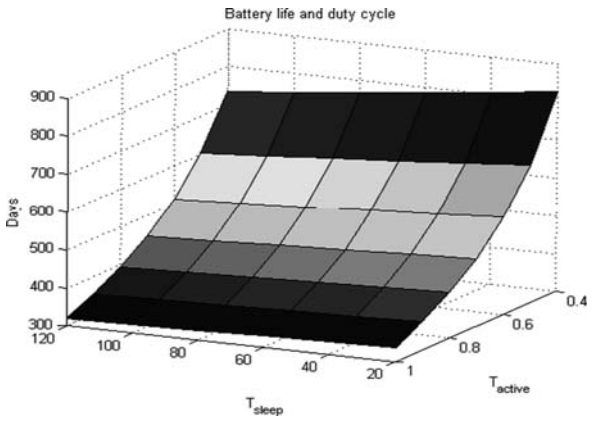


Fig. 38.2 Estimated battery life

6 Conclusions

In this chapter we have presented a Wireless sensor network application for automatic meter reading that has been developed for monitoring the energy consumption and the peak loads of many different buildings in a wide commercial area.

The proposed architecture allows controlling WSN devices using an embedded web server that provides a web service interface as well as common services on the Internet.

Acknowledgments The project was funded by the "Parco Scientifico e Tecnologico della Regione Liguria."

The author would like to thank Giorgio Allasia and Giovanni Giannotta for the technical contributions.

References

1. ACME Systems, Rome, Italy. *Fox Board, a Linux Core Engine in just 66 × 72 mmv*.
2. Atmel Corporation, San Jose, CA. *AT86RF230 ZigBee/IEEE 802.15.4–Transceiver Datasheet*.
3. Axis Communications, Sweden. *Axis ETRAX 100LX Datasheet*.
4. A. Bagnasco, D. Cipolla, A. Poggi, and A.M. Scapolla. Service-Oriented Architectures for distributed cooperative instrumentation grids. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Grid Enabled Remote Instrumentation*, pp. 227–235, New York, 2009. Springer US.
5. D. Dardari, A. Conti, C. Buratti, and R. Verdone. Mathematical evaluation of environmental monitoring estimation error through energy-efficient wireless sensor networks. *IEEE Transactions on Mobile Computing*, 6(7):790–802, 2007.
6. R. Glaschick, B. Oesterdieckhoff, and C. Loeser. Service oriented interface design for embedded devices. In *Proceeding of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2005.
7. J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22:12–24, Nov. 2002.
8. Institute of Electrical and Electronics Engineers. *IEEE 802.15–2006 standard for Low Rate Wireless Personal Area Networks*, 2006.
9. F. Jammes and H. Smit. Service-oriented paradigms in industrial automation in Industrial Informatics, *IEEE Transactions on Industrial Informatics*, 1(1):62–70, Feb. 2005.
10. H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley Interscience, John Wiley and Sons Inc., 2005.
11. Meshnetics Corporation, Phoenix, Az. *ZigBit OEM Module Product Datasheet* 2007.
12. B. Oesterdieckhoff, C. Loeser, I. Jahnich, and R. Glaschick. Integrative approach of Web Services and universal plug and play within an AV scenario. In *Proceedings of the 3rd International IEEE Conference on Industrial Informatics (INDIN)*, 2005.
13. J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Proceedings of the 4th Int. Symposium on Information Processing in Sensor Networks (IPSN)*, Los Angeles, California, 2005.
14. *Ad–hoc On–demand Distance Vector*, rfc–3561 edition 14 July 2003.
15. K. Tuan. *Zigbee System-on-Chip (SoC) Design*. High Frequency Electronics, Jan. 2006.
16. R.A. van Engelen and K.A. Gallivan. The gSOAP toolkit for web services and peer-to-peer computing networks. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.
17. R. Verdone, D. Dardari, G. Mazzini, and A. Conti. *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*. Elsevier, 2007.
18. ZigBee Alliance. *ZigBee Specifications*, Dec. 2006.

Chapter 39
Performance Evaluation of a Robust Data Aggregation Approach in Diverse Sensor Networking Environments

S. Kafetzoglou, M. Grammatikou, and S. Papavassiliou

Abstract In wireless sensor networks, a sensor node, besides being able to act on the environment, is also a network entity that performs networking-related functionalities and could be integrated within the networking and grid environment. In this chapter, we adopt a probabilistic approach that presents a robust and generalized framework for performing in-network processing in order to improve the overall data collection and monitoring process in a sensor network. The effectiveness and applicability of the framework under investigation in a diverse set of networking environments that present different constraints, limitations, and physical characteristics are demonstrated via modeling and simulation, under different traffic loads. Furthermore, different ways of dynamically varying and configuring parameters related to the aggregation process are investigated and evaluated for different applications and networking environments (terrestrial and underwater sensor networks).

Keywords Sensor networks · QoS · Data aggregation · In-network processing

1 Introduction

The evolutionary approach of the "Internet of things" describes a worldwide network of intercommunicating devices, capable of supporting the public good and leading to economic growth and personal enrichment of life. The vision of a world with a large number of networked and interconnected devices may be realized by embedding short-range mobile transceivers into a wide array of additional gadgets and everyday items, enabling new forms of communication between people and things, and between things themselves. Within such an environment, data collection will benefit from the ability to detect changes in the physical status of things, using sensor technologies. In this regard, sensors play a pivotal role in bridging the gap between the physical and virtual worlds and enabling things to respond to changes in

S. Kafetzoglou (✉)
Network Management & Optimal Design Lab (NETMODE), National Technical University of Athens, Iroon Polytechniou 9, 15780 Zografou, Greece
e-mail: skafetzo@netmode.ntua.gr

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_39,

their physical environment. Sensors collect data from their environment, generating information and raising awareness about context.

Among the key aspects of the Internet of things is the integration of heterogeneous sensors in a distributed system that performs actions on the physical world based on the environmental information gathered. Key challenges are the heterogeneity (in terms of hardware and software systems) and coordination aspects (how sensors coordinate in time and space). From the perspective to be considered in this chapter, a sensor node, besides being able to act on the environment, is also a network entity that performs networking-related functionalities and could be integrated within the networking and grid environment.

Therefore, we envision wireless sensor networks (WSNs) consisting of several dozens, hundreds, or even thousands of wireless sensors which have sensing, processing, and communication capabilities. A sensor node monitors the targeted phenomena, such as temperature, humidity, light, vibration, and sound depending on different types of applications, in its sensing range and reports the results, which can be raw sensing values or processed data, to its data sink in a single or multi-hop manner.

Compared to traditional networks, WSNs face additional challenges including network lifetime considerations, wide range of network densities, fault tolerance, and scalability issues that originate from the underlying physical and network characteristics.

1.1 Related Work

Wireless sensor networks are envisaged as the technology of the future and the present, for different applications such as smart houses, military purposes, earth observation, and medical appliances [5]. Therefore, there are a vast amount of papers in the literature concentrating on different aspects of wireless sensor networks, such as data gathering, sensor deployment, and configuration, most of which aim at providing frameworks that take into consideration the special limitations of these environments.

One of the most important characteristics of WSNs is that every sensor node is operated by built-in battery which is hard to be recharged or replaced. Therefore, reducing unnecessary energy consumption is a very important challenge. Energy conservation in such networks involves two dominant approaches to minimize communication overhead. The first one is at the MAC (Medium Access Control) layer where nodes turn off their radios when they are not required for multi-hop communication. The second one refers to data reduction through in-network processing (also called data aggregation), whereby correlations in data are exploited to reduce the size of data and correspondingly communication cost. Since local computation is much cheaper than radio communication, supporting some in-network processing to reduce data within the network can provide significant energy savings.

Specifically, in [2] sensor nodes are partitioned into clusters and the cluster heads maintain a TDMA (Time division multiple access) schedule to exchange data

between cluster members and cluster heads with no peer-to-peer communication. In order to communicate between clusters, a CDMA (Code division multiple access) mechanism, which, however, has been known as an expensive mechanism in WSNs, is used. Furthermore, S-MAC [8] expects sensor networks to be deployed in an ad hoc fashion, with individual nodes remaining largely inactive for long periods of time, but then becoming suddenly active when something is detected. Energy conservation and self-configuration are primary goals of S-MAC, while per-node fairness and latency are less important. S-MAC reduces idle listening energy waste by putting sensor nodes to sleep periodically. Neighboring nodes form virtual clusters to auto synchronize on sleep schedules.

On the other hand, the authors in [1] propose a heuristic algorithm that performs in-network aggregation and succeeds in extending the network lifetime. The authors perceive the issue of data gathering as an optimization problem. Specifically, given the location of the sensors and the base station as well as the energy of each sensor, they aim at finding an energy-efficient manner, in which the data are collected, aggregated, and transmitted to the base station, such that the network lifetime is maximized. In [6] the PEDAP data gathering and aggregation protocols are presented that compute a minimum spanning tree over the sensor network using Prim's algorithm and perform data aggregation as the data are transmitted to the base station, which is the root of the tree.

1.2 Motivation and Objective

In our work we follow the philosophy of using in-network processing (i.e., data aggregation), whereby correlations in data are exploited to reduce the size of data and correspondingly communication cost and overall network traffic. Specifically, motivated by the data centric nature of the communication in sensor networks, a distributed and probabilistic framework for data gathering is adopted [9, 4], which can be ideally applied to resource-constrained sensor networks. Sensor measurements are aggregated on the fly at intermediate sensor nodes on their way to the collection center, by taking into account two main characteristics: the lack of fixed infrastructure and the fact that the data are destined toward a specific destination (i.e., collection center) in these environments – to the contrary of other conventional wireless networks where a large number of different destinations may exist. Then, a distributed and probabilistic method is applied by the various sensor nodes to determine whether or not to perform aggregation. A typical sensor network topology is presented in Fig. 39.1. In this chapter, the emphasis is placed on studying the trade-offs and determining the appropriate values for specific configuration parameters of the overall end-to-end data aggregation process. It should be noted that the used approach aims at utilizing the available limited resources efficiently, effectively reducing significantly the network traffic, and as a result shortening the queuing delays at the intermediate nodes, reducing the corresponding collisions and minimizing the energy wastage in data transmission. Therefore it can extend the

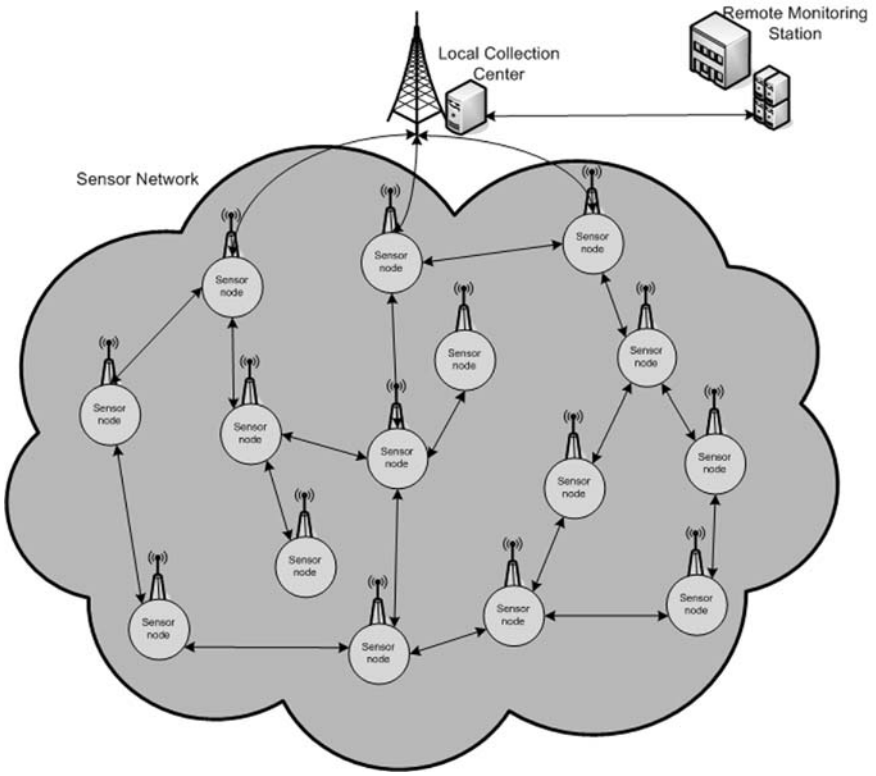


Fig. 39.1 A typical wireless sensor network

sensor network lifetime while at the same time the task QoS (Quality of Service) requirements (e.g., delay constraints) are taken into account.

Under this angle, the performance gains that can be achieved by the utilized data aggregation framework are evaluated via modeling and simulation under different aggregation scenarios and traffic loads, while the impact of several configuration parameters on the overall system performance is quantified. To demonstrate the robustness and applicability of the examined framework in diverse physical environments, two different environments are considered, namely, free space and underwater, each of which poses diverse constraints and design principles.

2 Framework Overview

The framework adopted here consists of a two-phase algorithm [9, 4]. First, a communication/routing tree rooted at the base station and spanning the whole network is constructed based on bridging techniques and the data link layer is used to forward

packets in the network. The tree assigns levels to the nodes of the network signifying their distance to the root and is used later for the data gathering procedure. During the second phase, the sensors transmit their collected data to the base station in a multi-hop manner, sending their readings to their father with respect to the data gathering tree. Each node is responsible not only to send its own data but also to forward the ones received by its children. At that point, it can decide whether or not to perform data aggregation, resulting in the transmission of less data, based on a probabilistic method. Moreover, in order to address time issues related to some time-critical applications a delay constraint D is considered, with respect to the time (delay) in which a packet received by the base station is considered valid. Consequently, each sensor when it generates and/or receives a packet performs one of the following actions:

1. If the delay constraint provided by the application can be met, with probability γ the node waits (i.e., packet transmission is deferred) for a certain time interval τ for other packets to arrive and aggregate the data received forming one single packet. When the time interval expires it forwards the data to the next sensor node where the same procedure is followed until the data reaches the sink (collection center). With probability $(1 - \gamma)$ the node simply forwards the data.
2. The node determines if the delay constraint can be met only if the packet is not deferred, and forwards it to the next hop.
3. The node determines if the delay constraint cannot be satisfied in any case, and discards the packet.

If a packet of collected data are delivered to the collection center within the given constraint D, it is considered to be a successful delivery. Otherwise, the data are obsolete and therefore not needed. It is noted here that parameters γ and τ are configurable parameters of the network and their values can be used to trade off several performance parameters such as successful delivery ratio, energy consumption, end-to-end delay. In order to realize such a data aggregation approach the sensor nodes have the ability to determine whether their data can be delivered to the collection center by making use of their level attribute obtained throughout the tree construction phase. More specifically the estimated time for a data packet to reach its final destination can be calculated as follows: $t_{\text{estimated}} = \left(\frac{1}{\text{bw}} t_{\text{pr}}\right) \cdot \text{level}$ where l is the total length of a data packet in bits, bw denotes the bandwidth of the transmitting media in bits/s, level identifies the level of the sensor node under consideration and represents the average per hop propagation delay.

3 Diverse Application Environments

As explained before, one of the main objectives of this chapter is to examine and evaluate how the considered generic framework behaves under different environmental constraints and for different values of the application parameters, such as different deferred period τ and propagation delay. In particular, two environments

are considered, the free space and the underwater environment. Terrestrial sensor networks can be used for a variety of applications, such as target monitoring [7] where timely delivery of data is of utmost importance, while underwater acoustic networks mostly focus on long-term monitoring of selected ocean areas for data collection, pollution monitoring, and disaster prevention. Therefore, efficient and reliable data communication protocols are required for maximizing the amount of data received by the base station.

One of the main differences between terrestrial and underwater sensor networks is that the wireless links of the underwater networks are based on acoustic communications and are mainly governed by three factors: limited bandwidth, time-varying multipath propagation, and low speed of sound underwater (1.5×10^3 m/s), instead of the RF used by the terrestrial sensor networks, where the signal travels approximately with the speed of light (3×10^8 m/s). Therefore, while propagation delay is negligible for short-range RF, it is a critical factor in underwater communications. As a result, timely delivery of data in an energy-efficient manner is important in time-critical terrestrial sensor networks, while reliable transmission of data and reduction or even elimination of wasteful transmissions is of high importance in underwater sensor networks due to the occurrence of large propagation delays as a result of the use of low-frequency acoustic communications.

4 Performance Evaluation and Discussion

As stated before, in-network processing, i.e., network aggregation, has been introduced as a means to transmit less data and consequently extend the network lifetime. In this section the effect of the data aggregation deferred period τ is studied in both underwater and terrestrial networks, in terms of probability of success and energy consumption.

4.1 Models and Scenarios

Specifically, we consider a wireless sensor network consisting of 48 nodes aligned in a grid topology placed in a $200 \times 200\,\text{m}^2$ grid. The sensor nodes are assumed to generate data following a poisson process with rate λ packets/s and send their readings to the collection center using the data gathering tree developed by the leveling algorithm. The transmission range of each node is assumed approximately 50 m, while when a node begins to transmit, all neighbors within its transmission range will receive the signal, which is considered as interference for a node if the packet is not destined for it.

Furthermore, the data transmission rate is assumed to be 0.5 Mbps and the signal propagation speed is 3×10^8 m/s for the terrestrial applications and 1.5×10^3 m/s in the underwater sensor networks. In order to comprehensively reflect the data aggregation gains we have used an aggregation coefficient, namely β, which essentially

represents the ratio of the length of the aggregated packet to the total length of all the received packets before aggregation occurs.

For demonstration purposes only, throughout our simulation experiments coefficient β is set equal to 0.8. Each sensor is presumed to consume energy whenever it performs sensing, transmitting, and receiving operations. In the case of sensing a fixed amount of energy is depleted, whereas for transmitting and receiving data the energy is consumed according to the first-order model presented in [3]. The simulation for each scenario lasts for 2000 s while each simulation scenario was repeated 10 independent times and statistical averages are calculated. Finally, we consider that whenever possible each sensor node performs data aggregation, i.e., $\gamma = 1$.

All the experiments were run for four different dynamic ways of varying the value of deferred period τ for each node, based on the corresponding level that each node belongs to. In the first case the deferred aggregation period is increased as the data flows from the leaves, i.e., the lower levels of the tree, toward the root, i.e., the upper layer in the tree (we refer to this case as "up" case). In the second case we enforce exactly the opposite policy, that is, the lower level sensor nodes have longer deferred period while the nodes close to the base station use smaller values (this is referred as the "down" case). We also investigated two mixed cases where in the first one the nodes near the leaves have small values for the deferred period τ while as we move up toward the middle point of the data gathering tree the corresponding waiting period reaches its maximum and then decreases again as we approach the root ("up–down" case). In the second mixed case, the opposite trend is observed, that is, large waiting periods are assumed for aggregation at the leaves and the root, while shorter values (decreasing in a similar way from the two edges of the tree) are considered at the intermediate nodes ("down–up" case). Finally, mainly for comparison purposes two additional cases are presented. In the first one a fixed deferred period τ for each node is considered independent of its level ("fixed" case) while in the last one no aggregation is performed at all ("no aggregation"). For each experiment considered the probability of successful data delivery and the corresponding energy consumption under the different cases are presented as functions of the traffic load.

4.2 Numerical Results

Terrestrial sensor networks: In Fig. 39.2 the successful delivery probability (P_{succ}), calculated as the ratio of the packets received by the collection center within the delay constraint D to the total amount of data packets generated, is obtained and evaluated as a function of the network traffic, i.e., the packets generated by each node. It is observed from Fig. 39.2 that the probability of a successful delivery decreases as the network traffic increases. For heavy traffic the up method performs better while for lighter traffic the up-down method outperforms all of them. Moreover, the case where no aggregation is performed shows the worst behavior as expected, since the network is flooded with data packets leading to greater losses and thus smaller probability of success.

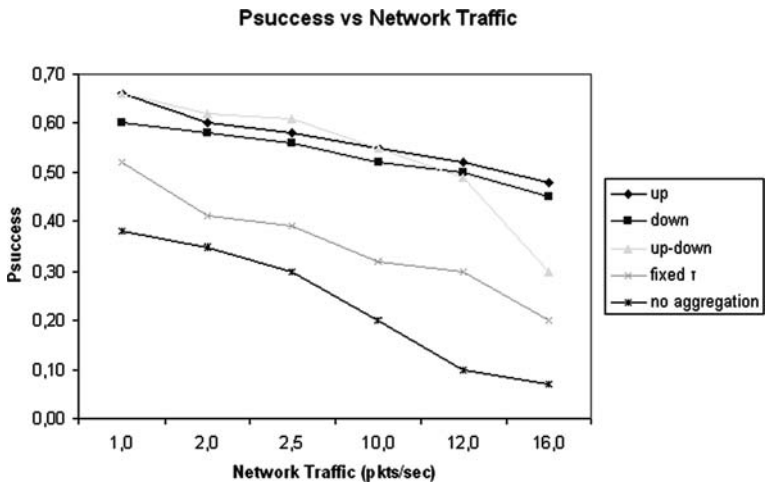


Fig. 39.2 Successful delivery probability (P_{succ}) vs. traffic load in terrestrial networks

However, as observed from Fig. 39.3 the energy consumed in the up-down case is less than all others making it suitable for long-term sensor applications. It is also noted that the case where no data aggregation is performed (no aggregation) consumes the largest amount of energy, followed by the case where all the levels have the same deferred period (fixed case).

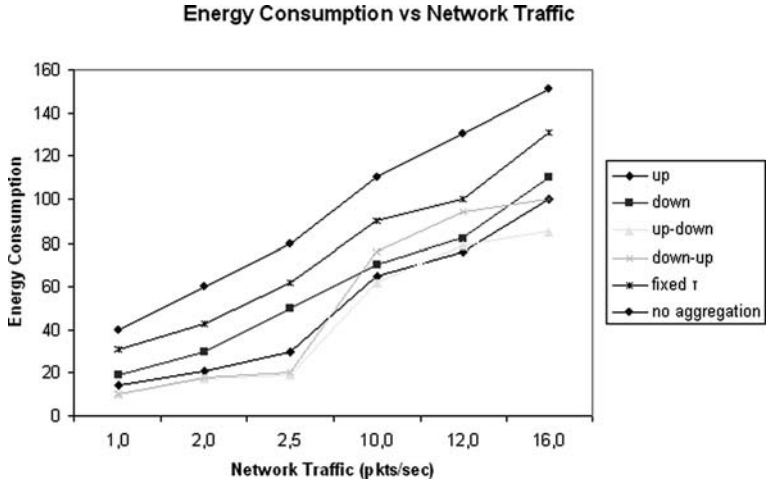


Fig. 39.3 Energy consumption vs. traffic load in terrestrial networks

Underwater sensor networks: The same set of experiments was conducted also for the underwater case study. Again it is shown in Fig. 39.4 that the less the traffic in the network the higher the probability of successful delivery. Moreover, the up-down

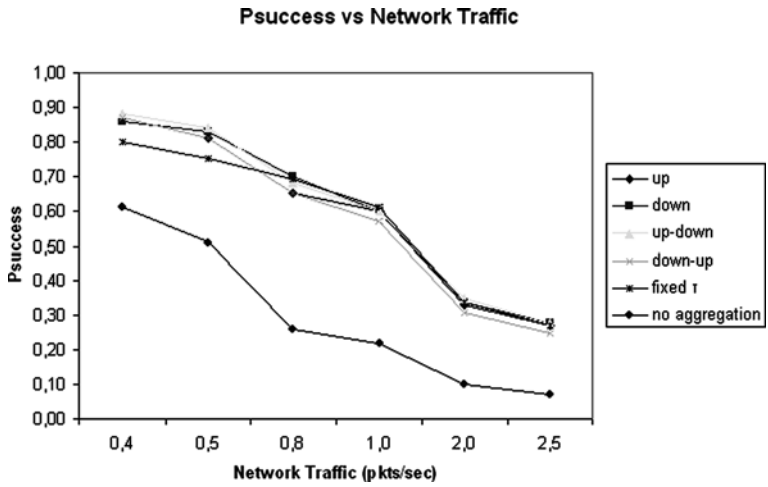


Fig. 39.4 Successful delivery probability (P_{succ}) vs. traffic load in underwater sensor networks

method seems to outperform the others, while all four approaches where dynamic values are selected for the deferred period τ outperform both the case in which the value of τ is the same at every level of the tree and the case where no aggregation is performed. Specifically, in the case where there is no in-network processing the probability of success is very low even for very light network traffic, making it not suitable for use in this kind of resource constrained networks.

In terms of energy consumption, as it can be seen from Fig. 39.5, in general the higher the network traffic the higher the overall energy consumption. Again, the up-down method consumes the least energy among all the approaches, while the no

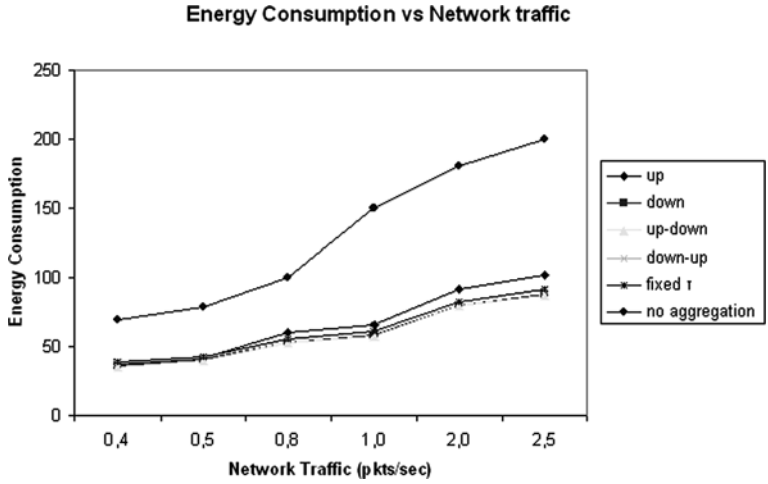


Fig. 39.5 Energy consumption vs. traffic load in underwater sensor networks

aggregation case consumes a very large amount of energy leading to quick depletion of the total network energy and therefore decreasing significantly the sensor network lifetime.

5 Conclusions

In this chapter, a distributed probabilistic and robust data framework has been utilized in order to study the trade-offs and benefits of an overall end-to-end data aggregation process in resource-constrained sensor networks and provide some insight regarding the critical parameters that may affect the corresponding performance in terms of successful data delivery and energy consumption. To demonstrate the robustness and generality of the considered data aggregation framework two different application environments were examined, namely the terrestrial and the underwater sensor networks, each of which poses diverse constraints and design principles.

The performance evaluation was carried out via modeling and simulation for different aggregation approaches regarding the configuration of critical parameters (both dynamic and fixed modes) and under different traffic loads for networking environments that simulate the physical characteristics of both terrestrial and underwater sensor networks.

Our current and future work emphasizes on further optimizing the operation of resource-constrained sensor networks, by considering MAC layer enhancements with the use of sleeping periods, and incorporating them to our distributed, probabilistic framework in order to further reduce the energy consumption.

References

1. K. Dasgupta, K. Kalpakis, and P. Namjoshi. An efficient clustering-based heuristic for data gathering and aggregation in sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking*, Mar. 2003.
2. W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, 2002.
3. W.R. Heinzelman, A. Chandrakasan., and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the IEEE Hawaii International Conference on System Sciences*, Jan. 2000.
4. S. Kafetzoglou, M. Alexandropoulou, and S. Papavassiliou. A novel data gathering framework for resource-constrained underwater sensor networks. *Ad Hoc & Sensor Wireless Networks*, 5(3–4):313–329, 2008.
5. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, Sep. 2002.
6. H.Ö. Tan and I. Körpeo. Power efficient data gathering and aggregation in wireless sensor networks. *ACM SIGMOD Record*, 32(4):66–71, 2003.
7. The Sequel Project. http://ralyx.inria.fr/2007/raweb/sequel/uid0.html

8. W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.
9. J. Zhu, S. Papavassiliou, and J. Yang. Adaptive localized QoS-constrained data aggregation and processing in distributed sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(9):923–933, Sep. 2006.

Author Index

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5,

Subject Index