Ramin Sadre   Aiko Pras (Eds.)

Scalability of Networks and Services

Third International Conference on Autonomous Infrastructure, Management and Security, AIMS 2009
Enschede, The Netherlands, June 30–July 2, 2009
Proceedings

Springer

Volume Editors

Ramin Sadre
Aiko Pras
University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands
E-mail: {r.sadre, a.pras}@utwente.nl

Library of Congress Control Number: 2009929558

CR Subject Classification (1998): C.2, D.4.4, H.3, H.4

LNCS Sublibrary: SL 5 – Computer Communication Networks
and Telecommunications

ISSN 0302-9743
ISBN-10 3-642-02626-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02626-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12707539 06/3180 5 4 3 2 1 0

Preface

This volume of the *Lecture Notes in Computer Science* series contains the papers accepted for presentation at the Third International Conference on Autonomous Infrastructure, Management and Security (AIMS 2009). The conference took place in Enschede, The Netherlands, hosted by the University of Twente. AIMS 2009 was organized and supported by the EC IST-EMANICS Network of Excellence (#26854) and co-sponsored by IFIP WG 6.6 and the Strategic Research Orientation of the University of Twente on Dependable Systems and Networks (DSN).

AIMS 2009 constituted the Third edition of a single-track and standalone conference on management and security aspects of distributed and autonomous systems, which took place initially in Oslo, Norway in June 2007, followed by AIMS 2008 in Bremen, Germany.

The theme of the AIMS 2009 conference was "Scalability of Networks and Services," focusing on how scalable networked systems can be monitored, managed, and protected in an efficient and autonomous way. The research papers that have been selected for publication in the present proceedings have approached this theme from different perspectives, covering topics such as network resource management, overlays and peer-to-peer networks, network configuration and optimization, and monitoring and visualization.

This diversity has helped AIMS 2009 once again to reach its main objective: to be a cross-disciplinary conference where researchers can exchange their results with sufficient time for discussion of novel ideas. For the main conference program, we received a total number of 28 paper submissions. The submissions came, according to the affiliation of the first author, from 14 different countries. Seven papers came from non-European countries, the same number were submitted from non-academic organizations.

The second objective of the AIMS conference series is to provide a venue for doctoral students to present and discuss their research ideas in a wider audience. In this way, students obtain feedback about their results achieved so far, as well as about their future research plans. This idea has been implemented in the form of a PhD workshop. In total, the workshop received 15 PhD paper submissions in which PhD students gave a description of their research problem and the chosen approach, and outlined the results achieved. In addition to the main conference program and the PhD workshop, AIMS 2009 also included a tutorial program (not part of the proceedings).

All submitted conference and PhD workshop papers were reviewed by three members of the Technical Program Committee (TPC) (and additional reviewers) and the PhD Workshop Committee, respectively. Based on the reviews, and discussions with the TPC members if needed, the Program Chairs identified 12 conference papers for publication and presentation at the AIMS conference. In addition, three short papers were selected for presentation. For the PhD workshop, the Workshop Chairs selected eight papers to be included in the proceedings and to be presented at the conference.

We would like to thank the many people who helped make AIMS 2009 such a high-quality and successful event. Our thanks first go to the PhD Workshop Chairs David Hausheer and George Pavlou and to the Tutorial and Keynote Chairs Emil Lupu and Iris Hochstatter for all their efforts in constructing the technical program. In addition, we like to acknowledge the great review work performed by the members of the committees and the additional reviewers. Another special thanks goes to all authors who submitted their contributions to AIMS 2009. Finally, we would like to thank Springer, namely Alfred Hofmann and Anna Kramer, for the smooth cooperation in finalizing these proceedings. Last but not least, special thanks go to Tiago Fioreze, Silvia Meijran, Giovane Moura, and Anna Sperotto for the local organization and to the University of Twente for hosting the AIMS 2009 conference.

April 2009

Ramin Sadre
Aiko Pras

Organization

AIMS 2009 was organized by the EC IST-EMANICS Network of Excellence (#26854) and co-sponsored by IFIP WG 6.6 and the Strategic Research Orientation of the University of Twente on Dependable Systems and Networks (DSN).

General Chair

Aiko Pras	University of Twente, The Netherlands

Program Chairs

Ramin Sadre	University of Twente, The Netherlands
Aiko Pras	University of Twente, The Netherlands

PhD Workshop Chairs

David Hausheer	University of Zurich, Switzerland
George Pavlou	University College London, UK

Tutorial/Keynote Chairs

Emil Lupu	Imperial College London, UK
Iris Hochstatter	Universität der Bundeswehr München, Germany

Steering Committee

Arosha Bandara	The Open University, UK
Mark Burgess	Oslo University College, Norway
Olivier Festor	INRIA Nancy - Grand Est, France
David Hausheer	University of Zurich, Switzerland
Aiko Pras	University of Twente, The Netherlands
Jürgen Schönwälder	Jacobs University Bremen, Germany
Rolf Stadler	KTH Royal Institute of Technology, Sweden

Technical Program Committee

Arosha Bandara	The Open University, UK
Marcus Brunner	NEC Europe Ltd., Germany
Isabelle Chrisment	ESIAL, Nancy University, France
Alva Couch	Tufts University, USA
Gabi Dreo Rodosek	Universität der Bundeswehr München, Germany
Olivier Festor	INRIA Nancy - Grand Est, France

Alberto Gonzalez	KTH Royal Institute of Technology, Sweden
David Hausheer	University of Zurich, Switzerland
Heinz-Gerd Hegering	Leibniz Supercomputing Center, Germany
Georgios Karagiannis	University of Twente, The Netherlands
Alexander Keller	IBM Global Technology Services, USA
Antonio Liotta	Eindhoven University of Technology, The Netherlands
Jorge Lobo	IBM T. J. Watson Research Center, USA
Emil Lupu	Imperial College, UK
Hanan Lutfiyya	University of Western Ontario, Canada
Jean-Philippe Martin-Flatin	NetExpert, Switzerland
Hermann De Meer	University of Passau, Germany
Jiri Novotny	Masaryk University, Czech Republic
Bruno Quoitin	Université catholique de Louvain, Belgium
Danny Raz	Technion, Israel
Helmut Reiser	Leibniz Supercomputing Center, Germany
Jürgen Schönwälder	Jacobs University Bremen, Germany
Joan Serrat	Universitat Politècnica de Catalunya, Spain
Maarten van Steen	Vrije Universiteit Amsterdam, The Netherlands
Radu State	University of Luxembourg, Luxembourg
Burkhard Stiller	University of Zurich, Switzerland
Robert Szabo	Budapest University of Technology and Economics, Hungary
Filip De Turck	Ghent University, Belgium
Kurt Tutschku	University of Vienna, Austria
Marcel Waldvogel	University of Konstanz, Germany
Lisandro Zambenedetti Granville	Universidade Federal do Rio Grande do Sul, Brazil
Tanja Zseby	Fraunhofer FOKUS, Germany

PhD Workshop Committee

Gabi Dreo Rodosek	Universität der Bundeswehr München, Germany
Olivier Festor	INRIA Nancy - Grand Est, France
Emil Lupu	Imperial College London, UK
Hanan Lutfiyya	University of Western Ontario, Canada
Joan Serrat	Universitat Politècnica de Catalunya, Spain
Rolf Stadler	KTH Royal Institute of Technology, Sweden
Burkhard Stiller	University of Zurich, Switzerland

Local Organization

Tiago Fioreze	University of Twente, The Netherlands
Silvia Meijran	University of Twente, The Netherlands
Giovane C. M. Moura	University of Twente, The Netherlands
Anna Sperotto	University of Twente, The Netherlands

Additional Reviewers

Rémi Badonnel	University of Nancy - INRIA, France
Desislava Dimitrova	University of Twente, The Netherlands
Idilio Drago	University of Twente, The Netherlands
Gábor Fehér	Budapest University of Technology and Economics, Hungary
Richard Holzer	University of Passau, Germany
Peter Kersch	Budapest University of Technology and Economics, Hungary
Wouter Klein Wolterink	University of Twente, The Netherlands
Andras Korn	Budapest University of Technology and Economics, Hungary
Vlado Menkovski	Eindhoven University of Technology, The Netherlands
Anna Sperotto	University of Twente, The Netherlands
Mark Timmer	University of Twente, The Netherlands
Ha Manh Tran	Jacobs University Bremen, Germany

Table of Contents

Secure User-Controlled Lightpath Provisioning with User-Controlled Identity Management

Bob Hulsebosch, Robert de Groote, and Martin Snijders

Telematica Instituut, PO Box 589,
7500 AN Enschede, The Netherlands
{Bob.Hulsebosch,Robert.deGroote,Martin.Snijders}@telin.nl

Abstract. To allow user applications to securely make use of various lightpath resources distributed across multiple domains in a user-friendly and privacy-preserving way requires identity management functionality. Identity and attribute information has to be provided by the user to allow for authorized access to these resources. An identity management framework can facilitate such information exchange. We describe an architecture and prototype that allows the user to setup an end-to-end lightpath that spans multiple network domains while being in control of the personal credentials he has to provide for that purpose. The architecture combines the user-controlled lightpath paradigm with novel user-centric identity management technology. This combination allows the user transparent and non-intrusive access to multiple services that are required for reservation and utilization of network resources in order to setup an end-to-end lightpath.

Keywords: User centric, identity management, federation, lightpath provisioning, network resources, trust.

1 Introduction

In the user-controlled lightpath provisioning paradigm, end-users take the initiative to set up an optical peering connection. This peering connection must fulfill the end-user's needs (desired capacity, duration and starting time) and capabilities (available budget). End-users indicate through an application which network resources to allocate for an end-to-end lightpath. For instance, the Dynamic Resource Allocation Controller (DRAC) application allows the end-user to schedule and use optical end-to-end connections, i.e. lightpaths [1]. These lightpaths may exist within a single optical network service provider domain but may very well involve multiple providers in different domains as well. Typically, the network service providers offer the user a service that allows him/her to schedule network resources for lightpath provisioning.

Controlling and enforcing access to lightpath resources belonging to different owners is one of the challenges of user-managed lightpath provisioning. A critical requirement for end-to-end connection provisioning in this context is the existence of a certain amount of trust among the multiple independent stakeholders involved. Identity and related attribute information have to be exchanged to satisfy the trust requirements and allow users to take control of and schedule optical network resources in collaborative domains. To allow user applications to make use of these network services that are distributed across multiple domains in a user-friendly way requires identity management functionality.

Identity management is concerned with controlling the pieces and types of information, i.e. attributes and identifiers, pertaining to a party that are (made) available to other parties. More concretely, it can be thought of as the processes/functions, protocols and policies used to establish, collect, interpret and access this information in a secure manner. User access to provider-offered network services has to be controlled and enforced and is facilitated by (federated) identity management. The identity provider plays an important role in any identity management framework. The identity provider is trusted by all parties in the federation and manages and links digital identities of the user. The identity provider can authenticate the user himself or delegate it to the authentication server that is authorative for that user.

Different identity management solutions exist nowadays that facilitate single sign-on and secure attribute exchange. Examples are SAML2.0 [2], Shibboleth [3], and WS-Federation [4]. These solutions are characterized as being identity provider centric, i.e. the identity provider controls the information flow, and can very well be used for secure lightpath provisioning [5]. Recent developments in the identity management arena, such as OpenID [6], Microsoft CardSpace [7] and Higgins [8], are much more user-centric and address ease-of-use and privacy protection among disparate business contexts. In these user-centric solutions the user is to a certain extent in control and at least aware of the information that is communicated to service providers. The use of such identity-centric identity management solutions for secure lightpath provisioning would be much more in line with the user-controlled paradigm and would therefore be preferred above the identity provider centric ones. In this paper we investigate if this assumption is true and if both paradigms can be combined into an overall user-centric architecture for secure lightpath provisioning.

The rest of this paper is organized as follows. Section 2 describes the concept of user-centric identity management and compares it with the identity provider centric model. Section 2 also briefly discusses and compares several user-centric identity management solutions. The architecture and prototype that combines user-centric identity management with user-controlled lightpath provisioning is presented in Section 3. This section discusses the underlying trust model and describes the single-domain case as well as the multiple-domain case. Finally, Section 4 concludes with a summary of our findings and indicates future steps for lightpath provisioning.

2 User Centric Identity Management

User-centric identity management - also referred to as Personal Identity Frameworks or Identity 2.0 - focuses on user empowerment in sharing personal information and self-determination in establishing relationships with relying parties. User-centricity distinguishes itself from other notions of identity management by emphasizing that the user maintains control over ‘what, where, when, and to whom’ a user’s identity attributes are released. The difference with the identity provider centric approach is shown in Fig. 1 below. Clearly, in the identity provider centric model the user is unaware of what identity information is communicated between the identity provider and the service provider also called relying party; the only thing the user has to do is to authenticate himself towards the identity provider. In the user-centric model the user is, besides an authentication session during e.g. the creation of an information card, also asked for consent regarding the identity provider to be used and the attributes to be communicated to the relying party by means of an information card.

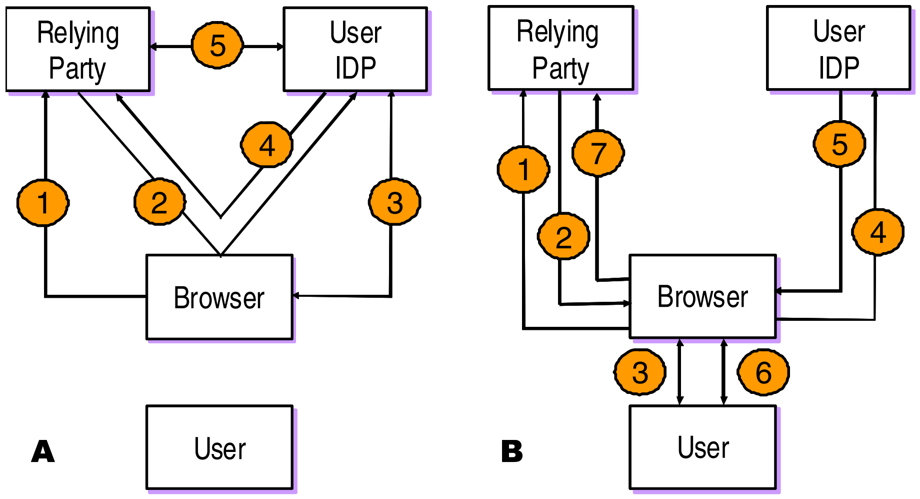


Fig. 1. Identity provider (IDP) centric (A) versus User-centric (B) identity management. The numbers reflect the sequence of the message flow.

Trust in user centric identity management still needs to be established. The relying party has to trust the user's identity provider. However, being in the credential exchange loop, the user can be made liable for the credentials he has provided thereby allowing for a more relaxed trust relationship between the relying party and the identity provider. On the other hand, the user is offered the means to control what credentials are exchanged with relying parties thereby guaranteeing his/her privacy.

The user-centric identity management approach thus allows the user to control the communication of personal credentials to relying parties. The primary approaches behind the user-centric model are identifier-based (such as OpenID) and information card (such as CardSpace) systems, plus other supporting standards and infrastructure components such as Higgins. The identifier OpenID protocol suffers from several drawbacks related to trust and privacy (i.e. sensitive to phishing attacks) making it less suitable to be used for business transactions [9]. Therefore the information card (infocard) approach adopted by CardSpace and Higgins seems best suitable for user-controlled lightpath provisioning as it offers more security, privacy and trust. Higgins should be preferred over MS CardSpace as it offers interoperability and therefore allows the use of both SAML and infocard-like solutions. The infocard approach also offers more security in the sense that phishing attacks that occur during the frequent redirections of e.g. OpenID are prohibited. The infocard always ensures that the user is redirected to the right identity provider and not to a rogue one.

In the next section we illustrate how infocard-based identity management solutions like CardSpace or Higgins may be used for user-controlled lightpath provisioning based on DRAC services that are used for optical network resource management in for instance the SURFnet6 network of the Dutch national research and educational network provider SURFnet [10].

3 User-Controlled Lightpath Provisioning with Infocards

This section describes an architecture for secure user-controlled lightpath provisioning by means of user-controlled identity management.

3.1 Scenario

The following scenario illustrates the scope of our work:

John is a surgeon and has his own personalized internet portal that provides links to recorded high-resolution videos of surgical operations. He uses these videos to learn how colleague surgeons tackle the operation or for educational purposes. The portal allows him to watch the videos at the hospital or at his private office. Prior to getting access to his portal John has to select an infocard from the Identity Selector application. John selects his self-issued 'Surgeon' card and gets access to his portal. Subsequently, he clicks on one of the videos he is interested in. Immediately the lightpath service is triggered informing John that a lightpath is required for the selected video and that more personal information is required to setup such a lightpath. For this purpose, John selects his 'DRAC' infocard and presents it to the lightpath service. Satisfied with the provided credentials, the lightpath service asks John when he wants to see the video. John wants to see it immediately. The lightpath service determines the most suitable lightpaths and asks John to select one. After having selected the shortest path, the DRAC services of the network providers constituting the lightpath are contacted by the lightpath service. A few seconds later the lightpath is provisioned and John is able to watch the video on his screen.

Two cases can be identified in this scenario. One case that involves only a single network provider and thus a single DRAC-service and one case that involves multiple providers. In order to coordinate the lightpath provisioning efficiently we assume the existence of a so-called lightpath service. This lightpath service takes care of the discovery of DRAC-services, concatenating them into an end-to-end lightpath, scheduling, and provisioning of the lightpath. Before we describe the message flows of both cases in more detail we first present a high-level functional overview.

3.2 Trust Model

The trust model for user-controlled lightpath provisioning is complex and by far not established yet. For single domain provisioning the network service provider can execute the authentication and authorization process himself. This solution becomes less manageable when a large group of users from different institutions are allowed access to lightpath resources and this is typically the case for many network service providers. A better approach is to share identity information that is stored and secured in one domain with other domains. In other words, identity information is made portable across contexts and institutional boundaries according to established policies that dictate, among other things, formats and options, as well as trust and privacy/sharing requirements. Such portability is usually realized by means of identity federation. The federation establishes trust between the network service providers and the users. The most ideal federation topology is represented by Fig. 2A.

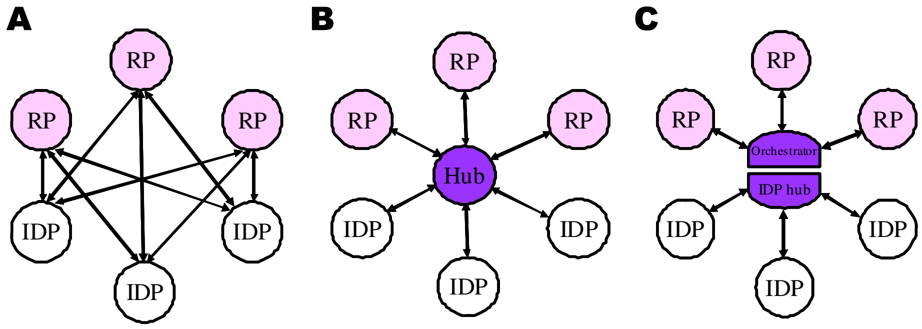


Fig. 2. Federated identity management topologies

This topology assumes bilateral trust relationships between identity providers (IDPs) and Relying Parties (RPs). However, if the number of IDPs and RPs increases, and more importantly, if access to the services offered across all IDP-users is required, scalability issues are lurking. The introduction of a common platform or hub that is trusted by all parties is a solution to make the federation more scalable. The hub-model is illustrated in Fig. 2B.

The hub more or less becomes an identity broker that is trusted by all parties. The model implies, however, that there is less trust between the RPs and the IDPs. The RPs and the IDPs only have to trust the hub.

Besides federation of identity, lightpath provisioning also requires federation of network resources. In particular for the case of cross-domain lightpath provisioning. For cross-domain lightpath provisioning somehow the service providers constituting the end-to-end lightpath must be determined. Once they are discovered and contacted resources must be scheduled in such a way that all concatenated parts form, at the right time, a lightpath to a certain destination. This requires a lightpath orchestrator service that operates across and is trusted by multiple network service providers. Merging the orchestrator service with the federated identity hub results in a topology that is shown in Fig. 2C. We call the orchestrator service Lightpath Service.

Instead of a central orchestrator service an alternative hop-by-hop via routing-based mechanisms is imaginable for the establishment of a cross-domain lightpath [11]. This approach however results in fragile and long chains of trust between network service providers and hampers efficient identity exchange. Either all user information must be communicated between the network service providers or each of the providers individually must request the user's IDP for credentials. In a user-centric identity management scheme this will not work, i.e., the user will be bothered too much by all network service providers making the service too intrusive and unfriendly. Therefore the trust topology as depicted at the bottom of Fig. 2 will be used as the basis for our user-controlled lightpath provisioning by user-controlled identity federation architecture.

3.3 Architectural Overview

Fig. 3 provides a high-level overview of the functional components that are needed for secure user-controlled lightpath provisioning with user-controlled identity management.

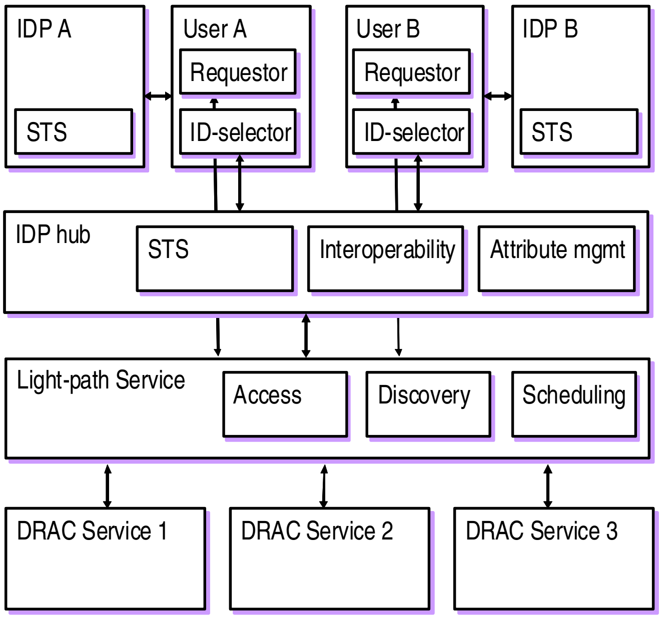


Fig. 3. High-level functional overview

The user, interacting through the identity selector on the service requester (a client application running on a client system), may have identities issued by one or more identity providers. Each such digital identity of the user is represented by an infocard that the identity selector on the service requester system can process. An infocard endows the identity selector with the ability to request and obtain security tokens from the corresponding identity provider when the user selects that digital identity for use in a given interaction context. For instance, the infocard presented to the Portal will differ from that presented to the DRAC services.

At the IPD side, either self-issued or managed identity providers are available that run a Security Token Service (STS) to which a requester or target service (Portal or Lightpath Service) can submit security token requests. The STS can issue security tokens containing the requested claims after the requester has provided suitable proof of authentication as required by the identity provider's security policy. According to our scenario the user's IDP will provide tokens for accessing the video portal whereas the IDP-hub will provide tokens for accessing the Lightpath Service.

The architecture supports both single- and multi-domain lightpath scenarios. It assumes a Lightpath Service that act as an orchestrator service towards multiple network service providers that use a DRAC-service for their network management. The Lightpath Service collects relevant information regarding lightpath provisioning across multiple DRAC service providers and thereby operates on a meta-level. This relevant information is related to accessing, discovery and scheduling of lightpaths. The architecture furthermore assumes that the Lightpath Service is trusted by the network service providers. The Lightpath Service interacts with the IDP-hub that represents the federation of the user's IDPs.

The Lightpath Service offers, on behalf of all network service providers, a common policy that must be fulfilled prior to getting access to its services. This policy is established under mutual agreement of all members of the federation. This implies that a user only once has to authenticate himself towards the Lightpath Service and consequently has access to all DRAC services in the federation.

The IDP hub is trusted by the network service providers and collects user identity claims from the user IDPs. These claims are converted into tokens via its secure token service. The tokens are presented to the Lightpath Service that uses them to enforce user access. Additionally, the IDP-hub may provide identity interoperability functions as different user IDPs may use different identity management standards and products. Furthermore, it may play a role in attribute management aspects related to for instance semantic and syntactic mapping of attributes or it may add several attributes/claims to the token such as federation membership status (e.g. gold – silver –bronze).

3.4 Cross Domain Provisioning

Being the most complex case we start with cross-domain lightpath provisioning. In that case, multiple DRAC-services constitute the end-to-end lightpath and multiple lightpath requests have to be made. To simplify the scenarios studies here, we assume that all DRAC-service providers act in the same federation. This implies that we may also assume that there is a common policy for each user in the federation. In other words, the federation likely will define and make obligatory such a common policy for all its DRAC-service providers. In that case, a single infocard would be sufficient to serve all the DRAC-service providers in the federation. The Lightpath Service is the most likely candidate to enforce this policy and to receive the infocard. Offering the Lightpath Service an infocard allows the user access to its services. The user then indirectly, i.e. via the Lightpath Service, obtains access to the network resources of the individual DRAC-service providers. The corresponding message flow is shown in Fig. 4. We assume that the user has already entered the portal via username/password authentication or via the use of a self-issued infocard.

The message flow starts with the creation of an infocard. This out-of-band process is done via the IDP of the user and in cooperation with the IDP-hub. For instance, the infocard may be created and downloaded during a first time visit at the Lightpath Service. During the creation process the user has to authenticate himself towards his IDP. After all, the user's IDP knows best how to authenticate the user and typically stores user attributes or knows where to find them. Being part of the federation, both

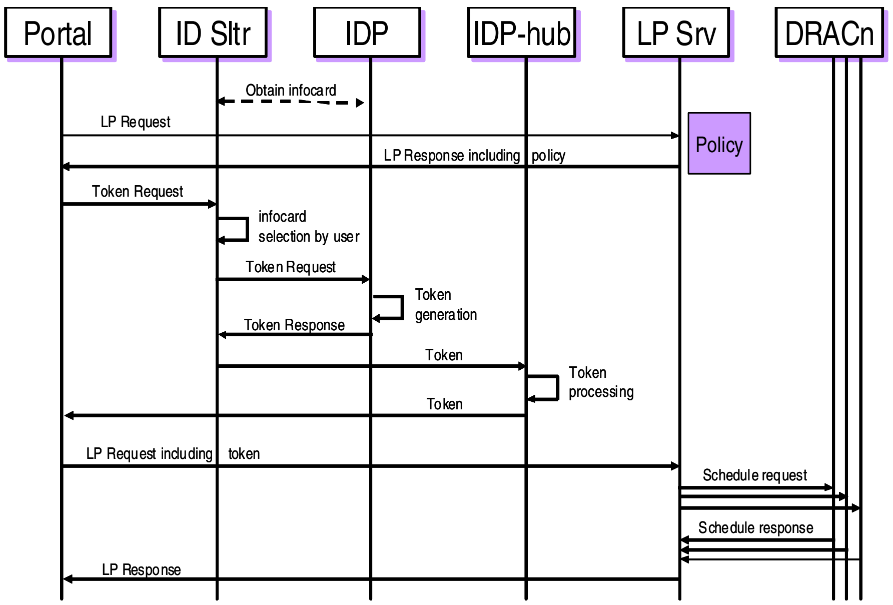


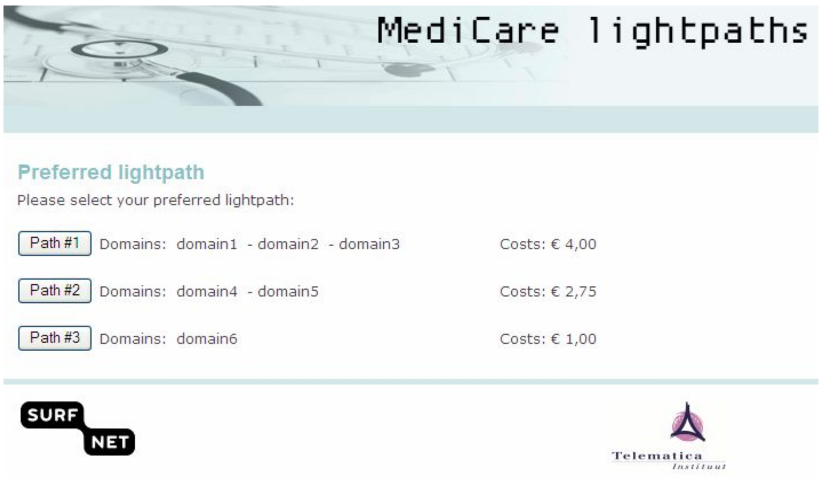
Fig. 4. Message flow for nDRAC services that trust the Lightpath Service and the IDP-hub. The IDP-hub is trusted by the user IDPs.

the IDP and the IDP-hub know and trust each other and thus can securely exchange security tokens with each other.

Note that the infocard is not the security token; it represents the token issuance relationship that the user has with the IDP and indirectly with the IDP-hub. In order to provide some assurance to the user that an infocard was indeed issued by the IDP, the infocard should carry inside a digitally signed envelope (i.e. enveloping signature) signed by the IDP. Additionally, to meet the Lightpath Service policy requirements, the token is signed by the IDP-hub as well. After all, the Lightpath Service Provider only trusts the IDP-hub and not the user's IDP.

The Lightpath Service receiving a request for a lightpath specifies in its policy what kind of identity information is required in order to access and use its service. The policy also specifies the IDP-hub that it trusts, i.e. is part of the federation. This allows the identity selector service to present the right infocards to the user. Once the user has selected an infocard it will be presented to the IDP and a token will be requested. The IDP creates a token with the token service and returns it to the identity selector service. Subsequently, the latter forwards the token to the IDP-hub for further processing. The IDP-hub could add additional federation-specific attributes such as membership status (Gold – Silver – Bronze) to the token or it could convert the token into a SAML authentication and attribute requests format that is desired by the Lightpath Service. Here the use of Higgins shows added-value compared to MS CardSpace. The Higgins-enabled IDP-hub and identity selector are able to talk infocards as well as SAML towards each other. After processing, the new token will be signed by the IDP-hub and returned to the identity selector service. The identity selector service on its turn sends it to the Lightpath Service. If the received token contains the proper credentials, the user will be granted access and can specify his lightpath requirements. The Lightpath Service will then try to find possible lightpaths that meet the specification. For this purpose the Lightpath Service needs to access the different DRAC differences that constitute the end-to-end lightpath. It will use the token for this purpose as well.

Prior to sending the token to the DRAC-service provider, it will be encrypted with the DRAC-service provider's public key. This is done for multiple DRAC-service providers. The token grants the Lightpath Service access to network resource information available at the different DRAC-service providers allowing it to calculate lightpaths after the user has specified the desired lightpath via the Lightpath Service GUI (see Fig. 5). The possible lightpaths are shown via the GUI to the user who selects the most suitable one. The Lightpath Service then schedules the selected lightpath resources in the DRAC-services. This scheduling ultimately results in the provisioning of the lightpath. Note that during the whole process of lightpath provisioning the user only has to present infocards, he doesn't need to authenticate himself once.



Preferred lightpath

Please select your preferred lightpath:

Path #1	Domains: domain1 - domain2 - domain3	Costs: € 4,00
Path #2	Domains: domain4 - domain5	Costs: € 2,75
Path #3	Domains: domain6	Costs: € 1,00

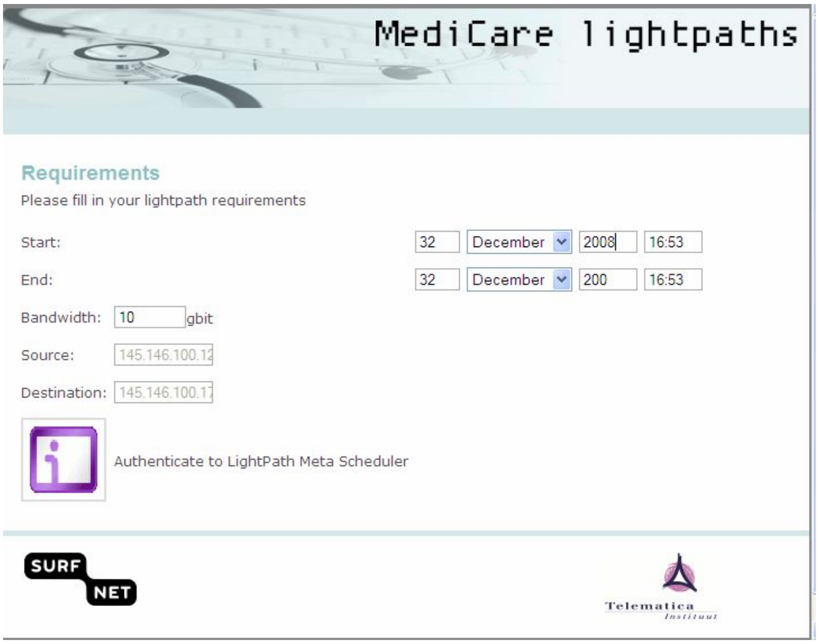
Fig. 5. Lightpath selection in the Lightpath Service GUI

3.5 Single Domain Provisioning

In case a user requires an end-to-end lightpath inside a single network domain, e.g. the SURFnet6 optical network, the message flow shown in Fig. 4 is applicable as well. The added value of the Lightpath Service becomes negligible in this scenario; functionality-wise the Lightpath Service collapses with the DRAC service at the network provider.

3.6 Implementation

The goal of our implementation effort is to demonstrate and validate the role of user-centric identity management in user-controlled lightpath establishment. The

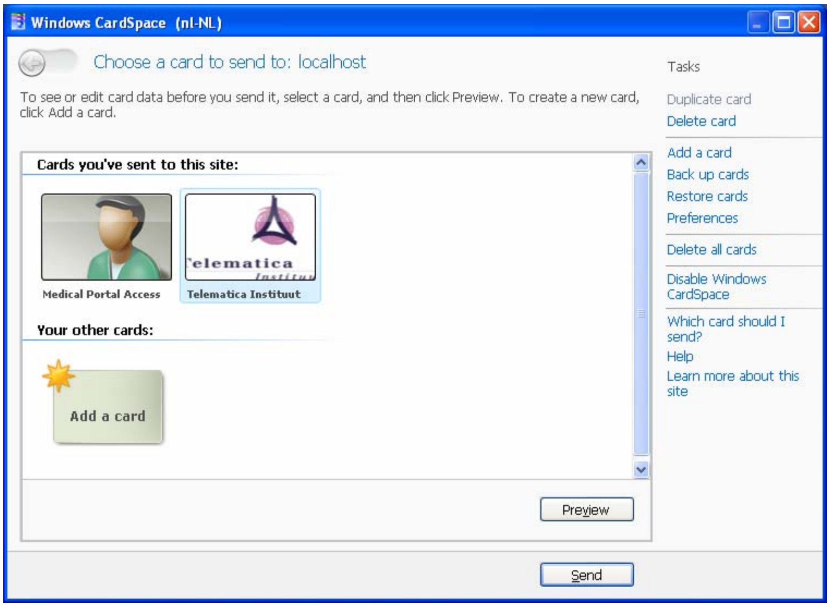


Requirements
Please fill in your lightpath requirements

Start: 32 December 2008 16:53
End: 32 December 200 16:53
Bandwidth: 10 gbit
Source: 145.146.100.12
Destination: 145.146.100.17

Authenticate to LightPath Meta Scheduler

Fig. 6. Lightpath Service GUI



Choose a card to send to: localhost

To see or edit card data before you send it, select a card, and then click Preview. To create a new card, click Add a card.

Cards you've sent to this site:

Your other cards:

Fig. 7. Identity Selector GUI

above-mentioned scenario motivated us to build a medical video portal application that could benefit from user-controlled lightpaths, allowing high definition video material to be streamed without delays from and to locations preferred by the user. The video portal and the Lightpath Service it leverages operate in different administrative domains. This requires the user to authenticate twice – once to the video portal, and once to the Lightpath Service. The whole system is implemented and relies heavily on the Higgins Open Source Identity Framework [12]. The IDP is an extension of Higgins' STS IDP, both the portal and IDP-hub are web applications implemented in Java. All components run in an Apache Tomcat web server.

Several WS-* implementations are used during the whole process. WS-MetadataExchange is used by the Identity Selector to obtain the policy from the IDP/STS and by the Portal to retrieve the policy from the Lightpath service. The policies issued by the Lightpath Service and IDP/STS are specified using WS-SecurityPolicy. Policies inform the Portal that the Lightpath Service requires a SAML token from the STS of the IDP and that such a token must contain a claim with the membership status of the requestor, and inform the identity selector of e.g. the required authentication method. WS-Trust is used by the identity selector in order to obtain a security token from the IDP/STS.

The identity selector used in our implementation is the standard Microsoft CardSpace card selector that comes along newer versions of Microsoft Windows. The card selector is triggered by simple policy objects embedded in an HTML page. A simple mechanism is in place to bootstrap WS-SecurityPolicy policies into these embedded objects so that they can be intercepted by the card selector.

The Lightpath Service is developed in Java. All GUI widgets are Java Swing controls. The application acts as a wizard, leading the user from authentication, to specifying lightpath request parameters, viewing available lightpaths, and finally reserving a chosen lightpath. The GUIs of the Lightpath Service, including the CardSpace login button, and the identity selector are shown in Fig. 6 and Fig. 7, respectively.

4 Related Work

The use of identity management solutions for lightpath establishment is not new. An identity provider centric solution is presented in [5]. DNS and its security extensions (DNSsec) are used to guarantee trust between the involved parties. We have proposed a user centric alternative to offer the user better insight in the communication of his personal information among those parties. The DNSsec approach can very well be combined with user centric identity management for dynamic trust establishment between the parties involved for lightpath setup.

In section 3.2 we already mentioned that the hop-by-hop approach for secure lightpath provisioning, as described by Gommans at al. [12], is not very suitable for user centric identity management. In this model authentication and authorization is communicated between the authentication, authorization and accounting (AAA) servers of the different network providers without any user involvement. The advantage of the model, however, is that there is little reliance on central or coordinating entities in the network whereas we rely on a central identity hub and Lightpath Service orchestrator. A similar, approach for user-controlled lightpath provisioning is described in [13]. In this paper a policy restricted signaling architecture is proposed that allows users to reserve lightpaths over multiple domains whilst ensuring that management rules of each domain are enforced. Again, the user is offered little means for controlling the release of his personal information in order to get access to network resources.

GRID-based solutions typically rely on a Virtual Organization (VO) that enforces centralized access to distributed resources [14]. While such a VO-based, central AAA approach is straightforward and intuitive, it becomes impractical to administer as soon as VOs expand into distributed multi-institutional collaborations, VO memberships change dynamically, and user rights vary on a periodic basis or per user's role in an organization. Obviously, it is impractical and does not scalable for the VO itself to store and manage all the identities and corresponding attributes. Such identity-related information could better be stored and managed by the registered members themselves, allowing the VO to only administer at the organizational level. User centric identity management provides this functionality.

5 Discussion and Conclusions

We have presented an overall security architecture that combines secure and trustworthy user-controlled provisioning of lightpaths with user-centric identity management.

Though the user wants to be in control of the dissemination of his personal credentials, too much consent works counter productive. Straightforward implementation of user-centric identity management solutions is therefore not going to work as multiple services have to be provided with credentials. From a user's point of view it is not friendly to submit an infocard to each requesting network service provider that is part of the end-to-end lightpath. We foresee, however, the existence of a meta-service that facilitates the user in setting up lightpaths across multiple network service providers. Such a meta-service is trusted by all parties that besides providing basic functionality as service discovery and scheduling also takes care of access control. The latter implies that the user only once has to provide an infocard to this meta-service, called Lightpath Service, prior to getting access to multiple network service providers. Furthermore, an IDP-hub seems, from a scalability point of view, required to orchestrate the provisioning of identity information towards the Lightpath Service. The IDP-hub federates the IDPs of the users. Given the heterogeneity of identity management solutions it is best to turn this IDP into some kind of identity meta-system that is capable of interworking with the different identity management flavors. Higgins for instance provides such a meta-system.

With an increasingly more prominent role and use of web services in lightpath provisioning federating them becomes important: lightpath services need to talk to DRAC services in a secure and trusted manner. Besides lightpath specific information, identity information is required as well for authorization. However, federated web services communication not always occurs via the browser and thus without the knowledge of the user. Somehow the user must be informed about the usage of identity information by federated web services. The current user-centric identity management solutions, however, do not support this functionality. It is therefore questionable if today's user-centric identity management will flourish in service oriented architectures. The challenge is to find solutions that allow the user some control of the release of his personal information in federated web services environments without being too intrusive. Delegation models might provide a solution for such environments and might be worth for further future investigation. For instance, the Lightpath Service could be authorized by the user, via its IDP, to contact and access certain DRAC service providers. Already, the MS Geneva Framework, that comprises amongst others MS Cardspace, provides functionality for such delegation [15]. Alternatively, the applicability of Dynamic SAML due to its flexibility and scalability benefits regarding trust could be considered as well [16].

Lack of IDPs that generate trustworthy tokens is another drawback of current user centric-identity management solutions. The entity controlling the hub could very well become a provider of such a secure token service for lightpath provisioning (or for service access in general). Furthermore, the user's infocards are all stored on a single PC thereby hampering the user in setting up lightpaths from another device. Future research could therefore also focus on making infocards accessible from any device, anytime, and anywhere. They should move along with the user.

Acknowledgments. The work described in this paper was conducted in the GigaPort Next Generation research program, sponsored by the Dutch government and coordinated by SURFnet.

References

1. Nortel. Application Brief: Dynamic Resource Allocation Controller (DRAC) (2006)
2. OASIS Security Assertion Markup Language (SAML) 2.0 (March 2005), `http://saml.xml.org/saml-specifications#samlv20`
3. Shibboleth project website, `http://shibboleth.internet2.edu/`
4. Lockhart, H., et al.: WS-Federation Language, Version 1.1 (2006), `http://www.ibm.com/developerworks/library/specification/ws-fed/`
5. Hulsebosch, R.J., Bargh, M.S., Fennema, P.H., Zandbelt, J.F., Snijders, M., Eertink, E.H.: Using Identity Management and Secure DNS for Effective and Trusted User-controlled Lightpath Establishment. In: International Conference on Networking and Services, ICNS 2006, Silicon Valley, USA, Marriott Hotel, Santa Clara, July 16-19 (2006)
6. OpenID, `http://openid.net/`
7. MS Cardspace, `http://msdn.microsoft.com/en-us/netframework/aa663320.aspx`
8. Higgins, `http://www.eclipse.org/higgins/`
9. Bournez, C., Bichsel, P.: First Report on Standardisation and Interoperability - Overview and Analysis of Open Source Initiatives, Combined deliverable: Merger of D3.3.1 and D3.4.1, FP7 EU Primelife project, May 30 (2008)
10. Van der Pol, R., Dijkstra, F.: Network and Capacity Planning in SURFnet6. In: TNC 2009 (2009) (submitted)
11. Gommans, L., Dijkstra, F., de Laat, C., Tall, A., Wan, A., van Oudenaarde, B., Lavian, T., Monga, I., Travostino, F.: Applications Drive Secure Lightpath Creation across Heterogeneous Domains. IEEE Communications Magazine, Optical Control Planes for Grid Networks: Opportunities, Challenges and the Vision 44(3), 100–106 (2006)
12. Higgins Open Source Identity Framework (2008), `http://www.eclipse.org/higgins`

13. Truong, D.L., Cherkaoui, O., Elbiaze, H., Rico, N., Aboulhamid, M.: A Policy-based approach for user-controlled Lightpath Provisioning. In: IFIP/IEEE NOMS, April 2004, pp. 859–872 (2004)
14. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco (2003)
15. Brown, K., Mani, S.: Microsoft Code Name "Geneva" Framework Whitepaper for Developers (2008)
16. Harding, P., Johansson, L., Klingenstein, N.: Dynamic Security Assertion Markup Language: Simplifying Single Sign-On. IEEE Security & Privacy 6(2), 83–85 (2008)

A Statistical Analysis of Network Parameters for the Self-management of Lambda-Connections

Tiago Fioreze[1], Lisandro Granville[2], Ramin Sadre[1], and Aiko Pras[1]

[1] University of Twente
Design and Analysis of Communication Systems (DACS)
Enschede, The Netherlands
{t.fioreze,r.sadre,a.pras}@utwente.nl
[2] Federal University of Rio Grande do Sul,
Institute of Informatics
Porto Alegre, Brazil
granville@inf.ufrgs.br

Abstract. Network monitoring plays an important role in network management. Through the analysis of network parameters (*e.g.*, flow throughput), managers can observe network behavior and make decisions based on them. The choice of network parameters although should be relevant for each specific objective. In this paper, we focus on the analysis of network parameters that are relevant for our self-management of lambda-connections proposal. This proposal consists of an automatic decision process to offload large IP flows onto lambda-connections. This paper aims at statistically analyzing a list of potential network parameters as relevant estimators for flow volume. The main contribution of this work is the introduction of a statistical methodology to validate that some few network parameters can be considered as good predictors for flow volume. These predictors are therefore of great interest to be used in our automatic decision process.

1 Introduction

Recently, a clear change in the set of core technologies that form the Internet is being observed. Internet backbones that once relied solely on IPv4 routing to deliver end-to-end communications are moving towards hybrid solutions. One example is the rising of hybrid optical switching and packet forwarding networks. These hybrid networks are composed of intermediate devices that are both switches at the optical level and traditional routers at the network level. In such an environment, network flows can traverse a hybrid network through either an optical path or a chain of routing decisions. In this hybrid environment, moving large flows from the IP level to the optical level presents the following advantages: (a) flows experience faster and more reliable transmissions with optical switching than with traditional IP routing; (b) remaining flows at the IP level also experience better services because the network layer is less congested after offloading these large flows.

The decision today to select which flows will be placed in which level (optical or network) is still up to network human operators. However, this is not a trivial decision to be made. Operators have to cope with several network parameters (*e.g.,* flow throughput, flow duration) to come up with a balanced plan. The choice of a proper set of network parameters is therefore crucial for this task. Besides, network operators have to select flows in a timely manner, since it is far cheaper to send traffic at the optical level than at the IP level, as pointed out by Cees de Laat's research [1].

A new approach for the management of hybrid networks has been investigated by us [2] to speed up the process of flow selection. Our approach, called self-management of hybrid networks, consists of monitoring a network of interest to automatically decide which IP flows should be transported at the optical level and which other flows should remain at the network level. The main targets of our approach are flows that despite being small in number are responsible for most of the IP traffic, *i.e.*, the so called *elephant flows* [3] [4]. However, to properly select those flows, our self-management approach needs to employ a set of parameters while analyzing network data.

The goal of this paper is to present a study on selecting a proper set of potential parameters to be used as good predictors for the traffic volume generated by *elephant flows*. Volume prediction is employed in our self-management approach to reduce the time (and consequently cost) of a high-volume flow staying at the IP level consuming network resources. The faster a high-volume flow is detected, the lesser the network resources it consumes. In this context, the main research questions that motivate the investigation presented in this paper are: *Which network parameters could be used to predict the volume of flows?*, and *How can these parameters be evaluated?*

In order to answer these questions, we have first carried out a study on the literature about the parameters (*e.g.*, throughput, duration) that can be observed using current management technologies (*e.g.*, NetFlow, SNMP). To find out how accurate each parameter is, we collected and analyzed network traces from the University of Twente (UT) network. From the collected traces, those flows that have been classified as elephant ones have been filtered and analyzed. We then propose the employment of statistical techniques, such as correlation and tree classification, as a methodology to evaluate which network parameters are more relevant to predict flow volume. Such a methodology is therefore the main contribution of this paper.

The remainder of paper is structured as follows. In Section 2 we review the current research work on traffic characterization. In Section 3 we describe our methodology by presenting the decisions and steps taken to make our statistical analysis. In Section 4 we present the selected parameters. Finally, we close this paper in Section 5, where we draw our conclusions and future work.

2 Related Work

Investigations about flow characteristics have been carried out by researchers for some years already. Thompson *et al.* [5] present a flow-based characterization of network usage and workloads on a commercial backbone. The authors analyzed traffic data collected at one observation point on different time scales. Recently, Kim *et al.* [6] presented a detailed analysis of flow-based traffic characteristics. The metrics that have been considered were packets, bytes, and port distribution. It is interesting to mention that these two papers did not present an extensive reasoning about the chosen metrics.

More recently, Ribeiro *et al.* [7] used packet sampling as the measurement technique while mainly observing its effect on the flow size distribution. They also observed the effects on the packet counts, SYN information, and sequence number information. They concluded that TCP sequence numbers are essential for accurate flow size estimation, but no conclusions have been drawn about the best estimators.

We believe these works, both historical analysis of traces and comparison of diverse observation points, even if suitable for highly detailed studies, are missing one important dimension of analysis: they focus on estimating precise flow size distribution or packet distribution, but none of them focus on the best parameters to predict flow size.

3 Methodology

This section first presents our approach to create the list of potential network parameters. After that, the measurement set up to collect network traces is introduced. Finally, we explain the statistical techniques we used to evaluate the potential parameters.

3.1 List of Potential Network Parameters

Network parameters provide valuable information about the status of network traffic and devices (*e.g.*, routers, switches). In the context of our self-management approach, network parameters are important to predict the volume of each flow. This prediction is used to decide which flows should be moved to the optical level and which others should stay at the network level.

To define a set of relevant parameters for our autonomic decision process, a research on the literature has been carried out to list candidates parameters. Such candidates parameters have been taken from the following sources: (a) MIB modules MIB-II [8], RMON-2 [9], and SMON [10]; (b) the information model for the IP Flow Information eXport (IPFIX) protocol [11]. Since these sources also deal with information other than related to flows (*e.g.*, MAC to IP address translation in MIB-II), we have intuitively selected the subset of information that are helpful in predicting the volume of flows. The outcome of this research is the classification of network parameters divided in two main groups: flow identifier parameters, and flow behavior parameters.

Flow Identifier is a set of fields that defines groups of packets (flows) that share some common fields. Since the number of fields in a flow can be extensive [11], we limited the fields to those relevant to our approach:

1. *TCP/UDP port numbers* represent the communication end points that network applications use to exchange data via transport protocols (*i.e.*, TCP and UDP). Ports are important because some applications can generate more traffic than others. For example, a Telnet session is expected to generate far less data than an FTP session.
2. *IP addresses* identify network devices belonging to a certain network. Following the same reasoning above, some devices can generate more traffic than others. For example, a file server is expected to generate more traffic than an ordinary desktop.
3. *Network segments* are portions of computer networks that vary in size from small networks (*e.g.*, LAN) to large ones (*e.g.*, WAN). Some network segments can generate more traffic than others. For example, the University of Groningen network receives lots of data from the LOFAR [12] sensor network segment. The parameters that identify network segments are:

   - *Subnet*: Continuous bits in an IP address prefix used to identify a subnet;
   - *Autonomous System*: A collection of IP routing prefixes.
4. *Protocols* allow the communication between end points on top of IP. TCP and UDP are the most important Internet protocols. Several other protocols exist, but few of them have a representative significance in terms of traffic [13]. The value of the protocol number in the IP header is the parameter we consider.
5. *Type of Service (ToS)* is a 8 bits portion of the IP header that is reserved to define a service level request. Even though the support for ToS is not widely employed in current routers and therefore not widely used, it can represent a potential network parameter to track flows that generate considerable amount of traffic.

Flow Behavior is a set of network parameters used to characterize the behavior of flows. The potential flow behavior parameters that are relevant to our research are:

1. *Duration*: literature [14] [15] [16] has shown that some large flows may also be long in duration, normally presenting a heavy-tail distribution. Duration can therefore be a potential input parameter for our automatic decision approach.
2. The number of *packets* of a flow can give a good indicative about a flow's behavior.
3. The number of *bytes* of a flow is naturally an important parameter to be considered, since the automatic decision module aims at offloading high volume flows.
4. *Throughput* is the average rate of a communication. It is usually expressed in bytes per seconds (Bps), but it can also be measured in packets per second (Pps). Those two throughput units of measurement will be considered.

It is worth mentioning that analysis on flow identifiers has been previously carried out by us [17] [18] [19], and for the sake of space it will not be included in this paper. We thus focus on the analysis of the flow behavior group only.

3.2 Collecting Network Traces

This subsection shows how network traces have been collected in order to evaluate the set of parameters to describe flow behaviors. The collection process consisted in collecting NetFlow data from the UT NetFlow-enabled router running NetFlow version 9. This router exported NetFlow records to a flow collector hosted in our department. Traces from the UT network have been collected over a period of one day (Sep 18th, 2008). Once the traces were collected, they needed to be combined for our analysis. Since NetFlow reports long-lived flows in different records, one needs to combine the NetFlow records in order to closely compute the original flow duration, number of packets, and octets. The throughputs (in Bps and Pps), however, are calculated as an average throughput over the entire duration of the flow.

In order to combine NetFlow records, there is a need to determine the maximum gap that separates two consecutive flow records of the same flow. We have deliberately chosen a gap of 30 seconds, which is a common value for the TCP TIME-WAIT state. We then decided that all NetFlow records of the same flow whose gap was smaller or equal to 30 seconds are grouped into the same flow. Our whole analysis has been made then over combined flows rather than using the original NetFlow records.

Once the NetFlow records have been combined into flows, we stored the flows in a format suitable for our analysis. In our case, we imported the flows into a MySQL database. MySQL has been chosen due to the familiarity of the paper's authors with such a tool. The amount of collected NetFlow records stored in MySQL accounted for 30.51 GB, plus 37.28 GB of indexes to speed up our analysis. Once combined, flows accounted for 26.08 GB plus 37.24 GB of indexes.

3.3 The Steps of Our Statistical Analysis

The statistical analysis of the potential network parameters could have been performed in different ways. Simulation tools, for example, could be employed to reproduce a network being measured. A controlled environment of a lab network could be used too. However, none of these methods can 100% capture the real behavior of flows. Considering that, we believe that statistically analyzing data collected from real networks would provide more significant and relevant conclusions.

The initial number of flows considered in our statistical analysis was 378,363,608, which generated a volume of 18.11 TB on Sept 18th, 2008. We started our statistical analysis by defining the set of flows we are focused on. This set of flows is based on the target our self-management of lambda-connections aims at, *i.e.*, the *elephant flows*. We focus our analysis therefore on flows that have the following characteristics: (a) few in number, (b) persistent in time, and (c) represent most of the traffic.

Out of the total number of collected flows (378,363,608), a very small percentage of the flows (0.82%) accounted for the biggest percentage (97%) of the total traffic. This percentage of flows (3,092,885 in numbers), from now on addressed as big flows, matches the characteristics (a) and (c) previously mentioned. Besides, it also confirms the long tail distribution on the network traffic, showing that few flows are responsible for most of the traffic.

Our next step was to check, out of the big flows, the ones persistent in time. Persistent means that they do not have a short duration, as it is the case of bursty flows [6]. Table 1 shows some statistics about the duration of our big flows. It also shows that most of these flows (75%) have a considerable short duration, *i.e.*, a duration shorter than 57 seconds. This allows us to conclude that most of the flows are short-lived.

Conditional Probability

Statistics about flow duration do not say much about the persistence in time of the other 25% of the flows. To have a better insight about the persistence of these flows,

Table 1. Statistics of duration of the big flows

Flows	3,092,885
Mean	274 sec
Median	19 sec
Minimum	0 sec
Maximum	86,507 sec
Percentiles (25%)	12 sec
Percentiles (50%)	19 sec
Percentiles (75%)	57 sec

more specifically about when the majority of them tend to get stable regarding to their duration, we used conditional probability [20]. Conditional probability is the probability of some event A happening given the occurrence of some other event B ($P(A \mid B)$).

In our analysis, we are interested in knowing the probability of flows being persistent in time. For that, we observe the conditional probability of flow duration as follows. Given that the duration (D) of a flow has already lasted at least certain amount of time B ($D >= B$), what is the probability this flow will last for at least one more minute ($D >= B + 60\ sec$)?

Table 2 shows the conditional probability of duration of our big flows expressed in seconds. It shows that there is a small probability (24%) that a flow will last at least one more minute, given the fact it has just started. However, there is a considerable improvement (67%) in this probability when flows have elapsed at least one minute. This probability gets more stable the bigger the minimum flow duration is. This allows us to conclude that the longer a flow has already elapsed, the smaller is its probability of ending in the next time period.

Table 2. Conditional probability of duration of the big flows

$P(D >= B + 60\ sec \mid D >= B)$	*Percentage*
$P(D >= 60\ sec \mid D >= 0\ sec)$	24%
$P(D >= 120\ sec \mid D >= 60\ sec)$	67%
$P(D >= 180\ sec \mid D >= 120\ sec)$	78%
$P(D >= 240\ sec \mid D >= 180\ sec)$	83%
$P(D >= 300\ sec \mid D >= 240\ sec)$	86%
$P(D >= 360\ sec \mid D >= 300\ sec)$	89%
$P(D >= 420\ sec \mid D >= 360\ sec)$	90%
$P(D >= 480\ sec \mid D >= 420\ sec)$	91%
$P(D >= 540\ sec \mid D >= 480\ sec)$	92%
$P(D >= 600\ sec \mid D >= 540\ sec)$	92%

Applying this finding in our self-management approach results in a trade-off between flow duration and decision time. A decision about selecting a flow to be offloaded to the optical level cannot be taken too soon because there is a high probability that the flow is going to last short. In contrast, this decision cannot either be postponed too long because the flow will be consuming resources at the IP level while the decision is not taken. Moreover, there is a slight chance that the flow may end when the decision is finally made.

We choose therefore an elapsed duration of 5 minutes to define a flow as being persistent in time. The reason for that comes from the fact the percentage of flows lasting for at least another minute gets relatively stable (around 90%) when flows reach a minimum duration of 5 minutes. Moreover, in our analysis, flows with duration below 5 minutes did not represent a considerable amount of traffic (26% of the total traffic), whereas flows with duration above or equal to 5 minutes represented 74% of the total traffic.

Based on the outcome of the conditional probability applied on our traces, another filtering was done in the 3,092,885 flows in order to remove flows with a duration shorter than 5 minutes. This resulted in the selection of 283,783 flows (0.07% of the total number of collected flows) having duration above or equal to 5 minutes. These are thus the flows whose metrics throughput (Pps and Bps), packets, and duration are observed in relation to the volume generated.

Correlation and Classification Tree Techniques
Once we defined our set of flows to be analyzed, we observe how those metrics relate with flow volume by drawing correlation charts. These charts provide us a visual impression of what the correlation is, although they are not very quantitative. In order to obtain quantitative correlation values regarding the considered metrics, we used Pearson's correlation method [21]. Pearson's correlation computes the pairwise associations for a set of variables and displays the results in a matrix. This is useful for determining the strength and direction of the association between two metrics. It can be used therefore to measure the linear association between two metrics. In our context, there is an unmistakable intuition that Bps and duration have the strongest correlation to tell about the volume of flows. We use Pearson's correlation, however, to see if this fairly obvious correlation may contain some other unsuspected correlations. Even though we suspected that there are some correlations, we did not know which are the strongest. Applying therefore correlation analysis on our data can lead to a better understanding.

Even though Pearson's correlation is a good method to find out the relationship between two metrics only, it does not consider the combination of more than two of them. It could be therefore that an interaction of more than two metrics could give a better refinement about the best predictors for a flow volume. In order to find that out, we used a classification tree technique widely used in data mining areas, called *CHi-squared Automatic Interaction Detector* (CHAID) method. CHAID is a method that divides a data set into exclusive and comprehensive partitions that differently relate with an observed dependent variable [22]. These partitions are defined by a tree structure and they are classified in descendent order of independent variables, called *predictors*. For each partition of predictors, CHAID assigns a probability of response. All probabilities are subsequently used to rank the partitions with the strongest relation with the dependent variable. It is worth mentioning that the partitions of each predictor are merged if they are not significantly (significance level of 0.05%) different in regard to the dependent variable.

In the case of our analysis, CHAID calculates which independent metrics – duration, packets, Pps, and Bps (the predictors) – have the strongest relation with flow volume (the dependent variable). The CHAID outcome and the results of our correlation analysis are presented in the next section.

4 Results

The potential flow behavior parameters have been statistically evaluated by observing how they individually contribute to identify flows that generate large amounts of data. This section starts by presenting the results regarding our correlation analysis, followed then by the CHAID classification tree.

4.1 Volume *vs.* Considered Metrics

Figures 1, 2, 3, and 4 show the relation of packets, duration, Bps, and Pps with the number of octets (flow volume), respectively. All figures present both axes with logarithmic scale. The figures show that there is a linear relation between packets, Bps, and Pps with octets, but that does not hold in the case of duration. Figure 2 shows that a flow can have a long duration and a small amount of octets, a short duration and a great amount of octets, and all in between. On the other side, the metrics packets, Bps, and Pps walk along with octets in the sense that the bigger those metrics are, the bigger the flow volume is expected to be generated.

It is worth mentioning that there is a certain degree of linearity among those metrics. Figure 1 shows a strong linear relationship between the number of packets and flow volume. Figures 4 and 3, in turn, show a slightly bigger variability in the linearity when compared to packets. The reason for that comes from the fact that flow duration presents a strong variability, which affects Bps and Pps linearity when related to flow volume. For example, we have seen cases in which a lot of packets can be generated in a short amount of time (high Pps), but it is also possible that a small amount of packets can be generated in a longer period (low Pps).

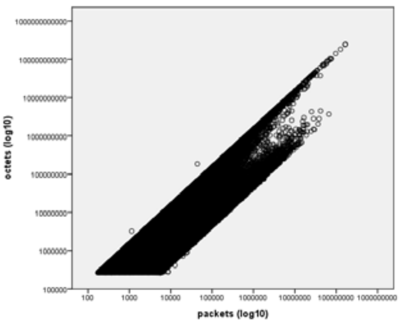


Fig. 1. Correlation between octets and packets

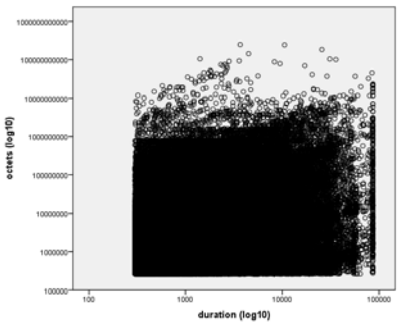


Fig. 2. Correlation between octets and duration

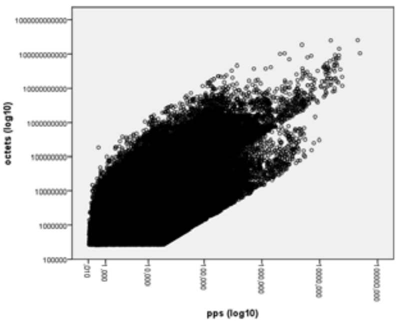


Fig. 3. Correlation between octets and Pps

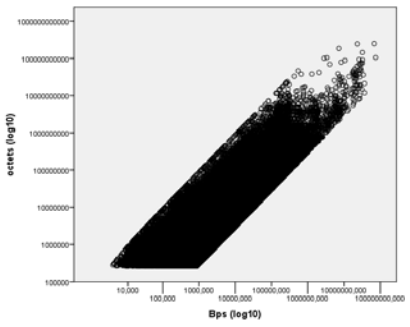


Fig. 4. Correlation between octets and Bps

In order to quantify this linearity, we used then Pearson's *r* correlation, which is defined as the sum of the products of the standard scores of the two measures divided by the degrees of freedom:

$$r = \frac{1}{n-1}\sum_{i=1}^{n}(\frac{X_i - \bar{X}}{s_x})(\frac{Y_i - \bar{Y}}{s_y})$$

where $(\frac{X_i - \bar{X}}{s_x})$, $\bar{X}$, and s_x are the standard score, sample mean, and sample standard deviation, respectively [23]. Table 3 shows the *r* correlation among our considered parameters.

Table 3. Pearson's correlation octets pairwise with other considered parameters

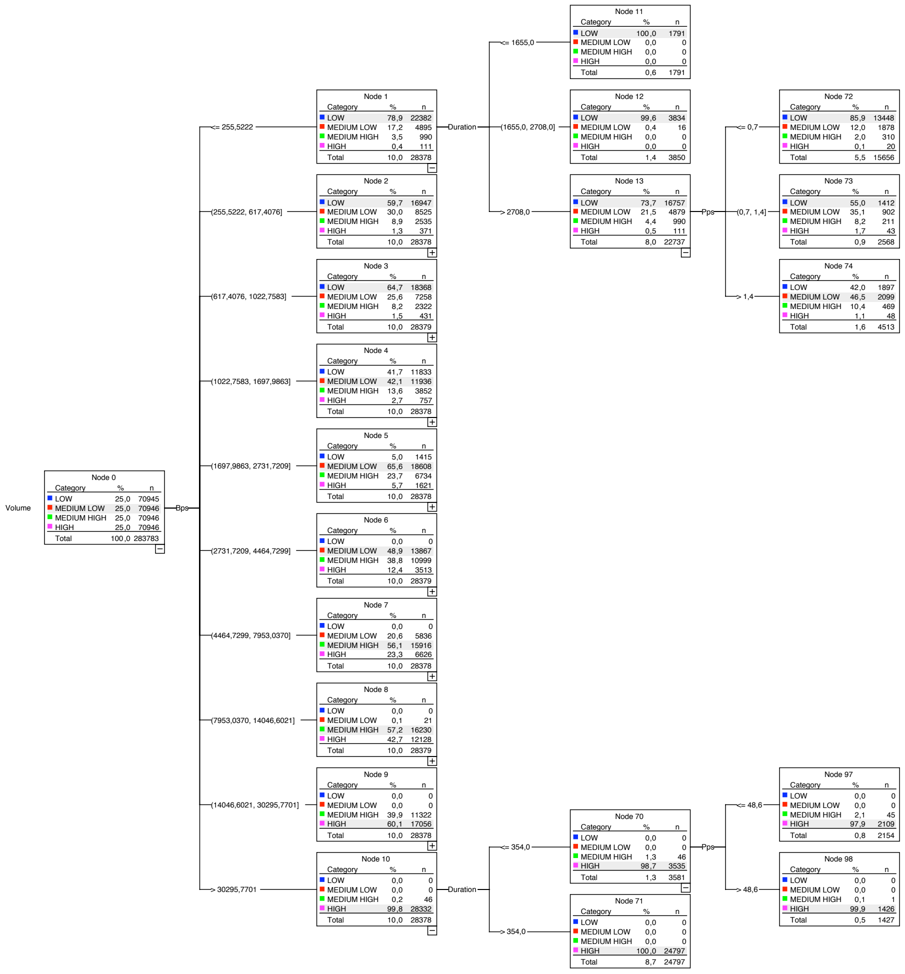
	Packets	Bps	Pps	Duration
Pearson's *r* correlation for octets	0.927	0.671	0.642	0.058

A correlation of 0 (zero) means that there is no linear relationship between the two variables. On the other hand, a correlation of 1 means that there is a strong positive linear relationship between the two variables. As table 3 shows, there is a strong linear relationship between packets (0.927), Bps (0.671), and Pps (0.642), with octets. On the contrary, duration does not present a strong linear relationship (0.058). This means that duration should not be exclusively focused on when trying to predict a flow volume.

Since Pearson's correlation only shows the relation between two metrics, we used CHAID in order to see the relation of more than two metrics with the flow volume. For that, we first divided flow volume into 4 categories according to their size. The 25% biggest flows are referred as *HIGH* volume, whereas the 25% smallest flows are named *LOW* volume. The medium flow sizes (*i.e.*, the remaining 50%) are then divided into the 25% biggest medium flows, referred as *MEDIUM HIGH* volume, whereas the 25% smallest medium flows are named *MEDIUM LOW* volume. Once the flow volume is categorized, we observe how the considered parameters relate to these 4 categories.

Figure 5 partially shows the result of our CHAID classification tree. Only *nodes 1* and *10* were expanded for the sake of space. Flow volume is the dependent variable and it is the *node 0* in the CHAID tree. *Node 0* contains the 4 aforementioned categories for the flow volume. The CHAID method then starts dividing the predictors (*i.e.*, Bps, Pps, duration and packets) into partitions (nodes) and cross-tabulating them against the dependent variable (*node 0*). The predictor that presents the smallest level of significance (*i.e.*, the most statistically significant relationship with the dependent variable) is placed at the first depth of the CHAID classification tree along with its partitions. After the CHAID method has decided about the first level predictor and its best merged partitions, CHAID begins to place other predictors beneath the initial predictor. The CHAID method continues this procedure until further sub-divisions cannot be performed.

CHAID chooses Bps as the best predictor for flow volume as it is ranked right below *node 0*. From *node 1* to *node 10* it is possible to see how each group of flows, classified by their Bps throughput, influences the flow volume. Flows with small Bps throughput tend to be small in flow size (*e.g.*, *node 1*). On the other extreme, flows with big Bps throughput tend to generate large flows (*e.g.*, *node 10*). The second best predictor



Node 0

Category	%	n
🟦 LOW	25,0	70945
🟥 MEDIUM LOW	25,0	70946
🟩 MEDIUM HIGH	25,0	70946
🟪 HIGH	25,0	70946
Total	100,0	283783

Node 1

Category	%	n
🟦 LOW	78,9	22382
🟥 MEDIUM LOW	17,2	4895
🟩 MEDIUM HIGH	3,5	990
🟪 HIGH	0,4	111
Total	10,0	28378

Node 2

Category	%	n
🟦 LOW	59,7	16947
🟥 MEDIUM LOW	30,0	8525
🟩 MEDIUM HIGH	8,9	2535
🟪 HIGH	1,3	371
Total	10,0	28378

Node 3

Category	%	n
🟦 LOW	64,7	18368
🟥 MEDIUM LOW	25,6	7258
🟩 MEDIUM HIGH	8,2	2322
🟪 HIGH	1,5	431
Total	10,0	28379

Node 4

Category	%	n
🟦 LOW	41,7	11833
🟥 MEDIUM LOW	42,1	11936
🟩 MEDIUM HIGH	13,6	3852
🟪 HIGH	2,7	757
Total	10,0	28378

Node 5

Category	%	n
🟦 LOW	5,0	1415
🟥 MEDIUM LOW	65,6	18608
🟩 MEDIUM HIGH	23,7	6734
🟪 HIGH	5,7	1621
Total	10,0	28378

Node 6

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	48,9	13867
🟩 MEDIUM HIGH	38,8	10999
🟪 HIGH	12,4	3513
Total	10,0	28379

Node 7

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	20,6	5836
🟩 MEDIUM HIGH	56,1	15916
🟪 HIGH	23,3	6626
Total	10,0	28378

Node 8

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	0,1	21
🟩 MEDIUM HIGH	57,2	16230
🟪 HIGH	42,7	12128
Total	10,0	28379

Node 9

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	0,0	0
🟩 MEDIUM HIGH	39,9	11322
🟪 HIGH	60,1	17056
Total	10,0	28378

Node 10

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	0,0	0
🟩 MEDIUM HIGH	0,2	46
🟪 HIGH	99,8	28332
Total	10,0	28378

Node 11

Category	%	n
🟦 LOW	100,0	1791
🟥 MEDIUM LOW	0,0	0
🟩 MEDIUM HIGH	0,0	0
🟪 HIGH	0,0	0
Total	0,6	1791

Node 12

Category	%	n
🟦 LOW	99,6	3834
🟥 MEDIUM LOW	0,4	16
🟩 MEDIUM HIGH	0,0	0
🟪 HIGH	0,0	0
Total	1,4	3850

Node 13

Category	%	n
🟦 LOW	73,7	16757
🟥 MEDIUM LOW	21,5	4879
🟩 MEDIUM HIGH	4,4	990
🟪 HIGH	0,5	111
Total	8,0	22737

Node 72

Category	%	n
🟦 LOW	85,9	13448
🟥 MEDIUM LOW	12,0	1878
🟩 MEDIUM HIGH	2,0	310
🟪 HIGH	0,1	20
Total	5,5	15656

Node 73

Category	%	n
🟦 LOW	55,0	1412
🟥 MEDIUM LOW	35,1	902
🟩 MEDIUM HIGH	8,2	211
🟪 HIGH	1,7	43
Total	0,9	2568

Node 74

Category	%	n
🟦 LOW	42,0	1897
🟥 MEDIUM LOW	46,5	2099
🟩 MEDIUM HIGH	10,4	469
🟪 HIGH	1,1	48
Total	1,6	4513

Node 70

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	0,0	0
🟩 MEDIUM HIGH	1,3	46
🟪 HIGH	98,7	3535
Total	1,3	3581

Node 71

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	0,0	0
🟩 MEDIUM HIGH	0,0	0
🟪 HIGH	100,0	24797
Total	8,7	24797

Node 97

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	0,0	0
🟩 MEDIUM HIGH	2,1	45
🟪 HIGH	97,9	2109
Total	0,8	2154

Node 98

Category	%	n
🟦 LOW	0,0	0
🟥 MEDIUM LOW	0,0	0
🟩 MEDIUM HIGH	0,1	1
🟪 HIGH	99,9	1426
Total	0,5	1427

Fig. 5. CHAID classification tree

pointed out by the CHAID method is flow duration. CHAID classification tree shows that flows with high Bps and long durations are those that have the greater flow volume. Finally, for flows with long duration, those with high Pps are responsible for most of the volume generated, being therefore Pps the third best predictor. The number of packets did not show a significant prediction in our model, and it was ignored by the CHAID method.

CHAID also provides a measure of confidence that the classification model is correct as presented in Table 4. Table 4 shows the accuracy of our classification tree model for the 4 observed flow volume categories. It indicates that our classification model correctly classifies about 89% of the flows while misclassifying a flow volume only in about 11% of the cases. That high accuracy allows us to assume that our CHAID classification model correctly selects the proper predictors for the flow volume.

Table 4. The accuracy of our CHAID classification model

Observed	**Predicted**				
	LOW	MEDIUM LOW	MEDIUM HIGH	HIGH	**Accuracy**
LOW	65,499	5,446	0	0	**92.3%**
MEDIUM LOW	6,381	60,334	4,231	0	**85.0%**
MEDIUM HIGH	521	3,016	62,835	4,574	**88.6%**
HIGH	63	48	7,257	63,578	**89.6%**
Overall percentage	*25.5%*	*24.3%*	*26.2%*	*24.0%*	**88.9%**

CHAID statistically confirms the intuition that Bps and duration are the metrics to be considered when observing flow volume. In order of importance, Bps, duration, and Pps (as an optional refinement) are the best predictors. These metrics have more impact on the flow volume than others. Our self-management of hybrid networks should therefore take them into account, rather than other metrics, when taking decisions on moving flows to the optical level.

5 Conclusions and Future Work

This paper presented a statistical evaluation of potential network parameters for our self-management of lambda-connections. Two research questions were risen:

Research question 1: *Which network parameters could be used to predict the volume of flows?* The number of potential network parameters is wide. The selection of the parameters shall be done depending on a defined objective. Since we focus on the prediction of the volume of flows, we narrowed down the selection of network parameters, dividing them into two main groups: flow identifiers parameters, and flow behavior parameters. The former was exhaustedly researched in previous works of ours [17] [18] [19]. The latter group was the focus of this paper and resulted in the evaluation of the parameters: duration, packets, Bps, and Pps.

Research question 2: *How can these parameters be evaluated?* These metrics were evaluated by using statistics methods. We started with conditional probability to know when most of the flows gets stable regarding to their duration. We found out that flows with a minimum duration of 5 minutes have 89% of chance of continuing running for at least the next minute. Our next statistical step was to find out correlation among the considered parameters. Even though initially some metrics such as duration and Bps were intuitively expected to influence flow volume, little was known about how strong this influence could be. Moreover, we were not aware if there were any other unsuspected parameters (*e.g.*, Pps and packets) that could have significant influence to flow volume. To solve this uncertainty, we used first Pearson's r correlation. Pearson's correlation showed that packets (r = 0.927), Bps (r = 0.671), and Pps (r = 0.642) have a strong linear relationship with flow volume, while duration (r = 0.058) has not. Thus, since all parameters have certain influence on the flow volume, they should not be used alone, but in groups. We used then CHAID technique to analyze that. As evaluated by CHAID, Bps and duration are the best predictors for flow volume, followed by Pps.

Even though packets was considered by Pearson's correlation as the parameter with the strongest linear relationship with flow volume, the total number of packets can only be known after the end of a flow, being therefore inadequate to predict flow volume.

As future research, we aim at performing further investigations on Bps and duration parameters. We will investigate how to use them in a tuning process for our decision process to take automatic and online decisions to offload elephant flows.

Acknowledgments. This research work has been supported by the EC IST-EMANICS Network of Excellence (#26854). Special thanks to Roel Hoek (UT) and Anna Sperotto for their valuable contribution in the flow collection process.

References

1. de Laat, C., Radius, E., Wallace, S.: The Rationale of the Current Optical Networking Initiatives. Future Gener. Comput. Syst. 19(6), 999–1008 (2003)
2. Fioreze, T., van de Meent, R., Pras, A.: An Architecture for the Self-management of Lambda-Connections in Hybrid Networks. In: Pras, A., van Sinderen, M. (eds.) EUNICE 2007. LNCS, vol. 4606, pp. 141–148. Springer, Heidelberg (2007)
3. Mori, T., Uchida, M., Kawahara, R., Pan, J., Goto, S.: Identifying Elephant Flows Through Periodically Sampled Packets. In: ACM SIGCOMM, pp. 115–120 (2004)
4. Wallerich, J., Dreger, H., Feldmann, A., Krishnamurthy, B., Willinger, W.: A Methodology For Studying Persistency Aspects of Internet Flows. Computer Communication Review 35(2), 23–36 (2005)
5. Thompson, K., Miller, G.J., Wilder, R.: Wide-Area Internet Traffic Patterns and Characteristics. IEEE Network 11(6), 10–23 (1997)
6. Kim, M.S., Won, Y.J., Hong, J.W.: Characteristic Analysis of Internet Traffic from the Perspective of Flows. Computer Communications 29(10), 1639–1652 (2006)
7. Ribeiro, B., Towsley, D., Ye, T., Bolot, J.C.: Fisher information of sampled packets: an application to flow size estimation. In: IMC 2006: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pp. 15–26. ACM, New York (2006)
8. Raghunarayan, R.: Management Information Base for the Transmission Control Protocol (TCP). RFC 4022 (Proposed Standard) (March 2005)
9. Waldbusser, S.: Remote Network Monitoring Management Information Base Version 2. RFC 4502 (Draft Standard) (May 2006)
10. Waterman, R., Lahaye, B., Romascanu, D., Waldbusser, S.: Remote Network Monitoring MIB Extensions for Switched Networks Version 1.0. RFC 2613 (Draft Standard) (June 1999)
11. Quittek, J., Bryant, S., Claise, B., Aitken, P., Meyer, J.: Information Model for IP Flow Information Export. RFC 5102 (Proposed Standard) (January 2008)
12. Schaaf, K., Broekema, C., Diepen, G., Meijeren, E.: The Lofar Central Processing Facility Architecture. Experimental Astronomy 17(1-3), 43–58 (2004)
13. Estan, C.: Internet Traffic Measurement: What's Going on in my Network? PhD thesis (2003)
14. Brownlee, N., Claffy, K.: Understanding Internet Traffic Streams: Dragonflies and Tortoises. IEEE Communications Magazine 40(10), 110–117 (2002)
15. Soule, A., Salamatia, K., Taft, N., Emilion, R., Papagiannaki, K.: Flow Classification by Histograms: or How to Go on Safari in the Internet. In: SIGMETRICS 2004/Performance 2004: Proceedings of the joint international conference on Measurement and modeling of computer systems, pp. 49–60. ACM, New York (2004)
16. Lan, K., Heidemann, J.: A Measurement Study of Correlations of Internet Flow Characteristics. Computer Networks 50(1), 46–62 (2006)

17. Fioreze, T., Wolbers, M.O., van de Meent, R., Pras, A.: Finding Elephant Flows for Optical Networks. In: Application session proceeding of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007), Munich, Germany, Piscataway, pp. 627–640. IEEE Computer Society Press, Los Alamitos (2007)
18. Fioreze, T., Wolbers, M.O., van de Meent, R., Pras, A.: Offloading IP Flows onto Lambda-Connections. In: Clemm, A., Granville, L.Z., Stadler, R. (eds.) DSOM 2007. LNCS, vol. 4785, pp. 183–186. Springer, Heidelberg (2007)
19. Fioreze, T., Wolbers, M.O., van de Meent, R., Pras, A.: Characterization of IP Flows Eligible for Lambda-Connections in Optical Networks. In: Proceedings of the 11th IEEE/IFIP Network Operations & Management Symposium (NOMS 2008), Salvador, Bahia, Brazil, Piscataway, pp. 256–262. IEEE Computer Society Press, Los Alamitos (2008)
20. Montgomery, D.C., Runger, G.C.: Applied Statistics and Probability for Engineers, 4th edn. Wiley, Chichester (2006)
21. Pearson, K.: Mathematical Contributions to the Theory of Evolution. III. Regression, Heredity and Panmixia. Philosophical Transactions of the Royal Society A 187, 253–318 (1896)
22. Kass, G.V.: An Exploratory Technique for Investigating Large Quantities of Categorical Data. Applied Statistics 29(2), 119–127 (1980)
23. Moore, D.S.: The Basic Practice of Statistics, 4th edn. W. H. Freeman, New York (2006)

Dynamics of Resource Closure Operators

Alva L. Couch and Marc Chiarini

Tufts University, Medford, Massachusetts, USA
couch@cs.tufts.edu, marc.chiarini@tufts.edu

Abstract. We propose a framework for managing resources via convergent operators. Operators represent their need for a resource to a designated *resource closure operator* that manages the resource. We evaluate a specific design for a resource closure operator by simulation and demonstrate that the operator achieves a near-optimal balance between cost and value without using any model of the relationship between resources and behavior. Instead, the resource operator relies upon its control of the resource to perform experiments and react to their results. These experiments allow the operator to be highly adaptive to change and unexpected contingencies.

1 Introduction

A *convergent operator*[1,2] is a process that acts upon a network to achieve a management objective. Each operator has several properties, including:

1. *awareness* (or "knowledge"): The operator has a mechanism by which it can determine if its goals have been achieved.
2. *idempotence* (or "convergence"): If its goals have been achieved and they remain so, the operator effects no changes in the network.

Each operator functions like an independent autonomic control loop: performing monitoring, planning, and system changes at every invocation. The function of a control loop is achieved by repeating one operator over time, either at regular time intervals or by chance[3]. Convergent operators are applied periodically to "immunize" a system against performance degradation[2,1,4]. This basic philosophy is exemplified by the CfEngine system for configuration management[5,6]. Prior work shows that autonomic computing can be "approximated" by a collection of convergent operators applied repeatedly at random[7]. Our operator model is motivated by the "maintenance theorem" of Burgess[3], which demonstrates that autonomous entities can converge to an emergent fixed-point without coordination or coercion.

In this paper, we show that Burgess' maintenance theorem can be applied to resource management, by defining appropriate operators and studying their behavior. This work is a continuation of our prior work on convergent operators[8]. Two convergent operators are *consistent* with one another if probabilistic, repeated application of both of them leads to a network state that achieves both of their management objectives simultaneously, and *inconsistent* otherwise. We demonstrated that consistency is best viewed as a dynamic, emergent property of an operator system, rather than something that can be analyzed statically. We also introduced the idea of a *closure operator* that acts on a system within well-defined limits and is self-healing only within a very narrowly defined domain of responsibility[9]. A closure is a self-managing part of an otherwise open system[10,11,12]. It is called a closure partly because it acts something like a closure in a programming language: it is an island of determinism in a perhaps unpredictable overall system. In this paper, we discuss the design of a specific closure operator for resource management.

This paper is motivated by current challenges to autonomic control models of system management. At Hot Autonomic Computing 2008 (HotAC08), one of the grand challenge problems discussed was that of composing several autonomic control loops with perhaps different objectives[13]. It was accepted that without further constraints, two control loops controlling the same behavioral domain cannot be used at the same time with predictable effects. Thus, a definition of the concept of loop composition remains unclear. Another grand challenge problem was to determine complete and accurate models of server performance. Without these models, traditional control theory (as presented in [14]) is significantly hampered in guaranteeing acceptable response.

Our research responds to these challenges with a new model of resource management that utilizes closures with partial information to replace closed loops with complete information. We describe the design of a closure operator that manages a one-dimensional resource parameter and balances demand (value) against available resources (cost). We demonstrate that an efficient balance between cost and value can be achieved without modeling system dynamics, even in cases where demand functions change suddenly. The result is a highly adaptive management strategy that utilizes minimal information about the environment to maximize its objective function.

The organization of this paper is as follows. We begin by introducing a scalar-value resource-management problem and our architecture and assumptions. We then define our algorithm and demonstrate its properties via simulation. Finally, we discuss limitations and directions for future work.

2 A Scalar-Value Management Problem

Our control model is depicted in Figure 1. A system to be managed outputs one value: a *performance measure* P. It is managed by controlling a *resource parameter* R, which is assumed to (perhaps indirectly) determine P, though there are other determining factors for P, including system load X (which is known to exist but is not considered in our model). P might represent response time, mean time in system, etc., while R might represent memory, server instances, etc. In this study, the goal of management is to control R so that the value of P balances with the cost of R.

This control model is motivated by concerns that arise in the context of Business-Driven Information Technology (BDIM)[15,16]. To truly optimize an

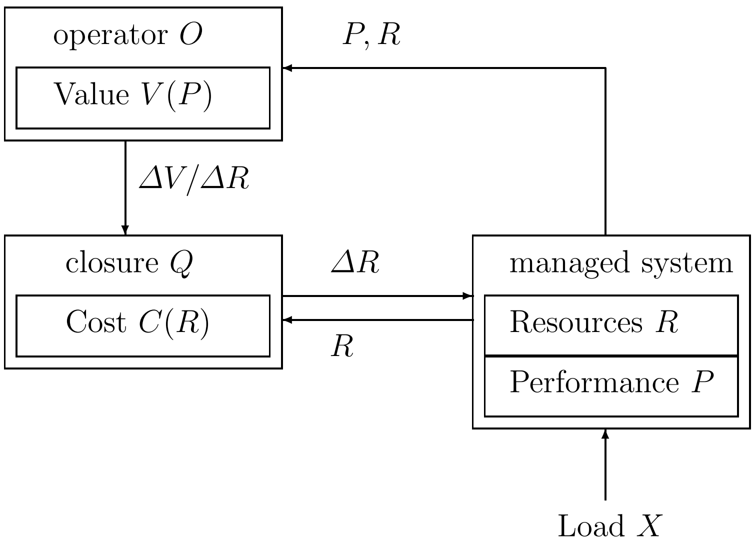


Fig. 1. Operator interaction occurs through a resource closure

enterprise, it is not enough to maximize delivered value; one must also balance cost and value. Thus our objective function is $V - C$, the difference between value V and cost C. We aim to maximize $V - C$ based upon constraints on resources R.

In our model, R is controlled via an intermediary *resource closure operator* Q that accepts recommendations from (distributed) operators O and then recommends changes to be made in R. Each recommendation is a quantity $\Delta V/\Delta R$ that represents the local rate of change in value to the operator in question. Q computes the sum of value estimates from each source to get an overall $\Delta V/\Delta R$ representing the total potential gain from increasing R. The closure then combines this with a local concept of the cost $C(R)$ of the resource to compute a net reward difference $\Delta(V-C)/\Delta R$. If this difference is positive, Q recommends a positive increment ΔR in resources, while if it is negative, a negative ΔR is recommended instead. Q decides upon the magnitude of ΔR independently of the value of $\Delta(V - C)/\Delta R$. For this paper, that magnitude is a constant value that differs only in sign over time.

While cost is a function of available resources ($C = C(R)$), value is a function of performance ($V = V(P)$). We think of P as average response time to queries. P has an "open" and unknown relationship with R. It is not a simple function of R, but rather, depends upon other unknowns (including, e.g., an unknown load X). We might notate P as $P(R, X)$ where X is some unknown vector of load attributes. Our model does not consider these influences: each operator models value inaccurately as a function of P alone. Inaccuracies in the model of P are mitigated via *agility* in both error detection and correction. This is exactly the approach advocated by Burgess[3].

Our model makes a bare minimum of assumptions. It assumes that $P(R, X)$ is simply decreasing in R for constant X, that is, $P(R + \Delta R, X) < P(R, X)$ when $\Delta R > 0$. In other words, more resources improve performance (by decreasing response time). If R can take real values, this implies that P is continuous in R, but R often assumes only a discrete set of numeric values (e.g., integers).

Other than this the model makes no general assumptions about the time-varying behavior of P. It does not assume that P changes predictably in response to changes in X, or even that P remains the same function between time t and time $t + 1$. It might remain the same or it might change.[1]

We have left behind the assumptions of classical control theory. Most applications of control theory require knowledge of $P(R, X)$ itself, and we do not presume such knowledge, because there may be other active management processes that modify the system and thus influence X (and P). We only control one factor influencing P, which is R. We sidestep knowledge of X and replace that knowledge with controlled experimentation in real time. Our overall model embodies an *open world assumption*, while most control theory operates in a *closed world* in which all influences on a system are known in advance.

2.1 Composition

Using our model, composing value estimates from multiple operators is straightforward and trivial, while composing typical autonomic control-loops is considered difficult or perhaps impossible, because of lack of ability to predict interactions between the loops. We have not simplified the problem; our model only moves the difficulty to a different place. The model describes a complex *dynamical system* with seemingly unpredictable behavior. While composition of multiple management loops is made easy by the architecture, "understanding" such a system is made more difficult by its nature. The precise behavior of our systems *cannot* be predicted, and approximate bounds on behavior must suffice as aids to understanding.

2.2 Observability

The assumptions we have made so far seem straightforward but have profound effects. The fact that the performance function can change without notice (e.g., by replacing the whole machine that performs the service) means that information collected far in the past may well describe a *different* P from the present one. To reasonably assure that we are estimating "the current P", only recent information can be safely used, and even that can be incorrect if P has just changed.

Although there is no rule against operators collecting information over long time periods, this information need not be useful under our assumptions. Thus we limit the operators O and Q to using estimates for P and R from the last few

[1] The effectiveness of our operators requires that it does not change very often, but frequency of change is not required to be limited in order to *apply* the strategies.

time steps only (e.g., ten previous operator invocations). While there is clear value in doing so, we do not attempt to detect whether X or the function P itself has changed. The simulations presented below demonstrate that we do not need to detect such changes in order to obtain useful results. In fact, we must limit our observation window to decrease the time in which the model responds to dramatic changes in P.

2.3 Estimating $\Delta V/\Delta R$

The convergent operators O provide some estimate of $\Delta V/\Delta R$ by fitting recent measurements of $V(P)$ to recent measurements of R via a linear model with least-squares fit. The slope of the R coefficient in this model serves as an estimate of $\Delta V/\Delta R$. Each operator has one tunable parameter, which is the amount of history it utilizes in estimating $\Delta V/\Delta R$. If this amount of history is small, it reacts quickly, while large amounts of history make response to sudden changes more gradual, and serve as a kind of noise filter.

This kind of sampling is very different from – and perhaps even contrary to – the kind of sampling currently done in Cfengine. Cfengine attempts to smooth out errors in measurement via moving averages. By contrast, we utilize un-smoothed measurements and expect inaccuracies and noise. The reason our strategy still works is that we filter the results in a different way, inside the resource closure Q.

Note that this is not a trial-and-error approach, nor do we advocate that approach. Our experiments indicate that the validity of this approach depends very much on how Q reacts to information from O. It is best to think of Q as performing controlled experiments. Each change in R is an experiment whose results are available during the next application cycle. Q's goal in this experimentation is to move R nearer to an optimum value, on the assumption that the estimates of $\Delta(V-C)/\Delta R$ become more accurate as the optimum value is approached. If one violates this pattern, the algorithm becomes divergent and of no practical use. We performed many experiments in which Q failed to condition its data sufficiently to allow managing R.

2.4 The Resource Closure

The resource closure Q sums the various $\Delta V/\Delta R$ estimates it receives to get a total estimate. It then subtracts its own estimate of $\Delta C/\Delta R$ to get a total estimate of $\Delta(V-C)/\Delta R$. This is the hypothetical gain in the objective function $V-C$ for unit change in R. If this is positive, the closure increases R by some ΔR, while if it is negative, it decreases R by some ΔR. The magnitude $|\Delta R|$ of the increment is the closure's only tunable parameter. If $|\Delta R|$ is too small, then convergence speed is too slow and the system undershoots behavioral goals, while if it is too large, then it overshoots goals and oscillates around the optimum value while never reaching it.

2.5 Perturbations and Seeding

The operators as defined above need data in order to function, creating a "chicken-and-egg' problem if the system is initially stable. If R is initially held constant, then there will never be enough data to estimate any $\Delta V/\Delta R$, because this requires that R is changing. To solve this problem, the parameter closure Q "seeds" the incoming data for the operators by incrementing the values of R by a fixed ΔR whenever operators seem to be in balance. In this situation, Q increments R (unless R is maximal, in which case it decrements), to create experimental data from which the other operators compute their value functions.

3 Simulation Results

To study the behavior of our proposed system, we wrote a custom simulator in the statistical environment R[17], and explored the design of Q through several simulation experiments. The basis for our experiments is the following simulated system, containing an environment, one management operator, and one closure operator.

The environment has the properties that:

1. $X(t) = 1000 \sin((t/p) * 2\pi) + 1000 + e(0, \sigma)$, time-varying periodic load between 1000 and 2000 with Gaussian error $e(0, \sigma)$.
2. $P(R, X) = X/R$, analogous to response time in a server farm of R servers, with perfect efficiency.

X represents load arriving from outside the environment, while P represents behavior as a function of load X and resources R. The formulae for these properties are not known to any operator, though each operator can observe a small number of pairs P, R to estimate $P(R)$.

There is one operator[2] O, with:

1. $V(P) = 200 - P$ (diminishing returns in proportion to increased response time).
2. w measurements (its *measurement window*), used to compute its least-squares estimate of $\Delta V/\Delta R$.

There is one resource closure Q, with:

1. $C(R) = R$ (analogous to adding same-cost increments of resources).
2. $|\Delta R| = d$ (constant increment magnitude).
3. $R \in [1, 1000]$ (finite resource bound).
4. $R_{\text{initial}} = 50$ (modest amount of resources allocated to reduce startup settling time).

$X(t)$ and $R(t)$ independently vary with time, and cause variation with time in $P(R, X)$, $C(R)$, and $V(P)$.

[2] Since operators compose by adding their outputs, it is sufficient to show that one such function is manageable.

In the above, there are four model parameters we can vary in simulation:

1. σ represents the standard deviation of a Gaussian error term $e(0, \sigma)$, representing a situation in which measurement errors are normally distributed.
2. p represents the period of the periodic load function, which affects the optimal increment d to use in updating R.
3. d represents the constant addend that Q uses to update R.
4. w represents the number of samples in O's estimate of the rate of change of V.

This instance was chosen for study because:

1. It has an easily computable theoretical best value for R, which is $R_{\text{best}}(X) = \sqrt{X}$.
2. The optimal solution does not reach the boundaries of R's range.
3. There is exactly one local maximum so hill-climbing is guaranteed to converge to that maximum.
4. P, V, C, and X are minimally complex but provide interesting behavior[3]

Note that the operator O estimates $\Delta V/\Delta R$ by assuming that there is a linear relationship between V and R, but that the actual relationship, unknown to O, is $V = 200 - P = 200 - X/R$, which is hyperbolic. Our tests indicate that this inaccuracy does not matter at all for our purposes, and we observed no difference between using a linear model and the more accurate hyperbolic model in simulation. Part of the reason is that neither the linear nor the inverse-linear model accounts for X, so that both are *equally* inaccurate.

There are several roles of Q in managing the system. One effect of a fixed $|\Delta R|$ is to quantize the values R can take to a set of finitely many achievable values, thus controlling the data from which $\Delta V/\Delta R$ is computed. Q enforces this set of values, and keeps values within feasible range. Quantization reduces errors in estimates of $\Delta V/\Delta R$ by keeping ΔR from becoming too small.

In early experiments, we tried to increment R by $\Delta R/\Delta(V - C)$ (for one unit improvement in cost), but this showed divergent behavior due to errors in estimating $\Delta(V - C)/\Delta R$. The behavior did not improve when substituting any constant multiple of the addend. Clearly, exposing a parameter to raw increment recommendations was "too risky". Thus we chose to utilize a constant increment instead.

3.1 Simulating the Model

In the simulation, we assume that all operators are invoked at each time step, followed by invoking Q on their results. We compared the performance of the system in each test case to the best-case performance in which the resource function tracks the theoretical optimum (with no noise).

Results of an example simulation are shown in Figure 2. In this simulation, $p = 300$, $d = 1$, $w = 10$, and $\sigma = 0$; in all figures in this paper, the first 200 steps

[3] Note that if V and C are both linear in R, then either the system saturates at one end of R's range, or is in balance for every R if $\Delta V/\Delta R = \Delta C/\Delta R$.

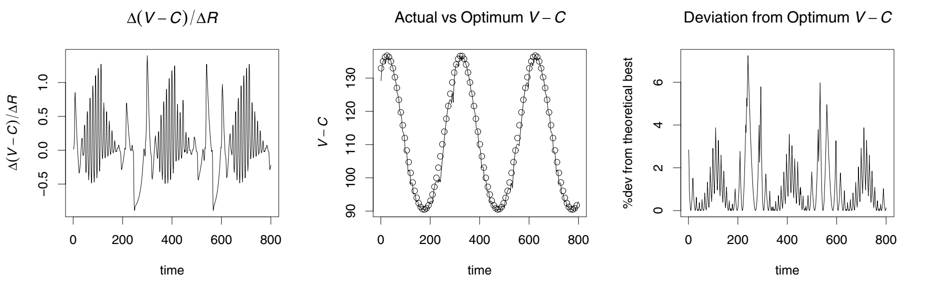


Fig. 2. Input to Q is seemingly unpredictable and random(left), but the output of the system is surprisingly close to the theoretical best performance, shown as circles in the middle plot. The percent difference from theoretical best performance is small (right).

are omitted to allow the system to settle into periodic behavior. On the left is the raw input to Q, expressed as $\Delta(V - C)/\Delta R$. This is often unpredictable, swinging from positive to negative as the naïve estimator of $\Delta V/\Delta C$ makes mistakes due to lack of knowledge of X. These mistakes are mitigated by ignoring the magnitude of $\Delta(V - C)/\Delta R$ and considering only its sign. The agile way in which the system corrects estimation errors allows it to track well the theoretical target (shown in the middle, as a series of circles). The percent error between theoretical and simulated behavior is surprisingly small (right).

The reader might assume incorrectly at this point that Q is merely receiving information from O and acting upon that received information. It is better to think of Q as "constructing experiments" to which O reacts by receiving and forwarding experimental results. While the results of experiments are somewhat random[4], Q's structure assures that the statistical behavior (e.g., how well the system tracks an ideal performance goal) is less random and more predictable.

To understand Q's role in constructing experiments, consider the unexpected downturn with X increasing in Figure 2, at about $t = 250$. The reason the downturn occurs is that X increases quickly enough that $\Delta(V - C)/\Delta R$ becomes negative, suggesting that R should be decreased when in fact it should be increased. The *next* experiment of Q decreases R and tries again, with the result that $\Delta(V - C)/\Delta R$ decreases more quickly. This allows Q to correct its mistake and proceed in the direction of best value, *without becoming aware* that changes in a latent independent variable X were involved in its mistake.

3.2 Effect of Increment Magnitude

The effect of changing the increment d in the simulation is substantial. Consider the effect of $d = 1, 3, 5$ upon the original model (Figure 3). In the figure, the circles represent theoretically optimal behavior. In this experiment, $p = 100$, $w = 10$, and $\sigma = 0$. Note that $d = 1$ causes convergence to be too slow and remain lower than optimal (undershoot), while $d = 5$ causes repeated jumps

[4] Again, due to lack of ability to observe determinism.

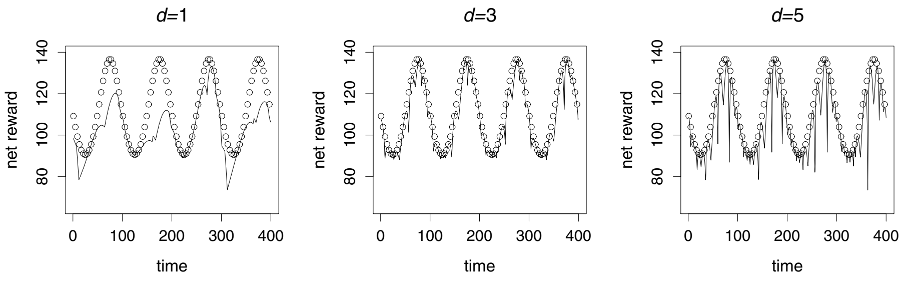


Fig. 3. The effect of different choices for ΔR includes overshoot (right) and lack of tracking (left)

Table 1. Efficiencies of combinations of increment and period indicate that increment is a critical parameter of the algorithm. Efficiencies for the conditions in Figure 3 are shown in bold. For all tests in this table, $\sigma = 0$ and $w = 10$.

period	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$	$d = 8$
50	96.6	92.6	91.1	94.9	93.7	92.3	90.7	86.8
100	**93.4**	97.4	**97.4**	96.6	**95.1**	92.9	90.5	80.4
150	93.4	98.4	97.9	96.9	95.5	92.7	89.7	80.7
200	97.3	98.7	98.2	97.1	95.3	91.8	89.7	81.7
250	98.3	98.9	98.3	97.0	95.3	92.1	89.1	81.1
300	98.9	99.1	98.3	97.1	95.1	91.9	89.2	80.6
350	99.2	99.1	98.2	96.9	95.0	91.8	89.4	79.4
400	99.3	99.2	98.4	96.8	95.0	91.7	89.2	81.0
450	99.5	99.2	98.2	97.0	94.5	91.8	89.3	81.5
500	99.6	99.2	98.3	96.9	94.6	91.8	89.2	80.7

around the optimum value (overshoot). $d = 3$ is approximately appropriate for $p = 100$, which controls how quickly X changes. The best choice for d depends upon the maximum of $\Delta X/\Delta t$ (which is unknown to and not discoverable by Q).

We compared the efficiency of several combinations of increment and period, to understand their relationship (Table 1). The efficiency of each run is defined as the ratio of sums of $V - C$ to the theoretical best $V - C$:

$$E = \frac{\Sigma_i (V_i - C_i)}{\Sigma_i (V_i^{\text{best}} - C_i^{\text{best}})}$$

(which is the same as the ratio of their averages), where $V_i^{\text{best}} - C_i^{\text{best}}$ is the maximum theoretical payoff. The results clearly show that there is a best d for every p in achieving maximum efficiency. We conclude that d is a critical parameter and a good candidate for dynamic tuning.

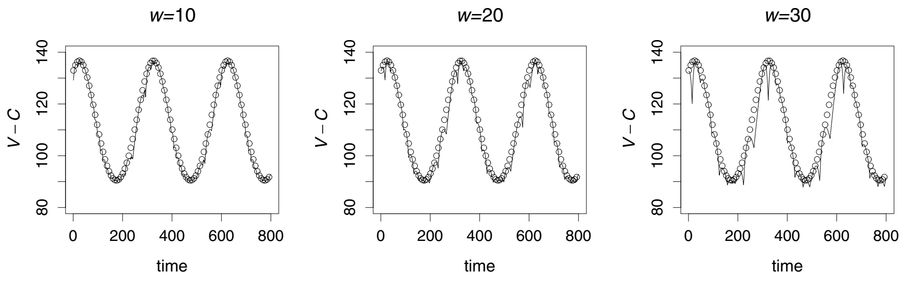


Fig. 4. The effect of increasing window size is to magnify errors in estimation

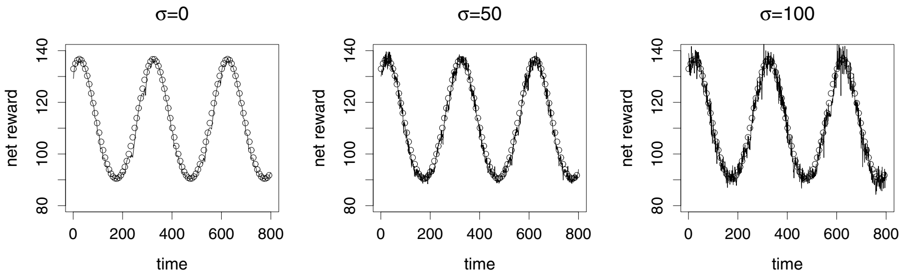


Fig. 5. The effect of greater amounts of noise is to increase convergence time. Notice the increase in oscillations around the optimal as noise increases.

3.3 Effect of Window Size

Our next experiment was to compare the effects of various window sizes $w = 10, 20, 30$ with period $p = 300$, increment size $d = 1$, and noise magnitude $\sigma = 0$ (Figure 4). Note that as window size increases, the tracking of the ideal value is less accurate, and that larger window sizes *magnify errors in estimation* rather than lending accuracy by smoothing. This is the opposite of what one might expect.

To understand this effect, consider that w determines the number of observations involved in estimating value, where at any one time some of these may be in error. Estimating value based upon erroneous data leads to erroneous conclusions, such that a larger w provides the potential for each erroneous observation to have a greater impact.

3.4 Effect of Noise

Noise has the effect of making the estimates $\Delta V/\Delta R$ inaccurate, and thus affects convergence time. We simulated the system described above with varying values of σ. Figure 5 exhibits three such experiments, corresponding to $\sigma = 0, 50, 100$. In this experiment, $p = 300$, $d = 1$, and $w = 10$. The circles represent optimum performance in the absence of noise.

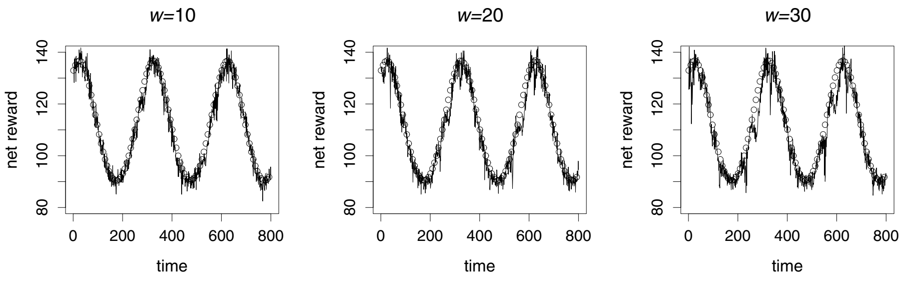


Fig. 6. The effect of different sizes for an operator's history window. A value of w that is too large interacts with the inaccuracy of the model to make overall performance less responsive.

3.5 Effect of Window on Noise

We next repeated the experiment in Section 3.3 but added a constant magnitude of Gaussian noise. We thought that larger windows would be beneficial, but actually smaller is always better (Figure 6). In the figure, noise magnitude $\sigma = 100$, period $p = 300$, increment size $d = 1$, and window $w = 10, 20, 30$.

The efficiencies of several variations are shown in Table 2. Again, the efficiency of each run is the ratio of the sum of payoffs over time to the sum of best payoffs. The first row contains data for zero noise. Larger windows always lead to worse efficiency. Thus we conclude that small values of w (between 3 and 10) are appropriate.

Table 2. Efficiencies of combinations of window and noise show counter-intuitively that response to noise in input is not improved by sampling in larger windows. Efficiencies for Figures 4 and 6 are shown in bold. For all tests in this table, $p = 300$ and $d = 1$.

σ	$w = 5$	$w = 10$	$w = 15$	$w = 20$	$w = 25$	$w = 30$
0	99.1	**98.9**	98.7	**98.5**	98.1	**97.6**
25	99.3	98.9	98.7	98.5	98.0	97.6
50	99.4	99.0	98.7	98.5	98.0	97.6
75	99.3	98.9	98.6	98.1	98.0	97.5
100	98.9	**98.8**	98.5	**98.2**	97.6	**97.6**
125	99.0	99.1	98.5	98.2	97.7	97.5
150	99.1	99.2	98.5	98.1	97.9	97.6

4 Limitations

Our algorithm has several theoretical limitations to its applicability. It is a hill-climbing algorithm that will always converge to a *local* maximum value, but is not guaranteed to converge to a *global* maximum value unless there is only one global maximum. It always seeks a global maximum only if there is exactly one local maximum for $\Sigma_i(V_i) - C$, which is true only for the values of R for which the finite difference $\Delta(\Sigma_i V_i - C)/\Delta R$ (which is a function of R) is 0. We must use difference equations to describe this system because derivatives are not always meaningful; the legal values of R can be discrete.

By the assumptions above, the finite differences between values of $\Sigma_i V_i$ and C for values of R near its current value are always strictly greater than 0; we notate these as $\Delta(\Sigma_i V_i)/\Delta R$ and $\Delta C/\Delta R$. Because of this, the combined difference function $\Delta(\Sigma_i V_i - C)/\Delta R$ is zero (the system is in balance) only when $\Delta(\Sigma_i V_i)/\Delta R$ and $\Delta C/\Delta R$ are equal. The only way to assure that there is only one local maximum is for $\Delta(\Sigma_i V_i)/\Delta R$ to equal $\Delta C/\Delta R$ for at most one value of R.[5]

There are only two ways in which this can happen: by a down-crossing, during which value initially changes faster than cost with respect to R and then proceeds to change more slowly than cost, or by the opposite event of an up-crossing. Any value difference function that crosses the cost difference function in the opposite direction from another has the potential to add an extra local maximum. The difference function of the sum of value difference functions that all cross the cost difference function in the same direction also crosses in that direction and is guaranteed to have a unique stable point.

The algorithm also only works if the V_i are never constant over an interval of P, and P is never constant over any interval of R. In other words, the system handles what might be called "highly dynamic" situations, but stops at a non-optimum value of R if V remains constant near that value.

Thus the assumptions we outlined at the beginning are necessary rather than just sufficient, which can make it difficult to design appropriate cost and value functions for practical situations. We believe that handling value functions with regions of constant value requires a different approach, which we plan to discuss in later work.

5 Future Work

Several open issues remain for future work.

First, the algorithms for adjusting d optimally have not been determined, and would be required in order to create a practical implementation. Our simulation data suggests possible solutions, but these have not been evaluated fully.

Second, the efficiency calculations we utilized to evaluate the algorithm are not available if the solution is deployed in the real world. Some way to estimate the efficiency of a real-world solution is needed.

Third, the system is easily generalized to the case in which R is a vector rather than a scalar. X can become a vector without modifying anything in the basic model, and different operators can even have varying *concepts* of X. If R is a vector, then $\Delta V/\Delta R$ becomes a gradient with respect to the components of R, and the parameter closure determines a *direction* in which to move one unit (ΔR) in a quantized R-space rather than a sign of movement in one-space. Much will be determined by exactly how we *quantize* R-space, and this design problem seems difficult at present.

[5] If these are never equal, then the maximum is either the highest or lowest value of R.

Fourth, there is an intimate relationship between this operator theory and Burgess's promise theory(as discussed in[7]) that is yet to be fully understood. A promise is a non-binding statement of intent. The value judgments communicated to Q are promises to Q, but whether and how one could reason about that information remains unclear, especially if there is coupling between services and if agents might be untruthful about parameter values to their private advantage.

Fifth, this model only handles simply increasing functions. Step-functions and functions with constant-value regions are common in service-level agreements, and there are ways of transforming any such function into a function conforming to our model. We have not studied this in enough detail to comment, though some techniques for approximating step functions with continuous functions seem relevant.

6 Conclusions

We have demonstrated that a model for managing the resources available to a localized service can base decisions on comparing local cost with distributed (and possibly disparate) concepts of value. The model converges to the balance point between value and cost without utilizing any knowledge of the relationship between performance and resources. Instead, success is based upon physical invariants of the environment (e.g., persistence of conditions) that allow the closure operator to statistically discover near-optimal resource levels through repeated experiment. The model is very flexible and requires a minimum of assumptions. One can add and remove at will, operators with different concepts of value. The performance function P can change unexpectedly with quick adaptation to a new optimum. There is a predictable relationship between the rate at which load varies and the best increment to be utilized by Q, although long windows of observation seem to hurt performance in all cases.

This paper centers upon a very simple result, but it opens a Pandora's box of possibilities. It is possible to use distributed concepts of demand and reasoning in concert with a centralized concept of service, and make sense of the global optimum. Convergence of this approach depends only upon the ability of management software to perform controlled experiments, and not upon any model assumptions. This is a first step toward distributed management with no centralized authority.

References

1. Burgess, M.: Configurable immunity for evolving human-computer systems. Science of Computer Programming 51, 197 (2004)
2. Burgess, M.: Computer immunology. In: Proceedings of the Twelth Systems Administration Conference (LISA XII), USENIX Association: Berkeley, CA, p. 283 (1998)
3. Burgess, M.: On the theory of system administration. Science of Computer Programming 49, 1 (2003)

4. Burgess, M.: Cfengine as a component of computer immune-systems. In: Proceedings of the Norwegian Conference on Informatics (1998)
5. Burgess, M.: A site configuration engine. Computing Systems 8(2), 309–337 (1995)
6. Burgess, M., Ralston, R.: Distributed resource administration using cfengine. Softw., Pract. Exper. 27(9), 1083–1101 (1997)
7. Burgess, M., Couch, A.: Autonomic computing approximated by fixed-point promises. In: Proceedings of the First IEEE International Workshop on Modeling Autonomic Communication Environments (MACE), pp. 197–222. Multicon Verlag (2006)
8. Couch, A.L., Chiarini, M.: Dynamic consistency analysis for convergent operators. In: Hausheer, D., Schönwälder, J. (eds.) AIMS 2008. LNCS, vol. 5127, pp. 148–161. Springer, Heidelberg (2008)
9. Couch, A.L., Chiarini, M.: A theory of closure operators. In: Hausheer, D., Schönwälder, J. (eds.) AIMS 2008. LNCS, vol. 5127, pp. 162–174. Springer, Heidelberg (2008)
10. Couch, A., Hart, J., Idhaw, E.G., Kallas, D.: Seeking closure in an open world: A behavioral agent approach to configuration management. In: LISA 2003: Proceedings of the 17th USENIX Conference on System Administration, Berkeley, CA, USA, pp. 125–148. USENIX (2003)
11. Schwartzberg, S., Couch, A.: Experience implementing a web service closure. In: LISA 2004: Proceedings of the 18th USENIX Conference on System Administration, Berkeley, CA, USA, pp. 213–230. USENIX (2004)
12. Wu, N., Couch, A.: Experience implementing an ip address closure. In: LISA 2006: Proceedings of the 20th USENIX Conference on System Administration, Berkeley, CA, USA, pp. 119–130. USENIX (2006)
13. Couch, A.: Summary of the third workshop on hot topics in autonomic computing (HotAC III). Login: Magazine 33(5), 112–113 (2008)
14. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. John Wiley & Sons, Chichester (2004)
15. Sauve, J., Moura, A., Sampaio, M., Jornada, J., Radziuk, E.: An introductory overview and survey of Business-Driven IT management. In: The First IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM), pp. 1–10 (2006)
16. Moura, A., Sauve, J., Bartolini, C.: Research challenges of Business-Driven IT management. In: The Second IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM), pp. 19–28 (2007)
17. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2008) ISBN 3-900051-07-0

Churn Tolerance Improvement Techniques in an Algorithm-Neutral DHT

Kazuyuki Shudo

Tokyo Institute of Technology,
2-12-1-W8-43 Ookayama, Meguro, Tokyo, 152-8552 Japan
shudo@computer.org

Abstract. Churn resilience is an important topic in DHT research. In this paper, I present techniques to improve churn resilience and their effects. All the techniques can be implemented in a DHT layer and require no change to underlying routing layers. In other words, they do not depend on a specific routing algorithm and can work with various algorithms.

Keywords: Overlay network, DHT, structured overlay, churn, emulation.

1 Introduction

Distributed hash tables (DHTs) are distributed systems, by which autonomous nodes provide a single hash table without center servers. There are extensive applications of a DHT because a hash table is a general purpose mechanism and can be applied to various kinds of name resolution.

Decentralized peer-to-peer systems such as a DHT provide the following merits to service providers utilizing it.

- Lower management and providing cost
- Higher scalability
- Higher reliability

If we choose an approach to improve reliability of each node in the same way as using a small number of servers, management cost rises much in proportion to the number of servers and it spoils the merits. Oppositely, it reduces management cost to use clients' computers to provide a service, but we cannot count on them to be reliable and highly-available.

To be reliable and cost-effective enough, a decentralized distributed system has to tolerate churn, the continuous process of node joining and leaving, and keep providing its service with churn.

In this paper, I present churn tolerance techniques for DHTs. Those techniques are independent of underlying routing algorithms. All techniques target the DHT layer and they require no change to underlying routing layers. The independence enables the techniques to work with various routing algorithms, Chord, Kademlia and others, which have their own advantages. The contribution of this paper is a demonstration that those techniques work with various algorithms effectively.

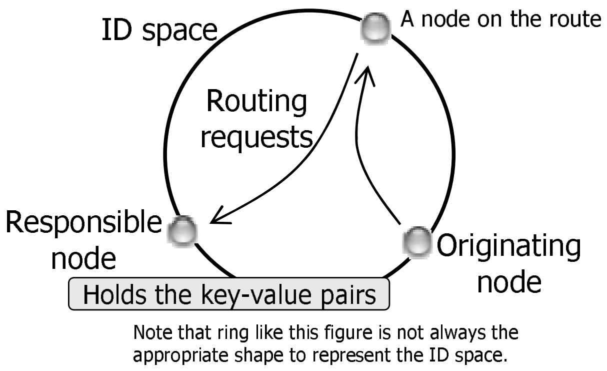

ID space
A node on the route
Routing requests
Responsible node
Originating node
Holds the key-value pairs
Note that ring like this figure is not always the appropriate shape to represent the ID space.

Fig. 1. Routing to put and get to a DHT

The churn tolerance techniques have been implemented in Overlay Weaver [1,2], an implementation of structured overlay, which provides multiple routing algorithms. Section 4 presents the effects of the techniques.

2 Churn Problem in a DHT

Put and get processes in a DHT are described as follows supposing an abstraction in which a DHT is an application of structured overlay (Figure 1) [3].

put(key,value). A node finds a responsible node by routing with the specified key, and transfers the key-value pair to the responsible node.

get(key). A node finds a responsible node by routing with the specified key, requests values associated with the key from the responsible node, and receives them.

It is possible for the get process to fail on condition that nodes in a DHT have joined and left. It is the churn problem in a DHT. Direct causes of the failure are as follows.

1. The key-value pair disappeared after put (because the responsible node departed).
2. The responsible node does not hold the key-value pairs.
   (a) A newly joined node became the responsible node for the key.
   (b) The routing for the put operation did not reach the suitable responsible node because of incomplete routing tables and other reasons.
3. Nodes in a route left during the routing. (This happens only in recursive routing [4].)

For instance, we cannot retrieve a key-value pair if a responsible node for the key departed after put. But the key-value pair remains on the DHT if it has been replicated to other nodes in advance of the departure. Section 3 presents these sorts of techniques to improve churn tolerance.

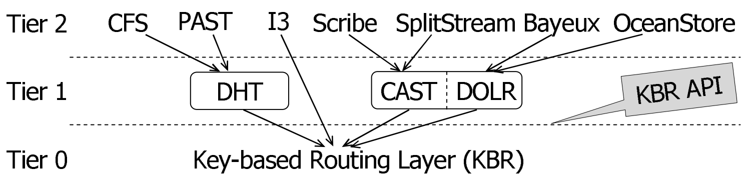


Fig. 2. Key-based routing (KBR) (Figure 1 from Dabek et al.[3])

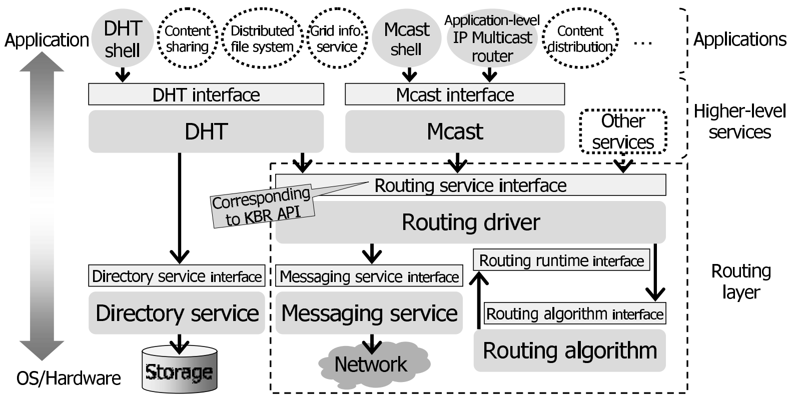


Fig. 3. Components organizing runtime of Overlay Weaver

3 Churn Tolerance Techniques

In this section, I describe techniques to improve churn tolerance of a DHT. I have implemented all the techniques in a DHT implementation of an overlay construction toolkit Overlay Weaver [1,2], and measured their effects.

Overlay Weaver follows an abstraction of structured overlay proposed by Dabek et al. [3], in which higher level services such as DHT and multicast are built on the basic routing layer (Figure 2). This model enables us to combine arbitrary implementations of each layer and Overlay Weaver demonstrated it. For example, we can choose and combine a suitable implementation of a routing algorithm for an application with a DHT implementation.

If the churn tolerance techniques are implemented only in the DHT layer, they are independent of underlying routing algorithms and work with various algorithms. This flexibility is a nature of churn tolerance techniques. The techniques are apt to refer to internal structures of a routing algorithm (e.g. leaf set in Pastry) and find contacts with other nodes. The techniques can be independent of a routing algorithm by accessing the routing layer only through a carefully-designed and algorithm-neutral interface.

3.1 Routing Layer API

In the abstraction proposed by Dabek et al., the routing layer provides a function supporting replication to upper layers including DHT. The function is `replicaSet(key,max_rank)`. It returns an ordered list of nodes which are candidates for the responsible node for the specified key. The order reflects how are the nodes adequate to be the responsible node and have a replica. In other words, after the first node departs, the next node is most adequate for the responsible node.

Overlay Weaver provides a similar mechanism to support churn tolerance techniques. A routing process results in the list of candidates for the responsible node, not just one responsible node. This mechanism is different from Dabek's one which is a function called locally. It serves a number of techniques not only replication. Additionally, Dabek's proposal does not include an empirical proof (Section 6), but this paper demonstrates efficiency of algorithm-neutral churn tolerance techniques.

Within the routing layer (Figure 3), the Routing algorithm component provides a function `rootCandidates(ID target,int maxNumber` to a Routing driver component. The function is similar to Dabek's `replicaSet` function.

Following sections present these four techniques.

- Replication
- Join-time transfer
- Multiple get
- Repeated implicit put

First three techniques utilize the list of responsible node candidates.

3.2 Replication

In a put process, not only the responsible node but also several nodes receive and hold the key-value pair which has been put. Even if the responsible node left after the put, later get requests can complete by obtaining the requested values from the other nodes.

The key-value pair is replicated to several nodes along the list of responsible node candidates, which is obtained as a result of routing. By this replication, even if the responsible node left, later routing reaches the next candidate, which returns the requested value, and get requests complete.

This is not the only method to select nodes on which a key-value pair is replicated. An example of other methods is generation of multiple keys. For example, a put process can generate another three keys based on the specified key by exclusive-or'ing 01, 10 and 11 to the first 2 bits of the key: $key \oplus 010..$(binary), $key \oplus 100..$, $key \oplus 110..$. But I have not adopted this method because it requires a larger amount of messages between nodes. Replication on responsible node candidates requires routing just once, but this method involve multiple routing the same times as the number of replicas.

Our implementation of replication allows both the node originated the put request and the responsible node to initiate replication. The number of messages sent to make replicas is identical in both cases, but efficiency of those messaging differ if node IDs reflect network proximity. The responsible node is better if the distance between node IDs in the routing algorithm reflects network proximity with proximity identifier selection (PIS) [5]. The efficiency is identical in both cases because our implementation does not take account of network proximity currently.

The implementation of replication has the two following parameters.

- The number of replicas
- Performing node: the node originated the put request or the responsible node

Only the responsible node hold the key-value pair if one is specified as the number of replicas.

Note that we should distinguish replication from caching. A cache is a key-value pair preserved by a node in an earlier put or get process in which the node is involved. It is for the purpose of performance improvement, not for churn tolerance though it contributes to churn tolerance unexpectedly. However, it is possible to cache a key-value pair with the intention of utilizing it also as a replica.

Replica consistency. The DHT implementation of Overlay Weaver, on which our research is based, preserves eventual consistency [6,7] at most even with the churn tolerance techniques in this paper.

One of natural concerns about consistency of replicas is different values associated with the same key according to nodes. It happens in cases that an old value to be overridden remains and almost simultaneous put requests compete.

But these are not the cases in our DHT implementation because multiple values can be associated with a single key and no value is removed by a put request. This is the same policy as a DHT implementation Bamboo [8] and its deployment, OpenDHT [6]. A get request yield all values associated with the specified key.

However, it is possible for a get request to result in an obsolete value which has been removed if a remove operation failed on some replicas. Resolution for such inconsistency is left for an application which uses the DHT. Vector clock in Dynamo [7] is a promising support to resolve such inconsistency even though the current implementation does not provide it.

3.3 Join-Time Transfer

It is possible for a newly joining node to be the most proper responsible node for existing key-value pairs. In this case, a node holding those pairs transfers them to the newly joining node. This technique is called join-time transfer in this paper. With this technique, the new responsible node can respond to later get requests.

Table 1. Causes of get failure each technique treats

	Replication	Join-time transfer	Multiple get	Repeated implicit put
• Key-value pairs disappeared	√			√
• A responsible node does not hold the key-value pairs:				(requires replication)
· A joined node became a responsible node		√	√	√
· A joined node became a responsible node		√	√	√
• Nodes in a route left				

In our implementation, a joining node asks a few nodes which are expected to hold key-value pairs the joining node should have. The asked nodes are responsible node candidates (Section 3.1) for the node ID of the joining node. Routing to join yields the list of the candidates and the joining node asks the specified number of the candidates in order as in the list. An asked node transfers key-value pairs which the joining node is more proper to have and the joining node holds them. Note that the transferring node does not expressly discard those pairs though the node can do it and keeps them.

The only parameter of this technique is the number of nodes a joining node asks.

3.4 Multiple Get

In a get process, the requesting node can ask values from multiple nodes, not only the responsible node for the key. The requested nodes are responsible node candidates the same as replication. With this technique, if a get request is routed to a newly joined node which does not have the value, an old responsible node is possible to be also asked and return the value.

A former mentioned technique, join-time transfer, also mitigates the problem where a newly joined node became a responsible node. Effects of the two techniques overlaps but are not identical.

Multiple get compensates suboptimal routing for putting. A routing can reach a node other than the responsible node for the specified key when routing tables are incomplete. In this case a suboptimal node holds the key-value pair, but later get requests can ask the holding node with the multiple get technique.

The number of asked nodes is the parameter of the multiple get technique. Only one responsible node receives a get request in case the parameter is 1.

3.5 Repeated Implicit Put

Each node composing a DHT puts key-value pairs it holds periodically and autonomically without explicit put requests. The implicit put process also makes replicas as an usually-requested put process.

The number of replicas gets fewer according to nodes leaving even though join-time transfer supplements replicas. The purpose of this technique, repeated implicit put, is supplementation of replicas.

This technique has the same effect as the join-time transfer. A node which joined after a key-value pair was put can receive it if the node is responsible for the key. But its effect is limited as it cannot save a get request issued before an implicit put process runs. There is an interval between implicit put processes.

This technique looks useless in case the number of replicas is 1. It is not correct. Even in the case, the technique has the same effect as join-time transfer and can transfers key-value pairs to the proper responsible node.

The parameter of this technique is an interval of implicit put processes. A node waits for the specified time between the processes. Actual intervals are fluctuated a little with random numbers and the fluctuation prevents synchronized behavior of many nodes, which put a much load on the overlay.

Only this technique does not refer to a list of responsible node candidates (Section 3.1) though other three techniques presented in this paper use the list.

3.6 Targets of Each Technique

Table 1 illustrates causes of get failure, listed in Section 2, which each technique treats.

Replication prevents disappearance of key-value pairs which happens according to node leaving. Repeated implicit put supplements replicas, however the technique does not take the effect if the number of replicas is 1.

Both join-time transfer and multiple get treat a problematic situation where a node does not hold key-value pairs which the node is responsible for. The former one prevents the situation and the latter one enables a get request to succeed in the situation.

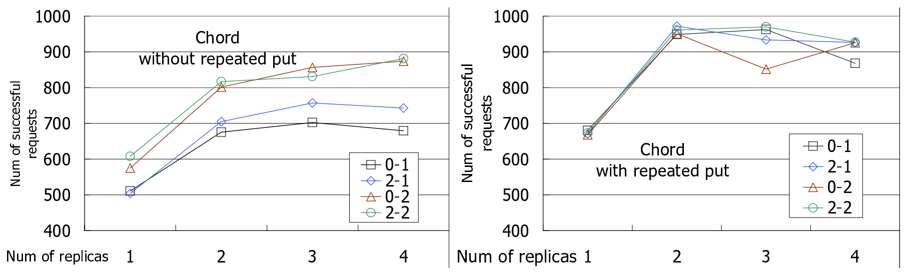
Repeated implicit put is also possible to save the situation (Section 3.5), but it takes effect intermittently.

This paper presents no solution to the last cause in which nodes on the route leaves during a recursive routing. Concurrent multiple routing processes can mitigate this problem.

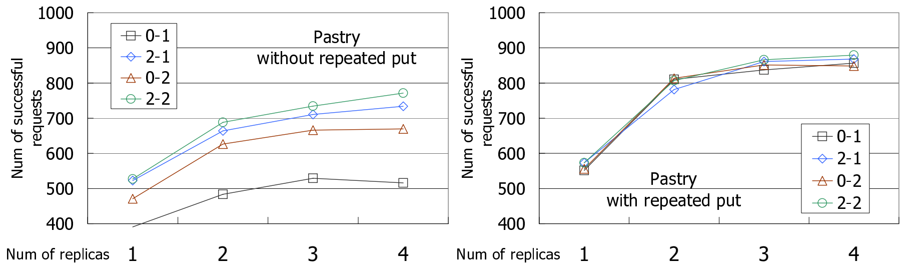
4 Effect of Techniques

This section demonstrates the effects of the techniques presented in Section 3. In an experiment, we ran 1000 nodes on a single computer, issued a number of get requests, measured the number of successful requests, and calculated the success rate.

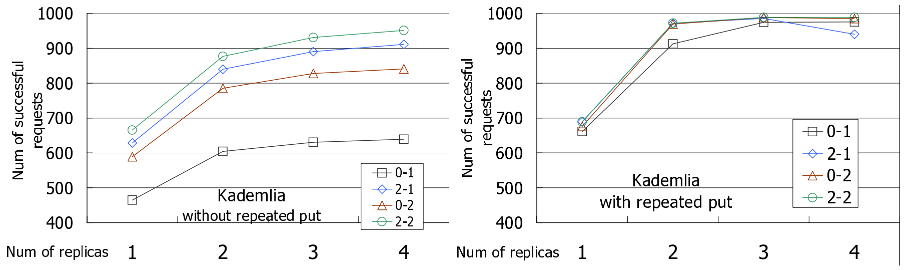
Those nodes ran on a Distributed Environment Emulator, which Overlay Weaver provides. It hosts a large number of nodes and controls them along a given emulation scenario. An Emulator hosts an application which can work on a real network without any modification. The Emulator provides a lightweight



(a) Chord



(b) Pastry



(c) Kademlia

Fig. 4. Effects of presented techniques

Messaging service which bypasses a TCP/IP protocol stack and lightens the communication process between hosted nodes though those nodes can use TCP/IP if we want.

The Messaging service delivers a message as fast as the hosting computer can. It does not copy a message body even in memory and then the bandwidth between nodes is infinity in theory. It models an ideal communication medium effectively and not limited by a physical medium. Churn is the only cause of failure of a get request because no message is lost. In other words, the communication medium, the in-memory Messaging service takes no effect on the success rate.

4.1 Conditions

We generated a churn scenario with a generating tool and gave the scenario to hosted 1000 nodes. All the following experiments were made with the identical scenario.

The scenario is as follows.

1. Starts 1000 nodes.
2. Lets all nodes join a DHT, one every 0.15 seconds.
3. Puts different 1000 key-value pairs on the DHT, one every 0.2 seconds.
4. Gets all 1000 key-value pairs from the DHT, one every 0.2 seconds.

Nodes to which put and get were determined at random when the scenario is generated.

Churn lasts from the start of put to the end of get. The scenario keeps the number of nodes as 1000 by letting a node leave and another node join immediately. This model of churn is identical to experiments by Rhea et al. [9], and different from the churn model adopted by Kato et al. [10] In the latter model, the number of nodes varies because another node does not join immediately after a node left.

Churn is timed by a Poisson process and the frequency of it is 2 times a second. Thus the average life time of a node is $1000(nodes)/2(times/sec) = 500 seconds$.

Communication timeout is 3 seconds and routing timeout is 10 seconds. Because of it, a put or get request timeouts and fails if tries to contact a failed node 4 times. A node removes a failed node from its routing table.

We used a PC with 2.8 GHz Pentium D processor, Linux 2.6.21 for x86-64 and HotSpot Server VM of Java 2 SE 5.0 Update 12. The version of Overlay Weaver is 0.6.4. All experiments were made 6 times and we adopted the average of middle 4 values as the result.

4.2 Results

Figure 4 shows the results. We conducted experiments with all combinations of all routing algorithms Overlay Weaver provides and two routing styles, iterative and recursive routing. While Overlay Weaver supports various routing algorithms including Chord, Kademlia, Koorde, Pastry, Tapestry and their variations, Figure 4 shows results with representative ones, Chord, Pastry and Kademlia, and the routing style is iterative routing.

The vertical axis indicates the number of successful get requests out of 1000 requests. Numbers closer to 1000 are better. The horizontal axis indicates the number of replicas from 1 to 4. Each algorithm has two graphs. Graphs on the left side shows the results with repeated implicit put enabled, and disabled in the right-hand graphs. The interval of the implicit put processes is 30 seconds.

Four lines in a graph differ in parameters of two techniques, join-time transfer and multiple get. The parameter of the former technique is the number of nodes to which a newly joined node asks. It is 0 (disabled) or 2. The parameter of the latter technique is the number of target nodes of get requests. It is 1 (disabled) or 2. Legends such as "<number>-<number>" in the graphs mean these two parameters. A couple of numbers means "<join-time transfer>-<multiple get>".

4.3 Observations

We observed the following facts in Figure 4.

- A larger number of replicas resulted in a better success rate.
- Both join-time transfer and multiple get improved success rates.
- In Pastry and Kademlia, join-time transfer (with the parameter 2) was more effective than multiple get (with the parameter 2). Oppositely, in Chord, multiple get was more effective.
- Repeated implicit put filled up the gaps between the results with different parameters of join-time transfer and multiple get. Four lines in the right graphs are very close.
- Replication with 3 replicas showed better results than 4 replicas in a few cases.

These trends were also observed in the results with recursive routing. Tapestry, another routing algorithm, shows the same trend as Pastry as expected. They share an important part of their routing algorithm and this result looks natural.

Part of get requests failed even though the churn tolerance techniques were applied. The reasons of those failure are as follows.

- The techniques and the parameters were not enough to compensate such degree of churn completely.
  For example, all replicas disappeared or a get request was issued before the responsible node received a replica by implicit put.
- Routing requests did not reach the responsible node because of incomplete routing tables.
- Timeouts happened many times in communication and a routing could not finish (Section 4.1).

Note that the results shown in this section do not support relative superiority of each routing algorithm. Each algorithm has its specific parameters and they have an effect on churn tolerance. In this paper, all parameters of routing algorithms were the default values of Overlay Weaver 0.6.4 and were not adjusted.

This section could demonstrate that algorithm-neutral churn tolerance techniques take effect. The next problem is which technique and what parameter we choose. The decision should be based on cost performance, not only performance. In the next section, I discuss cost performance of churn tolerance techniques.

5 Cost Performance of Churn Tolerance Techniques

In this section, I discuss how to calculate cost performance of churn tolerance techniques. It is rational that the effect is represented by a function which takes the success rate of get requests as an input, because the effect appears as the success rate.

Next, let me consider the cost. There are various kinds of cost of the churn tolerance techniques as follows.

- Traffic and the number of times of communication
- Time required to join, put and get
- Memory and storage consumption
- Processing time

Preceding work [9,10] focused on communication traffic and time to complete get operations. Other kinds of cost have drawn less attention and one of the reasons for it is that the other resources are seldom a bottleneck on today's Internet.

Nevertheless, replication with n replicas consumes n times larger storage or memory. It is fairly expensive with a large amount of data or on an embedded node with small resources. Therefore it is necessary to premise an application and an applied environment when considering the cost.

Table 2. DHT processes in which each technique is involved

	join	put	get	ordinary
Replication		√		
Join-time transfer	√			
Multiple get			√	
Repeated implicit put				√

It is also necessary to take account of the behavior of the system which depends on an application. Table 2 shows timing on which each churn tolerance technique works. This table indicates that replication is expensive with many put operations and multiple get is expensive with many get operations. Repeated implicit put is a process which keeps running and takes its cost continuously even without put or get requests. The cost in a time unit increases according to the number of key-value pairs in the DHT.

Suppose that the application is the domain name system (DNS), the frequency of get operations is much higher than other operations such as join and put. In this case, multiple get is expensive but replication and join-time transfer are relatively inexpensive. It may be possible to use an expensive parameter for such lower frequent operations. For example, an application like the DNS may allow a higher number of replicas. On the other hand, a different type of application shows different properties. A sensor network on a DHT will receive more put requests than the DNS and we cannot neglect the communication and storage cost of replication.

As shown here, we cannot calculate the cost of churn tolerance techniques without premising a concrete application. By making such a premise, we can estimate the frequency of each operations on a DHT, join, put, get and remove. If we have real-world traces of an application or a scenario which reflects application behavior, we can calculate the determinate cost based on them.

The emulation scenario for experiments shown in Section 4 is artificial. It does not make significant sense to inspect the cost in the experiments.

6 Related Work

A layered model of structured overlay proposed by Dabek et al. (Figure 2) [3] includes an API to implement replication (Section 3.1). The authors proposed the model and APIs between the layers. The paper includes no empirical proof of those proposals.

In contrast to it, I demonstrated that it is possible to implement a number of churn tolerance techniques such as join-time transfer, multiple get and repeated implicit put in addition to replication. Those techniques were implemented based on a single mechanism similar to Dabek's `replicaSet` function. The mechanism provides the DHT layer with the list of responsible node candidates. Section 4 showed the effects of those techniques combined with various routing algorithms.

Rhea et al. evaluated churn tolerance of Bamboo [8], that is their DHT implementation [9]. The authors used a network emulator ModelNet and ran 1000 nodes on 40 PCs. On the emulated environment, they measured the effects of methods to recover node failures, reactive recovery and periodic recovery. They also measured and evaluated techniques to reduce the required time to perform routing. Those techniques include TCP-style timeout calculation and proximity neighbor selection (PNS) [5].

All techniques in Rhea et al. are implemented in the routing layer of the Dabek's model [3], while the techniques described in this paper target the DHT layer. Rhea et al. premises their Bamboo implementation and its particular algorithm derived from Pastry. In contrast to it, the techniques in this paper naturally work in combination with various routing algorithms because their target is the DHT layer, which is independent of routing algorithms. The most significant contribution of this paper is empirical demonstrations of the combinations of the techniques and the routing algorithms shown in Section 4.

The techniques in this paper work together with all techniques shown in Rhea et al. because their target layers are different and they do not conflict. The communication layer of Overlay Weaver actually implements TCP-style timeout calculation proposed in Rhea et al.

Kato et al. evaluated 4 DHT algorithms, Bamboo, Chord, Accordion and FreePastry on their network emulator peeremu [10]. They conducted experiments in which up to 1000 nodes ran on 10 or 20 PCs. Their metrics were success rate of get requests and time to complete them.

There is a different type of approach to build a churn tolerant system, supernodes. The distributed system itself elects supernodes suitable for composing an overlay based on their attributes such as computation performance, bandwidth and running time so far. Only the elected supernodes maintain a DHT (an overlay) and they serve other ordinary nodes.

This supernodes architecture relaxes churn tolerance required for a DHT. It introduces another merit, by which the system qualifies convenient nodes capable of composing a DHT, for example, capable of bi-directional communication.

Conventional routing algorithms of structured overlay require all nodes on an overlay to be able to communicate each other. It makes DHT construction easier to choose supernodes as they can communicate in bi-direction, for example, not behind a NAT. With supernodes, churn tolerance is still an important property because even supernodes suffer churn.

7 Conclusion

I presented a number of churn tolerance techniques for DHT and demonstrated their effects. They work with various routing algorithms of structured overlay because they all target the DHT layer of the Dabek's model. I implemented those techniques in Overlay Weaver, measured their effects with all the algorithms Overlay Weaver provides, and showed the effects with Chord, Pastry and Kademlia in Section 4. The most significant contribution of this paper is the empirical demonstration of those routing-algorithm-neutral churn tolerance techniques.

In Section 5, I discussed how we can determine which technique and what parameter we choose. The decision should be based on cost performance of the techniques and I concluded that we need an emulation scenario which reflects a concrete application because the cost is heavily dependent on the frequency of join, get, put and remove operations on a DHT.

A promising next step is measurement and calculation of cost performance based on concrete applications such as DNS and sensor networks. It leads to the establishment of methodology by which we construct a churn-tolerant system with appropriate techniques and their parameters.

Acknowledgments

I would like to thank Daishi Kato, Youki Kadobayashi, Yusuke Doi, Akito Fujii, Mikio Yoshida and members of IDEON working group in WIDE project for insightful discussions.

References

1. Shudo, K., Tanaka, Y., Sekiguchi, S.: Overlay Weaver: An overlay construction toolkit. Computer Communications (Special Issue on Foundations of Peer-to-Peer Computing) 31(2), 402–412 (2008)
2. Shudo, K.: Overlay Weaver: An overlay construction toolkit (2006), `http://overlayweaver.sf.net/`
3. Dabek, F., Zhao, B., Druschel, P., Kubiatowicz, J., Stoica, I.: Towards a common API for structured peer-to-peer overlays. In: Kaashoek, M.F., Stoica, I. (eds.) IPTPS 2003. LNCS, vol. 2735. Springer, Heidelberg (2003)
4. Rhea, S., Chun, B.G., Kubiatowicz, J., Shenker, S.: Fixing the embarrassing slowness of OpenDHT on PlanetLab. In: Proc. WORLDS 2005 (2005)

5. Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of DHT routing geometry on resilience and proximity. In: Proc. SIGCOMM 2003 (2003)
6. Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., Yu, H.: OpenDHT: A public DHT service and its uses. In: Proc. ACM SIGCOMM 2005 (2005)
7. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: Amazon's highly available key-value store. In: Proc. SOSP 2007 (2007)
8. Rhea, S.C.: The Bamboo distributed hash table (2003), http://www.bamboo-dht.org/
9. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling churn in a DHT. In: Proc. USENIX 2004 (2004)
10. Kato, D., Kamiya, T.: Evaluating DHT implementations in complex environments by network emulator. In: Proc. IPTPS 2007 (2007)

PeerVote: A Decentralized Voting Mechanism for P2P Collaboration Systems

Thomas Bocek, Dalibor Peric, Fabio Hecht, David Hausheer, and Burkhard Stiller

Department of Informatics IFI, University of Zurich, Switzerland
{bocek,peric,hecht,hausheer,stiller}@ifi.uzh.ch

Abstract. Peer-to-peer (P2P) systems achieve scalability, fault tolerance, and load balancing with a low-cost infrastructure, characteristics from which collaboration systems, such as Wikipedia, can benefit. A major challenge in P2P collaboration systems is to maintain article quality after each modification in the presence of malicious peers. A way of achieving this goal is to allow modifications to take effect only if a majority of previous editors approve the changes through voting. The absence of a central authority makes voting a challenge in P2P systems.

This paper proposes the fully decentralized voting mechanism PeerVote, which enables users to vote on modifications in articles in a P2P collaboration system. Simulations and experiments show the scalability and robustness of PeerVote, even in the presence of malicious peers.

1 Introduction

Peer-to-peer (P2P) systems inherently support redundancy, scalability, fault tolerance, and load balancing [20,16] at a lower cost than client/server systems, since every participating user contributes with resources. These advantages incited the emergence of different P2P-based applications, including audio and video streaming [28], file-sharing [7], and storage [24,27]. A large-scale collaboration system, such as Wikipedia [26], could also benefit from the aforementioned characteristics of P2P systems [3]. In a P2P collaboration system, users share the resources necessary to host and distribute articles that can be modified by any other user.

A major challenge in P2P collaboration systems is to assure that user-generated content quality is being maintained or improved after each modification, despite the lack of a central authority. Since article quality is a highly subjective measurement, user-based voting is needed to allow users to express their opinion on whether or not a proposed modification shall be accepted.

The contribution of this paper is PeerVote, a decentralized voting mechanism that provides a method to maintain quality of content and its modifications. The proposed voting mechanism ensures that every modification to a document has the approval from the majority of previous editors. This helps to prevent vandalism, editing wars, and deliberate censorship. To the best of our knowledge, this is the first user-based voting scheme for P2P collaboration systems. PeerVote has been implemented in a P2P collaboration application and evaluated on top of a structured P2P network using a simulator and EMANICSLab [10]. PeerVote can be used for any kind of document writing with a primary focus on distributed collaboration, such as scientific reports, project deliverables, or online articles.

The remainder of this paper is structured as follows. While Section 2 discusses related work, Section 3 shows the design of PeerVote. In Section 4 implementation details and evaluation results are presented. Finally, Section 5 concludes this paper and suggests future work.

2 Related Work

One way of achieving the goal of improving quality of content in P2P collaboration systems is by using recommendation or reputation systems. [1] proposes a content-driven reputation system for Wikipedia, which is based on automatic analysis of edit-changes of Wikipedia articles and side-steps any user-based rating. The deriving author reputation can be used to predict the quality of future articles of such authors. Korsgaard and Jensen [14] outline the integration of a recommendation system in Wikipedia, which allows users to express their preference about the article and consult its general score, whereas the resulting system would still remain centralized. Although such systems can be used in conjunction with PeerVote, their focus is on the document itself, while PeerVote's main concern is on document's update compliance. While several P2P Wiki approaches exist [18,25,17,23], none of these approaches support user-based voting.

Different voting and consensus reaching algorithms have been proposed, with distributed systems and electronic voting as their main applications. In distributed systems, the main purpose of voting mechanisms is to ensure consistency among replicated data, usually by achieving an agreement between replica holder entities. Examples of such consensus protocols are the two-phase commit protocol [12], the weighted voting [11], and the decentralized weighted voting [19]. With the exception of the latter, such voting schemes have the disadvantage of being centralized. Another disadvantage of those schemes is that they are synchronous, thus, their application for human-based voting would require all voters to express their preference simultaneously, which is unfeasible for large numbers of participants.

Fully decentralized voting protocols, such as the inexact voting over wide area networks [13] and the Deno voting protocol [5] have been proposed. The first approach shows a message complexity quadratic with respect to the number of voters, which is not scalable. The Deno voting protocol, while being scalable, does not guarantee that updates commit in a bounded time.

Secret ballot protocols, or electronic voting protocols, implement a democratic voting system on electronic equipment. Some existing implementations [6,21] address specific security concerns, like eligibility, privacy, individual and universal verifiability, fairness, robustness, and receipt-freeness, but they work in a centralized fashion.

Table 1 shows a comparison among these voting mechanisms. None of these mechanisms reviewed support a decentralized user-based voting.

Table 1. Comparison of distributed voting mechanisms

	Decen-tralized	**Scalable**	**Time-bounded**	**Asyn-chronous**	**User-based**
2-Phase Commit	No	Yes	Yes	No	No
Weighted Voting	No	Yes	Yes	No	No
Decentralized Weighted Voting	Yes	Yes	Yes	No	No
Secure Ballot	No	Yes	Yes	Yes	Yes
Inexact Voting	Yes	No	Yes	Yes	No
Deno Voting	Yes	Yes	Yes	No	No
PeerVote	Yes	Yes	Yes	Yes	Yes

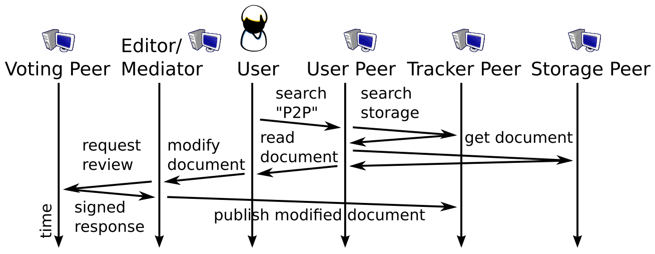


Fig. 1. PeerVote message sequence chart

3 PeerVote Design

The design of PeerVote defines roles and their interactions. Each role defines a set of actions and interacts with one or more other roles. The following use case introduces these roles.

3.1 Use Case

A typical use case for PeerVote is shown in Figure 1, where a user searches for a document including the keyword "P2P". The peer finds a tracker with addresses of peers holding documents with the keyword "P2P". After reading, the user finds a wrong statement and corrects it. Following the majority of previous authors agreeing on this correction, the modified document is published.

3.2 Roles

PeerVote defines six roles: tracker, storage, user, editor, mediator, and voter roles (cf. Figure 1). One peer can have one or more roles, depending on the action of a user.

Tracker Role: A tracker peer stores references to storage peers that store certain documents. Each tracker peer is responsible for storing references to documents with document-ids nearest to its peer-id. A tracker peer also evaluates voting metadata. A tracker can reject references with wrong or inconsistent voting metadata. References have a time-to-live value, so they are removed in case peers stop (re)publishing them.

Storage Role: A storage peer stores and periodically (re)publishes documents. Publishing is carried out by searching for trackers responsible for a particular document-id and storing addresses of storage peers and metadata. These documents are stored by peers that either created or viewed them.

User Role: A user interacts via a user interface with its user peer. The role of the user peer is to search for, download, and display documents. A search is carried out by searching trackers responsible for a particular document-id. These trackers reply with references to storage peers from which a user peer can download this document. If user peers publish downloaded documents, they also become storage peers.

Editor Role: An editor peer proposes modifications to a document. The editor hands over the change proposal to the mediator peer, which initiates the voting session.

Mediator Role: A mediator peer is responsible for a voting session. After a modification to a document is proposed, the mediator contacts its voting peers and requests them to review and sign the result. If a majority of those voters approve the change, the modified document is published. In the publishing process, references and the voting metadata are stored on tracker peers.

Voter Role: A voter peer can vote exactly once on a modification proposal. Allowed voters are editors of previous modifications of this document or editors with many approved votes.

3.3 Voting Algorithm and Data Structure

A voting session is initiated when an editor submits a proposal and becomes a mediator. As a mediator, it starts the voting phase by sending the proposal to all previous editors or to editors with many edits, if no previous editors exist in case of a new document. These editors can vote, if the change shall be accepted. The voting session is open for a specific amount of time. During this time, voters can review and vote for or against the change. If a voter does not vote, it is considered as a negative vote. Thus, there are no benefits of avoiding to contact peers for a review. A vote is accepted, if signed voting results reach a threshold. Figure 2 shows the pseudo code for the voting scheme, with the methods `voting()` and `incoming()`. The voting method is used by a mediator to start a voting session, while the incoming method is called on voting peers (previous editors).

In a decentralized system, it is important to take measures against dropping votes, *i.e.*, when a mediator ignores negative votes. For example, if a mediator

```
//start voting session
voting(Document d, time t) {
  //previous editor are stored in metadata
  MetaData m=d.getMetaData();
  List resultVotes;
  //ask previous editors to review document
  for editor in m.previousEditors() {
    resultVotes.add(requestReview(editor,d));
  }
  //this is blocking, use a thread and a
  //future object to make it non-blocking
  wait(t);
  return resultVotes;
}
//handle incoming voting requests
incoming(Voterequest r, Document d) {
  //display a popup for the user
  notifyUser();
  //ask the user to vote for or against
  boolean vote=displayAndReview(d);
  if(vote) {
    result=sign(d.getMetaData())
    //reply sends the result to the requester
    r.reply(result)
  }
  else {
    //reply sends null to the requester,
    //which means that the modification
    //was not approved
    r.reply(null)
  }
}
```

Fig. 2. Voting scheme pseudo code

```
//verify the metadata before
//adding to storage
addAndVerify(MetaData m1, PeerAddress n) {
  MetaData copyM1 = m1;
  MetaData m2=latestLocalMetaData()
  while (true) {
    //checks if version matches and previous
    //editors are in place
    if (isNextVersion(copyM1, m2)) {
      setLatestVersion(copyM1)
      //reached latest
      if (copyM1.equals(m1)) {
        //verification successful
        return true;
      }
      //start from the beginning
      m2 = copyM1;
      copyM1 = m1;
    }
    else  {
      copyM1 = copyM1.getPreviousMetaData();
      if (copy == null) {
        //reached first versios
        break;
      }
    }
  }
  //verification failed
  return false;
}
```

Fig. 3. Voting metadata verification

receives 98 negative votes and 2 positive votes, dropping 97 negative votes gets the modification accepted. One way of dealing with this is to introduce a threshold for a minimum number of votes for a valid outcome. Another way is to count no votes as negative votes and use a threshold for the positive votes to determine the voting outcome. PeerVote uses the second approach, thus dropping negative votes does not change the voting outcome.

Storage and user peers need to evaluate the voting metadata to detect unreviewed changes. The information required for this evaluation is stored in the voting metadata. The voting metadata in PeerVote consists of versioning information, hash code of the content of the document, document-id, and signed voting results. The document-id is constant for all changes on the same base document. The voting metadata has the structure as shown in Table 2.

It is important to verify the consistency of previous voting sessions. Otherwise, a malicious peer could replace, add, or drop signatures to pretend to have less editors or to be a previous editor. Thus, voting metadata is verified with the

Table 2. Voting metadata structure and example values

Nr	*Version*	*Document-id*	*Hash*	*Signatures (Signature, Peer-id)*	*Previous*
#1	1	0x123	0x234	(0x134,0x456)	
#2	2	0x123	0x235	(0x135,0x456) (0x136,0x567)	#1

following rules. Each version can have at most one new editor, a new version contains all peer-ids of previous editors, and all versions have the signed results of the voting sessions. These rules are verified as shown in Figure 3, which shows the pseudo code for verification of voting metadata. The voting scheme uses public-key cryptography for peer identification and signatures to prevent forgery. A public key is exchanged on first contact.

In such a decentralized system, peers can modify documents concurrently, because they operate independently. Concurrency in this voting scheme for change proposals uses a first come, first served scheme. This means that for conflicting changes, the peer that collected the votes first, publishes the change proposal and all other conflicting change proposals with the same version number are discarded. The user which submitted the failed change proposal is notified and can update to the latest version of the document and propose the change again.

3.4 Voting Scheme Example

The following example explains the design and the pseudocode, presented in Figure 3 with example values. While the new document (#1) in Table 2 with document-id 0x123 and hash value 0x234 has no reference to previous metadata, the modified document (#2) has a reference to (#1), which means that (#2) is based on (#1). For the modified document (#2), the hash value 0x235 changed, because of the modified content. The signature in the new document (#1) is based on the hash value 0x234 to confirm that its content was approved by peer with peer-id 0x456. This signature belongs to peer 0x456 in this example. All signatures of the modified document (#2) show that two voters have approved this modification, the original editor, and a second editor with peer-id 0x567. The signature for document #2 is based on the hash 0x235.

4 Implementation and Evaluation

PeerVote has been implemented in a P2P collaboration application, which has the primary focus on evaluating generic decentralized collaboration. Thus, the simulation does not focus on parameters for specific applications such as Wikipedia. This P2P collaboration application is based on TomP2P [2], a Distributed Hash Table (DHT) and tracker implementation.

4.1 P2P Collaboration Application Implementation

The P2P collaboration application supports document publishing, document downloading, and document modifying using a tracker-based approach. Publishing a new document requires first to search for a tracker with a peer-id closest to the document-id. A tracker is responsible for storing voting metadata and document references to storage peers. For new documents, the voting metadata is filled and stored with the reference to the storage peer on the tracker. A new document is accepted without any further validation because it is assumed that a new document is never published by a malicious peer.

Downloading a document starts with searching for trackers with a peer-id close to the document-id. These trackers contain addresses of peers that store the document. After those trackers are found, addresses and voting metadata are downloaded. The addresses from the trackers are used to download the document.

After a document is modified, all previous editors are requested to review the change. All voting peers that are online answer this request as described in the incoming method in the pseudo code of Figure 2. If a sufficient number of voters have signed the modified document, the document is published. For publishing a modified document, the tracker verifies voting metadata as described in the `addAndVerify` method (cf. Figure 3). If the verification is successful, the reference to the modified document and voting metadata is stored. Further search requests to this tracker return the modified metadata and its reference to the storage peer.

4.2 Simulation and Experimental Settings

Simulations and experiments have been run to investigate scalability, fault-tolerance, and robustness. This has been performed by using various combinations of the following parameters: number of peers, number of malicious peers, number of voting sessions, and number of change proposals. Since user-based voting in P2P collaboration applications have not been studied before, parameters, such as churn (10%), concurrency (50%), and number of publishing peers (10%, 20%, 30%), have been selected. Malicious peers can propose incorrect modifications, vote against correct modifications, or vote randomly. Malicious peers that vote randomly simulate users that vote without reviewing, because users may pretend to be active without investing resources for reviewing. A random voting peer votes with a probability of 50% for and with a probability of 50% against it. Malicious peers that propose incorrect modifications simulate users that intend to publish unsuitable information, *e.g.,* spam or contradicting information. Malicious peers that vote against correct proposals simulate colluding peers. A malicious vote is always detected because malicious modifications are marked as such and considering different opinions on modifications is not supported. In the current setting, the voting metadata is not verified by user peers, because malicious storage peers are not implemented. New documents do not have wrong information regardless of the publishing peer being malicious. A voting session lasts at most 45 seconds and a peer replies, if online, within 10 seconds to finish the simulation and experiments in a reasonable time. The voting meta data is accumulated during these changes. There is a 50% chance for peers to propose changes concurrently in a voting session. Currently, in the P2P collaboration application changes based on the same version are always conflicting and never merged.

Signatures are required to sign the result of a voting session. The current implementation generates a hash of the modified data as a replacement for the signature, because signing and verifying is CPU intensive and many peers are simulated on one machine, which makes the CPU a bottleneck. Since malicious peers do not exploit this, a signature can be verified by hashing the modified data. Thus, a distribution of public keys is not required.

PeerVote's evaluations are based on more than 1,000 peers. Churn has been set to 10%, which means that peers fail to reply in 10% of the time. Experiments and simulations have been run 10 times each and the averages have been calculated including the standard deviation.

All simulations have been performed on 2 Intel(R) Xeon(R) quad-core CPUs, 2.83GHz, with a Java HotSpot(TM) 64-Bit Server VM (build 11.2-b01, mixed mode). The parameters set for the simulation with Java were −d64 -Xmx10G, which allows to run in 64bit mode and use 10 GB RAM. The simulation required 2 days to complete. All experiments were run on 14 EMANICSLab [10] machines. EMANICSLab is a research network established among European research partners and consists of 7 different partner sites. The hardware used in EMANICSLab is heterogeneous with different CPU models and RAM sizes. The experiment required 2.5 days to complete.

4.3 Results and Discussion

Figure 4 shows the accumulated number of DHT and voting messages per peer with an increasing number of peers, from 100 to 1000. The number of published documents is set to 10%, 20%, and 30% respectively, proportional to the number of peers, which means that for 20% and 100 peers, 20 documents are published, for 1000 peers, 200 documents are published. A document is changed 2 times. All change proposals are carried out by 30% of the peers. This means that for 100 peers, 30 change proposals are submitted for 20 documents. Thus, some proposals are submitted concurrently. Malicious peers are not present in this simulation. Figure 5 shows the accumulated number of voting messages per peer with increasing number of peers using the aforementioned parameters.

Figure 4 indicates a logarithmic behavior for all three curves. Furthermore, Figure 5 shows a constant number of messages from 100 to 1,000 peers, and that the number of voting messages doubles if the publishing peers double. This indicates that the voting algorithm is scalable with an increasing number of peers. While the voting messages have a small number of messages (maximum of 4.75), the overall number of messages, including DHT overhead have a maximum of 575. Thus, this indicates that the DHT overhead contributes most to the number of messages.

Figure 6 shows the graph for voting traffic with an increasing number of documents, from 10% to 100% proportional to the number of peers. The number of peers shown are 400, 500, and 600. Voting parameters are set as described in the previous simulation. Malicious peers are not present. Figure 7 shows a graph for voting traffic with an increasing number of change proposals with the same settings as for Figure 6.

Both figures show for 400, 500, and 600 peers, that the 3 traffic graphs overlap. This indicates that the voting traffic is independent of the number of peers. The voting traffic is dependent on the number of documents (Figure 6) and the number of change proposals (Figure 7).

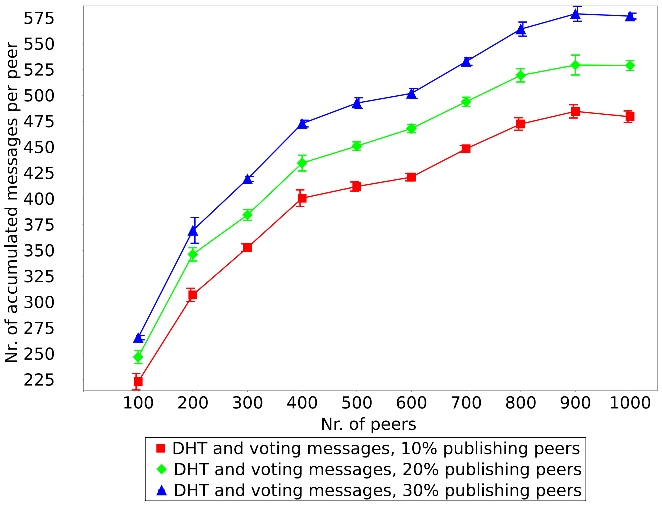


Fig. 4. Number of voting and DHT messages per peer

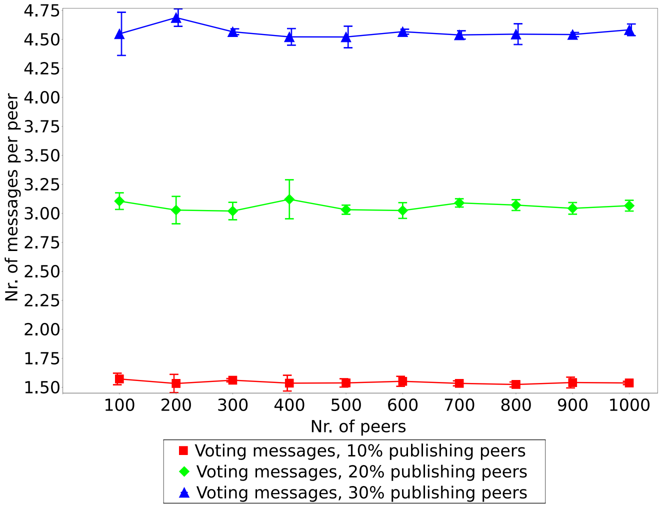


Fig. 5. Number of voting messages per peer

Figure 6 shows that traffic increases with linear complexity for an increasing number of peers proposing a change, while Figure 7 shows that traffic increases with polynomial complexity for an increasing number of change proposals per peer. The graph in Figure 7 is explained due to the increasing number of previous authors, which increase the message size and also the number of messages, because all previous peers need to be contacted. Thus, the list of previous authors for a document needs to have a fixed capacity.

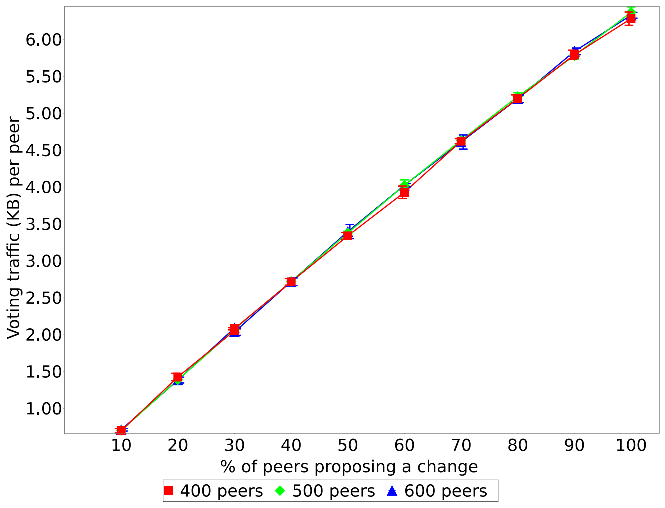


Fig. 6. Voting traffic for increasing number of proposing peers

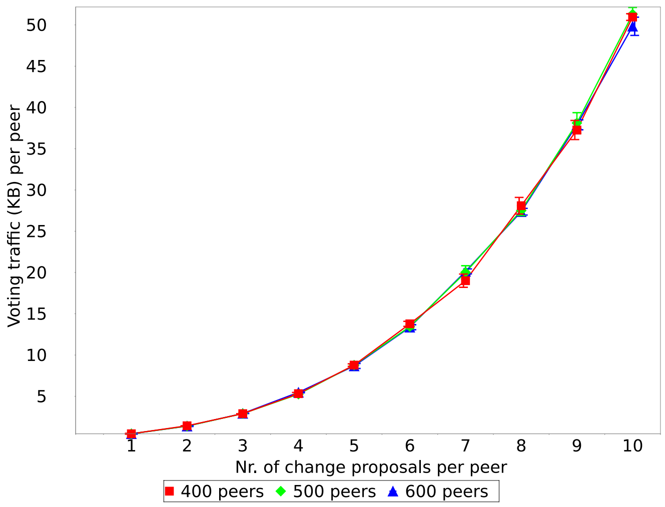


Fig. 7. Voting traffic for increasing change proposals per peer

To evaluate malicious behavior, up to 50% malicious peers have been simulated. Figure 8 shows an increasing number of malicious peers. The number of peers are set to 500. The number of published documents is set to 100. Both graphs are decreasing and start at 100% correct documents. The more malicious peers join, the more incorrect changes are present. The graph with random votes (Figure 8) has less incorrect changes compared to the graph with malicious votes. For 50% of malicious peers (250 peers), 65% correct documents are stored, while for 50% of random voting peers, 82% correct documents are stored. More than

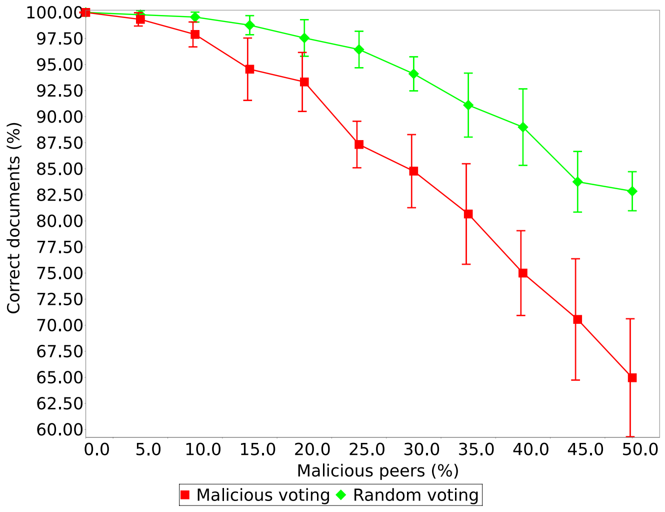


Fig. 8. Effect on documents with malicious and randomly voting peers

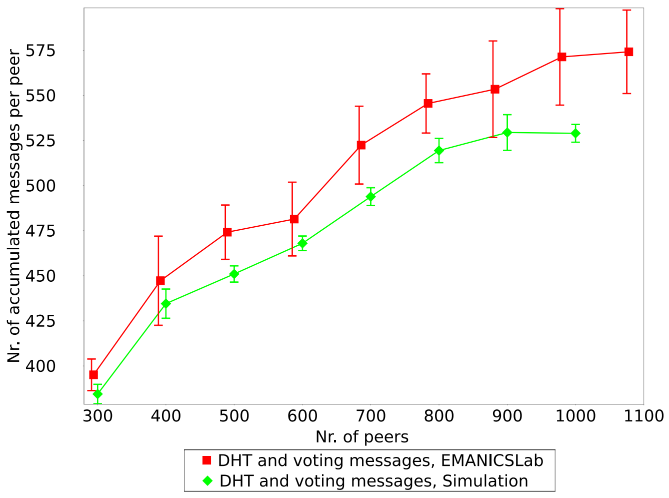


Fig. 9. Experiment on EMANICSLab and simulation comparison

half of the documents are correct, because the initial document is always correct, regardless if a peer is malicious or not.

Figure 9 shows the comparison of those simulation results above with the experimental results from the EMANICSLab implementation. Both settings have from 300 to less than 1,100 peers and 20% published documents. Two changes are proposed and the graph shows the number of DHT and voting messages.

The number of messages for the DHT and voting messages is higher on EMANICSLab, which is also observed for other comparisons. The higher message number is due to peers on EMANICSLab, which may fail to reply or which send a reply too late, because of other CPU or network intensive experiments running on EMANICSLab nodes. The larger error bars for EMANICSLab experiments also reflect this.

5 Summary, Conclusions, and Future Work

This paper presented PeerVote, a decentralized voting mechanism in a P2P collaboration application. Experiments and simulations with a prototypical implementation showed that PeerVote is scalable and robust, even in the presence of random and malicious voting peers. Evaluations showed that overall traffic increases logarithmically and that voting messages are independent on the number of peers. Furthermore, experiments on EMANICSLab showed that the algorithm deployed in a real network has similar traffic characteristics as in the simulation.

PeerVote considers no votes as negative votes, using a threshold to determine the voting outcome. The threshold has to be set on a per-application basis because an optimal value for this threshold depends on the application. Each application and its user may have a different social contract, which defines the ways how users act. If a social contract defines a loosely-coupled collaboration with voluntary contribution, it may see fewer responses to voting requests than in a tightly-coupled collaboration, where contributions are compulsory. Thus, *e.g.*, in a loosely-coupled online collaboration application the threshold has to be set to a lower value than in a tightly-coupled scientific collaboration application.

In contrast to P2P recommendation systems, which are typically used for recommending complete documents, PeerVote is used for managing changes of a document in a decentralized collaboration application. PeerVote allows previous authors to review changes. However, new documents and early proposals are prone to malicious behavior because only few or none previous editors exist. A possible solution is to combine a recommendation system with PeerVote. A second issue is that authors can block changes forever, if they do not respond to a review request. If for a first change proposal the previous author does not respond, then a majority is never reached. Furthermore, for old documents, previous authors may not be as responsive as for newer documents. Thus, such a threshold needs to be time-based, where old documents need less votes for a change. A third issue is that the list of previous authors grows, resulting in a polynomial traffic. This could be solved by limiting the capacity of this list.

The voting mechanism is not Sybil attack [9] proof, since it does not prevent a user from acquiring multiple peer identities. Mechanisms to prevent or detect acquisition of multiple identities have to be implemented. Such mechanisms may be implemented with certificates binding peer identifiers to real-world identities, either with a trusted third part or with a web of trust. Another mechanism to limit multiple identities is by exchanging or paying with resources to participate or not be excluded [4].

Future work will investigate how the PeerVote mechanism can be combined with an incentive scheme. With such a combination, each vote can be traded to encourage peers to review changes or to store and provide documents. Further work will also investigate PeerVote in other application domains. In other application domains more voting choices could be offered to voters.

Acknowledgements. This work has been performed partially in the framework of the EU IST Project EC-GIN (FP6-2006-IST-045256) as well as the EU IST NoE EMANICS (FP6-2004-IST-026854). Many thanks go to Lennart Svensson, who did initial work on this topic in his Master's thesis [22].

References

1. Adler, T.B., de Alfaro, L.: A content-driven reputation system for the wikipedia. In: Proceedings of the 16th international conference on World Wide Web (WWW 2007), New York, NY, USA, pp. 261–270 (2007)
2. Bocek, T.: TomP2P - A Distributed Multi Map (2009), `http://www.csg.uzh.ch/publica-tions/software/TomP2P`
3. Bocek, T., Stiller, B.: Peer-to-Peer Large-scale Collaborative Storage Networks. In: Bandara, A.K., Burgess, M. (eds.) AIMS 2007. LNCS, vol. 4543, pp. 225–228. Springer, Heidelberg (2007)
4. Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.: Secure routing for structured peer-to-peer overlay networks. SIGOPS Operating Systems Review 36(SI), 299–314 (2002)
5. Cetintemel, U., Keleher, P.J.: Light-Weight Currency Management Mechanisms in Deno. In: Proceedings of the 10th International Workshop on Research Issues in Data Engineering (RIDE), San Diego, CA, USA, February 2000, pp. 17–24 (2000)
6. Chaum, D.: Blind signature system. In: CRYPTO 1983: Advances in Cryptology, New York, USA, p. 153 (1983)
7. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON), Berkeley, CA, USA (June 2003)
8. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), Chateau Lake Louise, Ban (October 2001)
9. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
10. EMANICSLab, `http://emanicslab.csg.uzh.ch/`
11. Gifford, D.K.: Weighted Voting for Replicated Data. In: Proceedings of the 7th ACM Symposium on Operating Systems Principles (SOSP), pp. 150–162 (1979)
12. Gray, J.: Notes on data base operating systems. In: Flynn, M.J., Jones, A.K., Opderbeck, H., Randell, B., Wiehle, H.R., Gray, J.N., Lagally, K., Popek, G.J., Saltzer, J.H. (eds.) Operating Systems. LNCS, vol. 60, pp. 393–481. Springer, Heidelberg (1978)
13. Hardekopf, B., Kwiat, K., Upadhyaya: Secure and Fault-Tolerant Voting in Distributed Systems. In: Proceedings of the 2001 IEEE Aerospace Conference, Big Sky, Montana, USA, March 2001, vol. 3 (2001)
14. Korsgaard, T., Jensen, C.: Reengineering the Wikipedia for Reputation. In: Proceedings of the 4th International Workshop on Security and Trust Management (STM 2008), Trondheim, Norway, June 2008, pp. 71–84 (2008)

15. Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: OceanStore: An Architecture for Global-scale Persistent Storage. In: Proceedings of the 9th international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Cambridge, MA, USA (November 2000)
16. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, p. 53. Springer, Heidelberg (2002)
17. Morris, J.C., Lüer, C.: DistriWiki: A Distributed Peer-to-Peer Wiki. In: Proceedings of the 2007 International Symposium on Wikis (WikiSym 2007), Montreal, Quebec, Canada, October 2007, pp. 69–74 (2007)
18. Mukherjee, P., Leng, C., Schürr, A.: Piki - A Peer-to-Peer based Wiki Engine. In: Proceedings of the 2008 8th International Conference on Peer-to-Peer Computing (P2P 2008), Washington, DC, USA, September 2008, pp. 185–186 (2008)
19. Rodrig, M., LaMarca, A.: Decentralized weighted voting for P2P data management. In: Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access (MobiDe 2003), San Diego, CA, USA, pp. 85–92 (2003)
20. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
21. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
22. Svensson, L.: Decentralized Secure and Incentive-compatible Votin in P2P Networks. Master's thesis, Communication Systems Group, IFI, University of Zurich, Switzerland (March 2007)
23. Urdaneta, G., Pierre, G., van Steen, M.: A Decentralized Wiki Engine for Collaborative Wikipedia Hosting. In: Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST), Barcelona, Spain (March 2007)
24. Warner, B., Wilcox-O'Hearn, Z., Kinninmont, R.: Tahoe: A Secure Distributed Filesystem (March 2008), `http://allmydata.org/~warner/pycon-tahoe.html`
25. Weiss, S., Urso, P., Molli, P.: Wooki: A P2P Wiki-Based Collaborative Writing Tool. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 503–512. Springer, Heidelberg (2007)
26. Wikipedia, `http://www.wikipedia.org`
27. Wuala, your files online, `http://wua.la`
28. Zattoo — TV meets PC, `http://zattoo.com/`

Evaluation of Sybil Attacks Protection Schemes in KAD

Thibault Cholez, Isabelle Chrisment, and Olivier Festor

MADYNES - INRIA Nancy-Grand Est, France
{thibault.cholez,isabelle.chrisment,olivier.festor}@loria.fr

Abstract. In this paper, we assess the protection mechanisms entered into recent clients to fight against the Sybil attack in KAD, a widely deployed Distributed Hash Table. We study three main mechanisms: a protection against flooding through packet tracking, an IP address limitation and a verification of identities. We evaluate their efficiency by designing and adapting an attack for several KAD clients with different levels of protection. Our results show that the new security rules mitigate the Sybil attacks previously launched. However, we prove that it is still possible to control a small part of the network despite the new inserted defenses with a distributed eclipse attack and limited resources.

Keywords: P2P networks, DHT, Sybil attack, security, defense, KAD.

1 Motivation and Scope of Work

Peer-to-Peer (P2P) networks have proven their ability to host and share a large amount of resources thanks to the collaboration of many individual peers. They are known to have many advantages compared to the client-server scheme: P2P networks scale better; the cost of the infrastructure is distributed and they are fault tolerant. Most currently deployed structured P2P networks are based on Distributed Hash Tables (DHT). Each peer is responsible for a subset of the network. This organization improves the efficiency of P2P networks ensuring routing in O(log n) but can also lead to security issues like the Sybil Attack.

The Sybil Attack, as described by Douceur [3], consists in creating a large number of fake peers called the "Sybils" and placing them in a strategic way in the DHT to take control over a part of it. Douceur proved that the Sybil Attack cannot be totally avoided as long as the malicious entity has enough resources to create the Sybils. This problem was not considered when designing most of the major structured P2P networks. In this context, the goal of the defense strategies described in the literature is to limit the Sybil Attack as completely stopping it is impossible. Proposed solutions are based on a strong identification of the peer [2] or on a trusted central authority [1]. In [8], the authors propose to bound the degree of overlay nodes by anonymous auditing to limit localised attacks but they also rely on a central certification to limit the number of Sybils. A strong identification being not adapted to many P2P applications, other strategies are possible to limit the Sybil attack considering that a distributed assignment with free identifiers is possible but, in compensation, must be verified by resource consuming proofs [7] or an underlying social network [10]. The most successfully completed protection against the Sybil attack [4] uses distributed certification coupled with a social network, but no studies consider practical protections set in a real network.

Even if defense against the Sybil attack has been largely investigated, most of the proposed solutions are difficult to set up and are not always well suited for the P2P paradigm. Things are even worse when considering all the constraints of a large public P2P network like KAD without a managed support infrastructure and with the need of backward compatibility. KAD provides a file sharing application based on the academic Kademlia [5] DHT and has been proven vulnerable and totally unprotected against Sybil attacks that can highly affect the network [9]. In this context, managing and protecting the KAD network seems very challenging.

The latest versions of the major KAD clients have introduced new protection mechanisms to limit the Sybil attack, making the previous experiments concerning the security issues of KAD like [9] inefficient. These newly implemented protection mechanisms have neither been described nor been evaluated and assessed. The purpose of this study is to evaluate the implemented security mechanisms against real attacks. We will then be able to have an updated view of KAD vulnerabilities: Is the network still vulnerable to the attack proposed in [9]? Is it now fully protected? Which vulnerabilties have been corrected and which ones keep being a threat? This work is a first and necessary step to design improved defense mechanisms in future works. As far as we know, this paper is also the first attempt to experiment and assess practical protections set by the P2P community to protect a real network.

This document is structured as follows. Section 2 describes the background of KAD, the Sybil attacks performed on it, and the new defense mechanisms. We then present in Section 3 our evaluation of the mechanisms including our methodology, the results obtained by the different versions of KAD clients and the impact of our distributed eclipse attack. Finally, Section 4 concludes the paper with a discussion on the validity of the built-in protections and outlines our future works.

2 KAD and the Sybil Attack

2.1 Overview of KAD

KAD is a structured P2P network based on the Kademlia distributed hash table routing protocol [5]. KAD is implemented by the popular eMule file sharing application and its multi-platform version aMule, both are open source. Historically, these clients were mainly designed to join the eDonkey network composed of autonomous distributed servers. However, many major eDonkey servers have been closed by lawsuits so that the eDonkey network is becoming less attractive and users are progressively migrating toward the fully decentralized P2P network, KAD. With an estimated number of concurrent online users around 4 millions, KAD is the widest deployed structured P2P network.

Each node of KAD has a 128bits KADID defining its position in the DHT. Two types of requests are used to discover the network: the *Hello_REQ* is used by a peer to announce itself and the *Kademlia_REQ* is used to discover new peers toward a specific address. The routing is based on the XOR metric: to fill the routing table, each node registers at level **i** K-contacts (called a K-bucket), each contact being at a distance between 2^{128-i} and 2^{127-i} from its KADID regarding the XOR metric. The deeper the contact is in the tree, the closer it is to the node and the better the peer knows this part of the DHT; this provides routing in O(log n). The routing is done in an iterative way with parallel lookups: asking at first the **n** closest contacts to the target ID found in the routing table for even closer nodes toward the target with *Kademlia_REQ*, waiting for their responses and then reiterating.

As a file sharing application, the purpose of the DHT is to retrieve information like keywords and files. When sharing a new file, the binary and all the keywords are hashed separately with a MD5 function generating KADIDs and then published. The peers in charge of a file or a keyword are those close enough to their hash. This distance is called the tolerance zone and is set to the first common 8bits (most significant bits). The double indexation allows retrieval of a particular file, given a set of keywords. To publish a file, two types of requests are sent:

- *KADEMLIA2_PUBLISH_KEY_REQ* requests are sent to the hash of the keyword and associate a keyword with a file
- *KADEMLIA2_PUBLISH_SOURCE_REQ* requests are sent to the hash of the file and associate a file to a source (a node able to upload the file)

2.2 Previous Attacks in KAD

Recent investigations showed that KAD can be cheated in several ways using few resources and resulting in important outages.

Steiner et al [9] were the first to successfully launch a real Sybil attack on KAD, resulting in the full control of a part of the network. The attack is divided in two steps. The first step consists in a crawler gathering information about what peer is in a zone of the network and what its routing table is. This is done by issuing route requests (*Kademlia_REQ*) toward some predefined node IDs in order to obtain new contacts close to those IDs. When almost all nodes of the zone are known by the crawler, the second step consists in individually contacting each node of the list to pollute its routing table with Sybils. Previously trivial, injection of Sybils is now protected by several mechanisms studied in the forthcoming sections.

In [6], the authors describe an attack that allows denial of service on a great part of the network with few resources. This attack does not rely on a classical injection of Sybils but hijacks the references of contacts in the routing table. Malicious nodes are able to overwrite a legitimate routing table entry (KADID/IP address) with their own reference by sending a simple *Hello_REQ* announcing this KADID. This weakness highlights a real lack of identity management in KAD. Thanks to the information acquired by a crawler, this vulnerability allows to partition the network or to do a massive denial of service. A search request starts from bad references and is then kept by them, until it terminates without any real result, or fails because of a timeout. The experiments on PlanetLab show that the attack is effective and does not use much bandwidth with some optimizations. The authors also noticed the improvements of the latest versions of the clients which could mitigate their attack, but they did not evaluate them.

2.3 Protections in KAD

The importance of these weaknesses forced the developers to react by setting up new protections in the latest versions of the clients. Even if older clients are still vulnerable, progressive updates should be able to mitigate the attacks and constitute a healthy basis for the network. KAD clients implement different levels of protection regarding their version. For similar versions, eMule and aMule are based on the same code and behave in the same way. Table 1 sums up the protections implemented in each of the clients.

Table 1. Protection enabled according to the client version

Clients	Flood protection	IP limitation	IP verification
eMule 0.48a / aMule 2.1.3	No	No	No
eMule 0.49a / aMule 2.2.1	Yes	Yes	No
eMule 0.49b / aMule 2.2.2	Yes	Yes	Yes

No protection: aMule 2.1.3 was chosen as a client without any protection against the Sybil attack. Basically, aMule accepts all incoming messages without any check: requests can be sent as fast as the client can handle them. There is no restriction regarding the KADID or the IP address of the sender. As long as the sender of a request matches a place left in a K-bucket, it can be added. With the KADID of the target and a good knowledge of the routing table mechanisms, it is possible to quickly take control of a legitimate peer.

Packet tracking and flood protection: The packet tracking function records the history of recently received and sent packets (12 minutes). This history is used for two purposes. The first goal is to be able to detect and drop unsolicited request answers. The second purpose is to set up a protection against flooding. With the history and time of previously received packets, it is possible to detect if an IP address is a source of flooding. For each type of KAD request, a threshold is defined per peer under which the number of received requests is considered legitimate. If the threshold is not highly exceeded, the packets over the limit are simply dropped. If the rate of received requests is more than five times above the limit, the sender is considered as an attacker and his IP address is banned from the client.

IP address limitation: The IP address limitation tries to mitigate the Sybil attack, considering that public IP addresses are more or less restricted per participant. Before adding a new peer in the routing table, its IP address is checked and dropped if it is already used. This verification limits to one the number of KADID per IP address in the routing table. This really helps protecting a client against a Sybil attack which needs a lot of public IP addresses to be significant. The IP address limitation also takes care of IP addresses from the same /24 subnetwork or smaller. In fact, at most ten IP addresses from the same /24 subnetwork can be added to the routing table to prevent a single entity with many resources to be able to launch an attack. These ten peers are also constrained to be in different K-buckets of the routing table, or in other words to have spaced KADIDs. This is done to prevent the entity owning a /24 subnet to launch a localized attack which needs to have Sybils very close to the target ID. This protection does not affect NATed users which can fully participate to the network with the very unlikely constraint that two NATed peers with the same public IP address can not be referenced in the same routing table of another contact.

Identities verification: The last KAD clients implement even more protections concerning identity management. They aim to limit the identity spoofing (both IP address and KADID) between peers. Before adding a contact, KAD now uses a three-way handshake for new contacts, making sure they do not use a spoofed IP. Older version are verified with a similar check as presented in table 2. Oldest clients are checked with a second Hello_REQ - Hello_RES exchange including a random KADID as a challenge for the responding peer. Newer clients use an exchange of PING and PONG messages which add cryptographic identification. Finally, the last versions do the three-way handshake in the most effective way with the specially added new message Hello_RES_ACK.

Table 2. Three-way handshake is achieved differently regarding the client version

Clients	Sequence	Nb of messages	Encryption
eMule 0.48a	Hello_REQ - Hello_RES - Hello_REQ(challenge) - Hello_RES(challenge)	4	No
eMule 0.49a	Hello_REQ - Hello_RES - PING - PONG	4	Yes
eMule 0.49b	Hello_REQ - Hello_RES - Hello_RES_ACK	3	Yes

Moreover, KAD contacts are only able to update themselves in other's routing tables if they provide the proper private key corresponding to the public key initially presented (supported by 0.49a+ nodes), in order to make it impossible to hijack them. Finally, contacts which fail the challenge are marked unverified and are not used for routing tasks. Theses protections avoid attacks using IP-spoofing and contact overwriting. It is a response to the routing attack issues described in [6] and one of the solutions proposed by the authors.

3 Evaluation of the Protection Mechanisms

In this section we evaluate different KAD clients and the mechanisms described above when facing a Sybil attack.

3.1 Methodology

As the injection of Sybils is the basis for attacking the network, we measure how permeable the routing table of a single peer is regarding the different levels of protection, and how restricted in their possibilities are the Sybils when they pass the protections. To evaluate the behavior of the defense mechanisms, we developed a basic attack software with the initial objective of polluting the routing table of a single peer running an unprotected version of KAD client. We then incrementally added the protection mechanisms by evaluating newer versions of the client and modified our attack accordingly.

For some of our experiments, our Sybil attack does not target the entire network as the one in [9] but only aims a single client. In fact, the protection mechanisms that we want to study are local to the client and do not require any interaction between peers. In consequence, the results obtained when evaluating a single client are sufficient and can be easily generalized as a good indicator of what the network will be in the future when most of them will be up to date.

Our attack announces Sybils by sending many *Hello_REQ* with forged KADIDs. The main difficulty of this attack is to announce the right identity for each Sybils to match the algorithm filling the routing table of the target and as consequence, maximize the pollution. Designing the generation of Sybil identities requires knowledge of the structure and algorithms of KAD routing table.

Looking at the code of KAD, the real structure of its routing table is slightly different from the proper binary tree defined in Kademlia [5]. First of all, all leafs can be split until level 4 without restriction, resulting in 16 K-Buckets covering

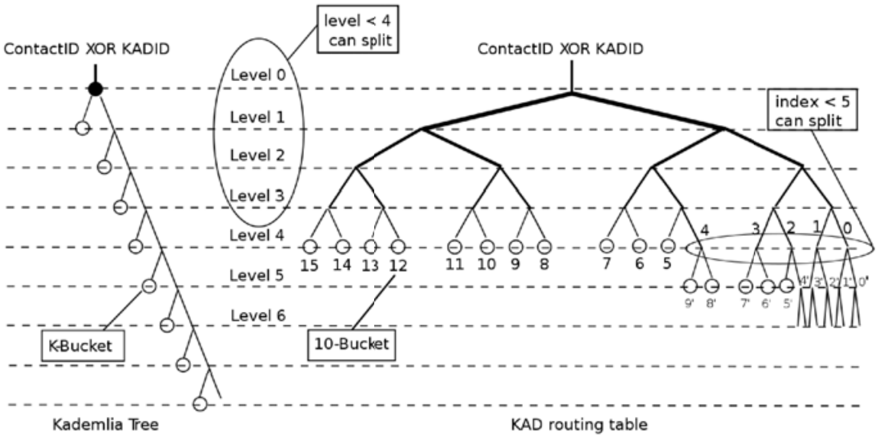


Fig. 1. KAD routing table scheme

the entire address space and used to route the farthest destinations. Then, at the level 4, the closest K-buckets with an index below 4 (which means that considering the 4 first bits: KADID XOR ContactID <= 4) can keep splitting until having an index of 8 or 9 by involving further bits as shown in figures 1. The depth of the routing table is not hardly limited but is limited de facto by the size of the network because fewer contacts are possible for deep buckets whose split becomes unlikely.

With this knowledge of the routing table and a targeted KADID, preparing the attack consists in generating a set of proper masks so that: targetID XOR mask = SybilID. The value of the masks are defined to generate SybilIDs progressively filling all the routing table of the targeted peer from the top to the bottom, by hitting the right K-bucket.

As we inject specifically forged contacts, they can hit deeper K-buckets in the routing table so that the size of the table under attack is higher than the size during a normal run. From the target point of view, the announcement of very close Sybils is considered as if the DHT space was more populated and consequently closer contacts easily findable.

3.2 Results

No protection: The first client attacked was aMule 2.1.3. This version is unprotected resulting to the realization of the simplest and most effective Sybil attack possible. Sybils KADIDs are forged as described previously and announced with a rate of 4 per second which is sufficient to populate the K-buckets faster that the normal algorithm without overloading the target client.

Actually, KAD contacts are periodically checked by sending an *Hello_REQ* and waiting for the *Hello_RES* to keep the routing table up to date. But this version also presents a kind of optimization that becomes a major design flaw when considering the Sybil attack. In fact, every *Hello_RES* from an IP address (whatever its KADID is) acknowledges every contact with this IP address. As all Sybils share the same IP address, a single *Hello_RES* aknowledges this IP address and is sufficient to keep hundreds of Sybils alive. Moreover, a peer can announce itself with a *Hello_RES* instead of a *Hello_REQ*, and still be added to the table. Following this discovery, our attack sends *Hello_RES* so that each new Sybil acknowledges and maintains the previous ones in the routing table without any message exchange.

The first results show that the routing table of the unprotected version is almost fully corrupted under attack (graph 2): at the end of an attack lasting 300s, the routing table of aMule counts 70% of Sybils (689 Sybils for 953 contacts). The major part of the routing table which is not polluted is made up of 200 good contacts saved from the previous execution and pre-loaded to bootstrap. Graph 3 shows at which speed the routing table of aMule is filled regarding if attacked or not. Even if the speed of *Hello_RES* sent by the Sybils was moderate, we can clearly see the Sybil attack as the routing table abnormally grows under the announcements of Sybils. Theses results confirm the need for the following protection mechanisms.

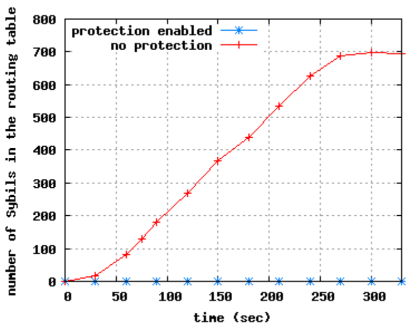


Fig. 2. Propagation of Sybils in the routing table

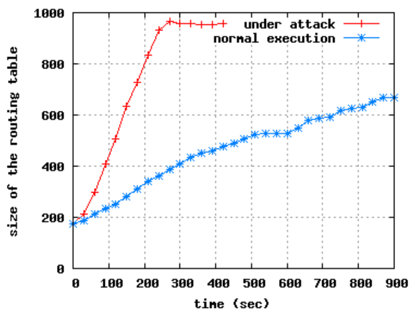


Fig. 3. Filling of the routing table under attack

Flooding protection: The second experiment aims to evaluate the flood protection. So, we launched an attack against the eMule 0.49a client with the IP limitation disabled. To be successful, the attack had to be improved in two ways. First, the packet tracking used to implement the flood protection drops *Hello_RES* without previous *Hello_REQ* so that the weakness described above to maintain Sybils is not possible anymore. Therefore, we modified our attack to keep each Sybil alive by responding to the *Hello_REQ* messages from the victim. The problem is that *Hello_REQ* messages do not include the targeted KADID whereas *Hello_RES* messages need it. A normal peer just has to manage its own KADID so that it clearly knows what to respond to a *Hello_REQ* message. On the contrary, our Sybil attack manages multiple identities and does not know which KADID is addressed by the *Hello_REQ* message. To bypass this limitation and find with which KADID to answer, each Sybil communicates on a different UDP port so that we can retrieve the wanted KADID. With this modified attack, we are able to maintain Sybils alive in the long run. Secondly, we sent *Hello_REQ* messages every 30 seconds to be under the threshold.

Our experiments show that the flooding protection works as expected. The limitation rate before dropping *Hello_REQ* is set to 3 packets per minute and above 15 packets per minute, our IP address is banned. The results are shown in graph 4. We can see that even if the attack is clearly slowed down, taking more than 8 hours to inject the same number of Sybils where previous attack only took few minutes, the routing table is still polluted with more than 60% of Sybils (540/873). The flooding limitation can protect the majority of short connections to the KAD network, but longer connections still suffer from a widely infected routing table.

Adding IP limitation: Our previous attacks were performed from a computer using a single IP address. With IP limitation enabled, we effectively measure that only the first Sybil enters the routing table, the other KADIDs presenting the same IP address can not pass. To bypass this limitation, we modified our attack using raw sockets so that each Sybil is announced with its own randomly spoofed

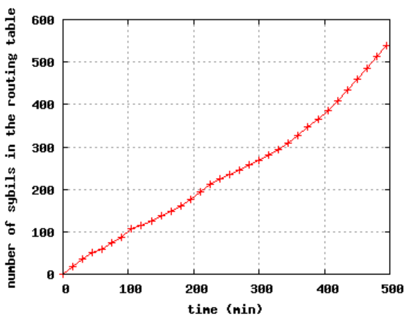


Fig. 4. Propagation of Sybils in the routing table with flood protection enabled

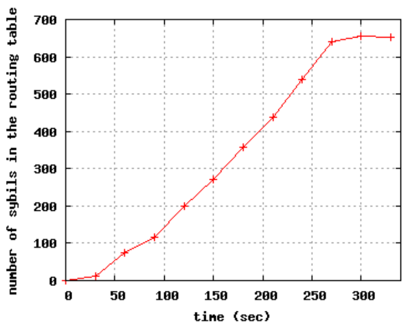


Fig. 5. Propagation of Sybils with spoofed IP addresses with flood & IP limitation protections enabled

IP address. The graph 5 shows that Sybils with spoofed IP addresses totally break this protection. The first time we launched the attack to evaluate the IP limitation, we disabled the flooding protection. When activating both of them in a second time, we noticed that the flooding protection is also completely bypassed by the IP spoofing. In fact, the packet tracking used to detect flooding measures the number of incoming requests from a given IP address. When spoofing IP addresses, each message appears to be from a different source so that the flooding is not detected. Even if IP spoofing can not be used to spy or eclipse content in the network, spoofed IP addresses were used in [6] with contact overwriting to partition the KAD network. This attack remains possible at this point, but the very last version of the clients implements a protection to mitigate both IP and KADID spoofing.

Adding identity verification: The attack launched to test the identity verification mechanisms is the same as the one described above, using Sybils with spoofed IP addresses. Part of the behavior of this last mechanism seems strange and disappointing because the Sybils that fail the three-way handshake are still added (at least temporary) to the routing table and marked as "unverified". The rate of unverified contacts is high even without attack during the first 15 minutes of connection (graph 6). Even if unverified contacts are not used for routing tasks, they can take the place of potentially good contacts in a K-bucket, resulting in a reduction of the routing efficiency for the peer. The positive aspects are that all Sybils are marked as unverified (graph 7) and that unverified contacts are dropped faster from the routing table than with the classical 1-hour timeout. Moreover, unverified contacts can not update themselves before being checked, so that Sybils with spoofed IP addresses really are active for only a limited time. As expected, we also experimented that KADID overwriting is from now on impossible without the proper private key and mitigates the attack described in [6].

The results obtained with the very last version show significant improvements of the Sybil attack defense. As expected, IP address limitation coupled with

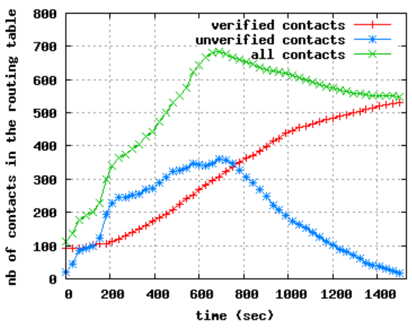


Fig. 6. Evolution of the contacts status in a normal run

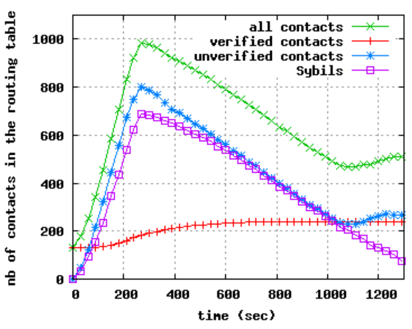


Fig. 7. Evolution of the contacts status under Sybil attack

identity verification avoids the Sybil attack from a single source when it previously would have been able to infect and spy or DoS the whole network.

3.3 Distributed Eclipse Attack

At this point, we have tested the resilience of the routing table to Sybils with the new protection mechanisms. But a massive Sybil attack is not the only and smartest way to damage the network. More localized Sybil attacks are possible and need fewer resources. The eclipse attack is another well-known issue of the KAD network as described in [9]. It consists in placing few malicious peers very close to the target ID in the DHT and making them known. Prior to the attack, the authors used a "crawler" in a tolerance zone to know many contacts and promote their Sybils. As all requests toward the target will pass through this zone where the peers are polluted, and considering the KAD search algorithm "return the X closest nodes...", the requests will arrive at malicious nodes with a high probability.

More than the massive injection of Sybils, we think that the heart of the eclipse attack is the possibility to freely chose its KADID very close to a target which keeps being possible. As this issue is not yet handled in the last versions of KAD, the eclipse attack remains a threat even if the IP limitation now requires its distribution (involving several public IP addresses). We did a distributed eclipse attack with 24 nodes from PlanetLab and EmanicsLab[1]. Our client is based on aMule 2.2.2, compiled to run as a daemon and modified in several points to make the attack. Our client uses a forged KADID to be very close (96 bits) to the hash of a given keyword. For this specific ID, our client spies the incoming requests and eclipses the keyword by answering to Publication requests and denying Search requests. To improve its efficiency, our client announces itself actively by periodically sending Kademlia routing and Hello requests toward

[1] EmanicsLab is an autonomous testbed network based on PlanetLab architecture and sponsored by the European Network of Excellence EMANICS.

pre-defined KADIDs in order to be known in the network and attract requests, as an alternative to a centralized crawler.

We first launched our attack on the keyword "the", known to be the most indexed keyword in the network. Despite a massive indexation of publish requests by our Sybils (16000 requests per minute), we cannot totally eclipse the keyword. So, we analysed the results obtained by a search request on "the" and we found out that all the answers came from the same IP address managing 256 KADIDs sharing 120bits with the keyword, but not denying any Search request. In other words, "the" was already the target of another classical Sybil attack, always possible as a large part of the network is not yet updated with the new clients. In a second time, we launched our attack on the keyword "document", receiving "only" 115 publish requests per minute but resulting in its total eclipse. Even if the needed bandwidth depends on the size of the attack and the popularity of the targeted ID, it keeps very small (few KB/s by node).

In conclusion, distributed and localized attacks do not require substantial resources and keep being a real threat despite the new mechanisms as long as the KADID can be freely chosen and the malicious nodes placed very close to a given target. Controlling a small botnet, or more generally some IP addresses distributed over multiple subnets, allows a malicious user to control several entries of KAD.

4 Conclusion

Considering the previous attacks and our evaluation of an old client, KAD was totally unprotected and very easily and badly hurt by a Sybil attack from a single computer. The local solution chosen by the authors of eMule fits the constraints of the network: no infrastructure cost and backward compatibility. The network is progressively being composed of new clients including the protections and becomming more robust. We have shown that the new defensive rules really make the Sybil attack harder to perform and successfuly mitigate the previously experimented attacks at a low cost. Where an attack could have been launched from a single computer few months ago [9], similar results can only be achieved now with a botnet or a more complex distributed architecture.

Even if these security mechanisms are a step forward, they are not yet sufficient. We have shown that a distributed eclipse attack focused on a particular ID still remains possible with a moderate cost. This result shows that the main weakness of KAD has shifted to the possibility to reference many KADIDs with the same IP address to the left possibility to freely choose its KADID. Moreover, if we consider an attacker with many resources, particularly considering the number of IP addresses, the overall protection can be threatened due to the specific design using local rules. As all protections are local to the client, a same pool of IP addresses can potentially infect all peers. Considering this, an attacker with several hundreds of spaced IP addresses can massively infect the network (for example, if controlling a medium-sized Botnet) and realize large attacks on the network scale. Thus, the IP limitation as currently defined, seems to be a very short-lived protection mechanism when considering IPv6. A single personal broadband connection could be assigned with a /64 or even more, which is sufficient to launch massive Sybil attacks with one different public IPv6 address per Sybil. The rules concerning the IP address limitation will have to be adapted carefully in order not to become inefficient.

However, even if not perfect, the defenses against the Sybil attack studied in this paper are absolutely generic and can be easily applied to every structured P2P network. We think that this kind of ”common-sense” protection based on self-management is the minimum that every implementation of a structured P2P network should have, unless being totally unaware of the Sybil attack issues. In our future work, we will design and evaluate new security mechanisms to mitigate the distributed eclipse attack. It is interesting to consider the impact of this attack on the routing table of KAD and on the indexing and search processes. Being given the existing security rules of the indexing scheme, evaluating the remaining weaknesses will lead us to design new constraints improving the local protection.

Acknowledgment. This paper was partially supported by the French Ministry of Research project, MAPE, under contract ANR-07-TLCOM-24 and by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.: Secure routing for structured peer-to-peer overlay networks. In: OSDI 2002: Proceedings of the 5th symposium on Operating systems design and implementation, pp. 299–314. ACM, New York (2002)
2. Dinger, J., Hartenstein, H.: Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration. In: ARES 2006: Proceedings of the First International Conference on Availability, Reliability and Security, Washington, DC, USA, pp. 756–763. IEEE Computer Society, Los Alamitos (2006)
3. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
4. Lesueur, F., Mé, L., Tong, V.V.T.: A sybil-resistant admission control coupling SybilGuard with distributed certification. In: Proceedings of the 4th International Workshop on Collaborative Peer-to-Peer Systems (COPS), Rome (June 2008)
5. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002)
6. Chan-Tin, E., Malchow, T., Kune, D.F., Hopper, N., Kim, Y., Wang, P., Tyra, J.: Attacking the kad network. In: SecureComm 2008: the 4th International Conference on Security and Privacy in Communication Networks, Istanbul, Turkey. ACM, New York (2008)
7. Rowaihy, H., Enck, W., McDaniel, P., La Porta, T.: Limiting sybil attacks in structured p2p networks. In: 26th IEEE International Conference on Computer Communications, INFOCOM 2007, pp. 2596–2600. IEEE, Los Alamitos (2007)

8. Singh, A., Castro, M., Druschel, P., Rowstron, A.: Defending against eclipse attacks on overlay networks. In: EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop, p. 21. ACM, New York (2004)
9. Steiner, M., Najjary, T.E., Biersack, E.W.: Exploiting KAD: possible uses and misuses. Computer communications review 37(5) (October 2007)
10. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.: Sybilguard: defending against sybil attacks via social networks. SIGCOMM Comput. Com. Rev. 36(4), 267–278 (2006)

NETCONF Interoperability Testing

Ha Manh Tran, Iyad Tumar, and Jürgen Schönwälder

Computer Science, Jacobs University Bremen, Germany
{h.tran,i.tumar,j.schoenwaelder}@jacobs-university.de

Abstract. The IETF has developed a network configuration management protocol called NETCONF which was published as proposed standard in 2006. The NETCONF protocol provides mechanisms to install, manipulate, and delete the configuration of network devices by using an Extensible Markup Language based data encoding on top of a simple Remote Procedure Call layer. This paper describes a NETCONF interoperability testing plan that is used to test whether NETCONF protocol implementations meet the NETCONF protocol specification. The test of four independent NETCONF implementations reveals bugs in several NETCONF implementations. While constructing test cases, a few shortcomings of the specifications were identified as well.

Keywords: Network Management, NETCONF, Interoperability Testing.

1 Introduction

The NETCONF protocol specified in RFC 4741 [1] defines a mechanism to configure and manage network devices. It allows clients to retrieve configuration from network devices or to add new configuration to these devices. The NETCONF protocol uses a remote procedure call (RPC) paradigm. A client encodes an RPC request in Extensible Markup Language (XML) [2] and sends it to a server using a secure, connection-oriented session. The server returns with an RPC-REPLY response encoded in XML.

The NETCONF protocol supports features required for configuration management that were lacking in other network management protocols, for instance SNMP [3]. NETCONF operates on so called datastores and represents the configuration of a device as a structured document. The protocol distinguishes between running configurations, startup configurations and candidate configurations. In addition, it provides primitives to assist with the coordination of concurrent configuration change requests and to support distributed configuration change transactions over several devices. Finally, NETCONF provides filtering mechanisms, validation capabilities, and event notification support [4].

The aim of this paper is twofold. First, we describe a NETCONF interoperability testing plan that is used to test whether NETCONF protocol implementations meet the NETCONF protocol specification in RFC 4741. The test plan particularly focuses on testing the correctness of NETCONF messages and operations; it is not our current goal to measure the performance of NETCONF implementations. Second, we will discuss the observations and results that show how the test plan found some NETCONF implementation bugs, and how it revealed a few shortcomings where the specification (RFC 4741 and RFC 4742 [5]) is either somewhat ambiguous or totally silent.

In order to make the paper concise and precise, we use the word request when we refer to an `rpc` request message and the word response when we refer to an `rpc-reply` response message. We refer to NETCONF operations such as `get-config` by typesetting the operation name in teletype font. The names of test suites are typeset in small caps, e.g., VACM.

The rest of the paper is structured as follows: An overview of the NETCONF protocol is presented in Section 2. Section 3 provides information about the systems under test before the test plan is introduced in Section 4. The NETCONF interoperability tool (NIT) is described in Section 5. Preliminary observations are reported in Section 6 before the paper concludes in Section 7.

2 NETCONF Overview

The NETCONF protocol [1] uses a simple remote procedure call (RPC) layer running over secure transports to facilitate communication between a client and a server. The Secure Shell (SSH) [6] is the mandatory secure transport that all NETCONF clients and servers are required to implement as a means of promoting interoperability [5].

The NETCONF protocol can be partitioned into four layers as shown in Figure 1. The transport protocol layer provides a secure communication path between the client and server. The RPC layer provides a mechanism for encoding RPCs. The operations layer residing on top of the RPC layer defines a set of base operations invoked as RPC methods with XML-encoded parameters to manipulate configuration state. The configuration data itself forms the content layer residing above the operations layer.

The NETCONF protocol supports multiple configuration datastores. A configuration datastore is defined as the set of configuration objects required to get a device from its initial default state into a desired operational state. The `running` datastore is present in the base model and provides the currently active configuration. In addition, NETCONF supports a `candidate` datastore, which is a buffer that can be manipulated and later committed to the `running` datastore, and a `startup` configuration datastore, which is loaded by the device as part of initialization when it reboots or reloads [4].

Table 1 shows the protocol operations that have been defined so far by the NETCONF working group of the IETF. The first two operations `get-config` and `edit-config` can be used to read and manipulate the content of a datastore. The `get-config` operation can be used to read all or parts of a specified configuration. The `edit-config` operation modifies all or part of a specified configuration datastore. Special attributes embedded in the config parameter control which parts of the configuration are created, deleted, replaced or merged.

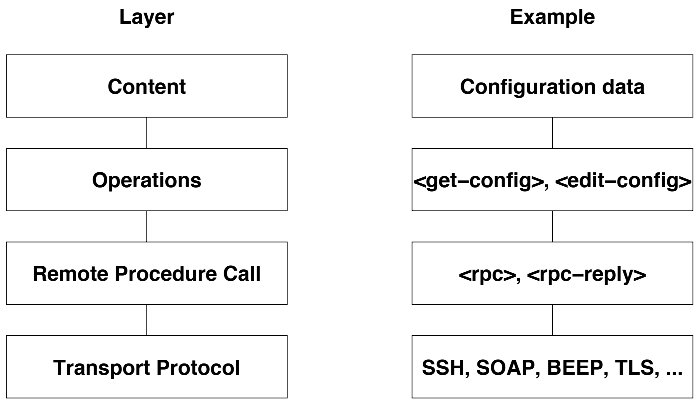


Fig. 1. NETCONF protocol layers [1]

Table 1. NETCONF protocol operations (arguments in brackets are optional) [4]

Operation	**Arguments**
get-config	source [filter]
edit-config	target [default-operation] [test-option] [error-option] config
copy-config	target source
delete-config	target
lock	target
unlock	target
get	[filter]
close-session	
kill-session	session-id
discard-changes	
validate	source
commit	[confirmed confirm-timeout]
create-subscription	[stream] [filter] [start] [stop]

The test-option and the error-option parameters control the validation and the handling of errors. The `copy-config` operation creates or replaces an entire configuration datastore with the contents of another complete configuration datastore and the `delete-config` operation deletes a configuration datastore (the `running` configuration datastore cannot be deleted).

The `lock` and `unlock` operations do coarse grain locking of a complete datastore and locks are intended to be short lived. More fine grained locking mechanisms are currently being defined in the IETF [4]. The `get` operation can be used to retrieve the running configuration and the current operational state of a device.

NETCONF sessions can be terminated using either the `close-session` operation or the `kill-session` operation. The `close-session` operation initiates a graceful close of the current session while the `kill-session` operation forces the termination of another session.

The optional `discard-changes` operation clears the candidate configuration datastore by copying the running configuration into the candidate buffer while the optional `validate` operation runs validation checks on a datastore. The optional `commit` operation is used to commit the configuration in the candidate datastore to the running datastore.

A separate specification published as RFC 5277 [7] extends the base NETCONF operations defined in RFC 4741 for notification handling. This is done by adding the `create-subscription` operation and introducing new `notification` messages carrying notification content. By using a notification stream abstraction, it is possible to receive live notifications as well as to replay recorded notifications.

NETCONF protocol introduces the notion of capabilities. A capability is some functionality that supplements the base NETCONF specification. A capability is identified by a uniform resource identifier (URI). The base capabilities are defined using URNs following the method described in RFC 3553 [8]. NETCONF peers exchange device capabilities when the session is initiated: When the NETCONF session is opened, each peer (both client and server) must send a `hello` message containing a list of that peer's capabilities. This list must include the NETCONF `:base` capability[1]. Following RFC 4741, we denote capabilities by the capability name prefixed with a colon, omitting the rest of the URI.

3 Systems Under Test

The systems used for the NETCONF interoperability testing comprise Cisco 1802 integrated services routers, Juniper J6300 routers, the Tail-f ConfD software for configuration management, and the EnSuite software [9] for configuration management. The ConfD software is an extensible development toolkit that allows users to add new components by writing a configuration specification for a data model and loading the generated object and schema files for the components. For the sake of consistency, we refer to the ConfD software as the Tail-f system. The EnSuite software contains a YencaP implementation used to test the NETCONF configuration protocol and extensible features on an experimental network management platform. It also supports web-based configuration management for NETCONF and additional modules and operations for the platform; e.g., the BGP_Module for configuring BGP routers and the Asterisk_Module for configuring VoIP servers. Table 2 briefly describes the four platforms and the SSH support of the four systems. The ConfD and EnSuite are installed and configured to run on Linux XEN virtual machines [10].

Table 3 presents the NETCONF capabilities announced by the systems under test. The Tail-f system supports all capabilities except the `:startup` capability. The Cisco, Juniper and EnSuite systems support fewer capabilities and apparently the Cisco implementation favours a distinct `startup` datastore

[1] `urn:ietf:params:netconf:base:1.0`

Table 2. Systems under test

System	Platform	SSH Support
Juniper	JUNOS ver. 9.0	ver. 1.5/2.0
Tail-f	ConfD ver. 2.5.2	ver. 2.0
Cisco	IOS ver. 12.4	ver. 2.0
EnSuite	YencaP ver. 2.1.11	ver. 2.0

Table 3. NETCONF capabilities supported by the systems under test

Capability	Juniper	Tail-f	Cisco	EnSuite
`:base`	√	√	√	√
`:writable-running`		√	√	√
`:candidate`	√	√		√
`:confirmed-commit`	√	√		
`:rollback-on-error`		√		
`:validate`	√	√		
`:startup`			√	√
`:url`	√	√	√	√
`:xpath`		√		√

while the Juniper implementation favours a `candidate` datastore with commit and rollback support. The EnSuite implementation supports both `startup` and `candidate` datastores. Note that some implementations can be configured to support additional capabilities, but we used the more standard default settings in our tests. In addition to the capabilities listed in Table 3, each system announces several proprietary capabilities.

The Tail-f and Juniper implementations use an event driven parser. They do not wait for the framing character sequence to respond to a request. The Cisco and EnSuite systems do not seem to have an event driven parser or at least they do not start processing requests until the framing character sequence has been received.

The Juniper implementation is very lenient. For example, it continues processing requests even if the client does not send a `hello` message or the client does not provide suitable XML namespace and message-id attributes. The Juniper implementation supports a large number of vendor-specific operations. In addition, it renders the returned XML content in a tree-structure that is relatively easy to read and it generates XML comments in cases of fatal errors before closing the connection. As a consequence, the Juniper implementation is very easy to use interactively for people who like to learn how things work without using tools other than a scratch pad and a cut and paste device. The EnSuite implementation shares the same characteristics with the Juniper implementation. Moreover, it returns an error message with an explanation of the reason and does not close the connection when the client sends illegal input. It, however, requires message-id attributes for requests.

The Tail-f and Cisco implementations are much less tolerant when processing input not closely following RFC 4741. They also return XML data in a compact encoding, minimizing the embedded white-space and thus reducing message sizes. Without proper tools, it is pretty difficult for humans to read the responses. In some cases, these two implementations close the connection when the client sends illegal input without an indication of the reason for closing the connection. In such cases, it can take some effort to investigate the wrongdoings.

Finally, we like to point out that the Cisco implementation we have tested does not support structured content; i.e., its configuration content is a block of proprietary IOS commands wrapped in an XML element. As a consequence, several of the advanced NETCONF features for retrieving and modifying structured configuration data cannot be applied. The EnSuite implementation still contains bugs and partially supports the `candidate` and `url` capabilities; e.g., several operations on the `candidate` datastore do not seem to work.

4 Test Plan

In this section we describe our NETCONF test plan. To make the execution of the tests efficient and to keep the collection of tests organized, we divided our test plan into five test suites. A test suite is a collection of test cases that are intended to be used to test and verify whether the systems under test meet the NETCONF protocol specification contained in RFC 4741 and RFC 4742.

Table 4 lists the test suites and the current number of test cases in each suite. The total number of test cases is 87. Each test case contains three parts: (i) a pre-configuration prepares the system under test for the test; (ii) a main test sends requests to, and receives responses from, the system under test, and verifies the responses; (iii) a post-configuration brings the system under test to the initial status. Our organization of test cases into test suites is not directly following the vertical layering model show in Figure 1 and the horizontal organization of operations and capabilities in the operations layer as one might expect. The reason is essentially our attempt to reduce the overhead of the pre-configuration and post-configuration parts during the execution of the test suite on the systems under test, e.g., in order to test the `edit-config` operation on a network interface, we describe a sequence of test cases for `create`, `replace`, `merge` and `delete` operations with setting up and cleaning up the interface once. This also led to a more tightly integrated organization of the test cases.

The first test suite is referred as the GENERAL test suite because it includes test cases for individual operations such as `lock`, `unlock`, `close-session`, `kill-session`, `discard-changes`, `validate`, and `commit`. The `lock` and `unlock`

Table 4. Test suites and current number of test cases

Test Suite	No. Test Cases
GENERAL	19
GET	11
GET-CONFIG	16
EDIT-CONFIG	15
VACM	26

test cases verify that the responses do not contain an error or the responses contain a proper error, e.g., a `lock` request to the datastore already locked causes an error. The `kill-session` test case contains a pre-configuration that prepares another running session before terminating it, while the `validate` test case contains a post-configuration that discards a change after validating it. This test suite also checks the format of requests and responses. Few test cases verify whether the responses contain the compulsory attributes and the attribute's value matches the value contained in the requests.

The next two test suites are the GET and GET-CONFIG suites. These suites aim to test the filter mechanism of the `get` and `get-config` operations. While `get` operates on the `running` configuration datastore and the device's state data, `get-config` operates on different sources of the configuration data such as the `running` and `candidate` datastores (depending on the support of capabilities), resulting in additional test cases for the GET-CONFIG suite. Test cases verify several types of subtree filters, e.g., a test case checks whether the system under test returns the entire content of the running configuration data plus the operational state when no filter is used, or another test case checks whether the system under test returns nothing when an empty filter is used.

The EDIT-CONFIG suite involves tests modifying the configuration data in the datastore. This suite includes test cases for the `delete-config`, `copy-config`, and `edit-config` operations. The `edit-config` operation test cases support the `create`, `replace`, `merge` and `delete` operation attributes. Several test cases in this suite are data model specific due to the lack of a common data model, thus we need to implement several tests in different ways. This extra work can be reduced if implementers volunteer to support a common data model.

The last test suite is the VACM suite verifying the NETCONF protocol operations against the VACM data model [11]. This data model is a YANG version of the `SNMP-VIEW-BASED-ACM-MIB` (View-based Access Control Model for the Simple Network Management Protocol [12]). YANG [13] is a data modelling language for NETCONF. Test cases in this suite are generated from this data model focusing on the `group`, `access`, and `view` lists.

5 Test Tool (NIT)

We have implemented a tool called NIT (NETCONF Interoperability Testing tool) to automatically execute the test suites against a system under test. Our NIT tool basically performs the following operations:

1. connecting to a system under test using the SSH
2. verifying the initial `hello` message
3. executing test cases by
   - sending a test request and receiving a response
   - verifying both the request and the response following the criteria defined by RFC 4741.
4. reporting the failure or the success of each test

The tool is equipped with an XML parser to analyze the responses for verification. The parser, upon receiving a response, provides a list of elements with quantity, a list of attributes with quantity, a list of attribute values and a list of text parts. With this information, the tool can detect possible flaws from the responses, such as whether any element or attribute is missing, whether an error must be returned. The following example presents a response without an error or a warning:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="1007"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

The parser provides the following information of the response:

```
-Element Types
    rpc-reply 1
           ok 1
-Attribute Types
   message-id 1
        xmlns 1
-Attribute Values
   message-id ['1007']
        xmlns ['urn:ietf:params:xml:ns:netconf:base:1.0']
-Text Parts
              []
```

We have used the Python unit testing framework [14]. The framework features test automation, shared configuration of setup and shutdown methods, arrangement of tests into collections, and independent reporting of the tests. The tool takes advantage of these features to maintain a single SSH connection for all tests and to group related tests into a collection; e.g., tests concerned with creation, modification and deletion operations are grouped together to re-use and clean the testing environment easily. The tool organizes test cases into several collections of test cases, namely test suites, as discussed in Section 4.

While the tool has been used successfully to test some specific devices (see next section), it possesses several limitations. Firstly, it lacks a resumption mechanism to continue the test run when it encounters connection loss due to the misbehavior of systems under test. Secondly, while the test cases are believed to comply with RFC 4741, the test scripts, i.e., the piece of code that implements test cases, depends on the specification and configuration of components of the tested systems to produce the requests and to verify the responses. Finally, the framework requires some extra work for complicated test cases; e.g., testing the `lock` operation requires an extra session to lock the database.

6 Preliminary Observations

We have used the NIT tool to test the systems described in Section 3. Since the result of the tests are specific to the different NETCONF implementations, we present the results by referring to system X and we leave out the mapping of X to the systems described in Section 3. Note that we did manually re-check the failed test cases in order to erase bugs in the test scripts. Despite these efforts, several test cases reflect our interpretation of RFC 4741 and there might not be full agreement with our interpretation and thus the numeric results presented below should be taken with a grain of salt.

Table 5. Test result summary organized by the systems under test

System	**Success**	**Failure**	**Irrelevant**
A	47.2%	14.9%	37.9%
B	82.8%	9.2%	8.0%
C	17.3%	10.3%	72.4%
D	17.3%	21.8%	60.9%

Table 5 presents the result of the NIT tool for the systems under test. The "success" and "failure" columns indicate the percentage of passed and failed test cases respectively, while the "irrelevant" column indicates the percentage of test cases that cannot be applied to a specific system due to either system configuration or implementation problems (e.g., the `vacm` data model is not implemented).

We learned that the systems A and B comply reasonably well with the RFCs. The system A fails 14.9% of the test cases and most of them are related to the basic format of request and response messages or the filter mechanism of the `get` operation. The system B performs better with very few failed test cases and most of them are concerned with the validation of XML elements in request messages. The two systems A and B have very few problems with the filter mechanism of the `get-config` operation or the usage of the `edit-config` operation for creating, modifying and deleting configuration elements. The systems C and D perform poorer with 72.4% and 60.9% irrelevant test cases and 10.3% and 21.8% failed test cases, respectively. The failed test cases are related to the format of requests and responses or the filter mechanism of the `get` operation.

Table 6 reports the passed and failed test cases organized by the test suites over the total number of running test cases for the systems under test. There are two remarks: (i) the GET suite obtains a high percentage of failed test cases 52.3%, and (ii) the EDIT-CONFIG suites obtains low percents of failed test cases 1.7%. We found that the majority of failed test cases from the GET suite is related to the filter mechanism of the `get` operation.

With the failed test cases in mind, we have looked back into the RFCs. There are several things where the RFC is either somewhat ambiguous or totally silent. In general, the RFC should provide more detailed descriptions for error situations and it might be necessary to better constrain the currently open ended format

Table 6. Test result summary organized by the test suites

Test Suite	Success	Failure	Irrelevant
GENERAL	73.6%	13.2%	13.2%
GET	29.5%	52.3%	18.2%
GET-CONFIG	48.4%	14.1%	37.5%
EDIT-CONFIG	38.3%	1.7%	60%
VACM	19.2%	5.8%	75%

of request and response messages since they for example allow arbitrary values for attributes. Furthermore, the RFC should be updated with clearer examples. Some particular issues are listed below:

– The RFC ignores the XML declaration

  `<?xml version$="1.0" encoding="UTF-8"?>`

  for requests and responses. Some systems do not execute a request without this declaration while other systems do. It seems that the IETF working group favours to have a mandatory XML declaration.
– The examples in RFC 4741 often omit namespace declarations for request and response messages. Only few systems execute a request without a proper namespace declaration and it would help interoperability if the examples would contain namespace declarations where necessary.
– RFC 4741 requires that additional attributes present in the `<rpc>` element of a request message must be returned in the `<rpc-reply>` element of the response message without any change (see section 4.1 of the RFC 4741). This requirement leads to problems when such an attribute conflicts with attributes generated by the implementation. One implementation generated duplicated attributes (and thus invalid XML) while another implementation removes a duplicated attribute resulting in violation of RFC 4741.
– RFC 4741 allows arbitrary strings for the `message-id` attribute. From the tests, we found that implementations terminate the session often without an error indication or return strange results when the `message-id` attribute in a request message contains unexpected content such as the literal string `]]>]]>` or the literal string `</rpc>`. Of course, a proper NETCONF client would not generate such request messages since they are invalid XML. But on the other hand, one can question whether arbitrary content in request and response attributes is a feature worth to support.

Some of the items listed above are meanwhile actively discussed on the NETCONF working group mailing list and work is underway to revise RFC 4741 in order to fix bugs and to clarify the processing of NETCONF messages [15].

7 Conclusions and Future Work

We have carried out some work on NETCONF interoperability testing. This work aims at observing the compliance of NETCONF implementations with

RFC 4741. It also aims at identifying inconsistencies in the RFC. We have proposed a test plan consisting of five test suites. Each test suite contains a number of test cases that involve a single operation or a group of related operations. The test cases exploit several aspects of RFC 4741 including the format of request and response messages, the filter mechanism supported by some operations, NETCONF capabilities, and so on. The test cases have been coded into the NIT tool, which automates the execution of test runs. It should be noted, however, that the test cases so far have not been reviewed and as such there might be disagreement on some test cases whether they are correct or not relative to RFC 4741.

We have used the NIT tool to test four different NETCONF implementations. Our preliminary observations indicate that the number of failed test cases is relatively high for some systems, thus raising the question of the compliance of these systems with RFC 4741. We have also noted some inconsistencies in RFC 4741 that should be addressed in a future revision of this document. It should be mentioned that some test cases are our interpretation of RFC 4741 and it needs to be worked out to what extend our interpretation meets the interpretation of the working group.

While some interesting initial results have been obtained, this work still requires several improvements. First, the coverage of RFC 4741 by the test cases needs to be evaluated and increased by adding additional test cases as needed. Furthermore, it would be nice to reduce the dependency of the test cases on different data models. Third, the NIT tool should be improved to better support more complicated test cases that involve multiple NETCONF sessions. Fourth, it would be nice to have a tool able to generate test suites out of YANG data models. And finally, it would be valuable to repeat the tests with a larger number of different NETCONF implementations and to evaluate how test results impact future software revisions and lead to more interoperability.

Acknowledgment

The work reported in this paper is supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Enns, R.: NETCONF Configuration Protocol. RFC 4741 (December 2006)
2. Sperberg-McQueen, C., Paoli, J., Maler, E., Bray, T.: Extensible Markup Language (XML) 1.0, 2nd edn. (October 2000), `http://www.w3.org/TR/2000/REC-xml-20001006` (last access in July 2008)
3. Case, J., Mundy, R., Partain, D., Stewart, B.: Introduction and Applicability Statements for Internet Standard Management Framework. RFC 3410 (December 2002)
4. Schönwälder, J., Björklund, M., Shafer, P.: Configuration Management using NETCONF and YANG. IEEE Communications Magazine (2009)
5. Wasserman, M., Goddard, T.: Using the NETCONF Configuration Protocol over Secure Shell (SSH). RFC 4742 (December 2006)

6. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251 (January 2006)
7. Chisholm, S., Trevino, H.: NETCONF Event Notifications. RFC 5277 (July 2008)
8. Mealling, M., Masinter, L., Hardie, T., Klyne, G.: An IETF URN Sub-namespace for Registered Protocol Parameters. RFC 3553 (June 2003)
9. Cridlig, V., Abdelnur, H.J., Bourdellon, J., State, R.: A NetConf Network Management Suite: ENSUITE. In: Magedanz, T., Madeira, E.R.M., Dini, P. (eds.) IPOM 2005. LNCS, vol. 3751, pp. 152–161. Springer, Heidelberg (2005)
10. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the Art of Virtualization. In: Proc. 19th ACM Symposium on Operating Systems Principles (SOSP 2003). ACM, New York (2003)
11. Schönwälder, J.: VACM Yang Data Model. Jacobs University Bremen (October 2008)
12. Wijnen, B., Presuhn, R., McCloghrie, K.: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). RFC 3415 (December 2002)
13. Björklund, M.: YANG - A data modeling language for NETCONF. Internet draft (January 2009)
14. Python Unit Testing Framework, `http://pyunit.sourceforge.net/` (last access in November 2008)
15. Enns, R., Björklund, M., Schönwälder, J.: NETCONF Configuration Protocol. Internet Draft <draft-ietf-netconf-4741bis-00.txt>, Juniper Networks, Tail-f Systems, Jacobs University (March 2009)

Knowledge Management and Promises

Mark Burgess

Oslo University College, Norway
mark@iu.hio.no

Abstract. Ontological modelling for machine inference has featured prominently in IT management research recently, but there is more immediate scope for knowledge modelling in the realm of human inference. This work discusses the relationship between ISO Standard Topic Maps and Promise Theory and shows how these two knowledge models models complement one another and offer a *semantic* approach to policy based management that can reduce organizational information complexity.

1 Introduction

Ontological modelling has been discussed at length as a way of using domain knowledge to enhance web services for IT management. Ontology models typically involve complex XML markups intended for such smart web interfaces [1, 2, 3, 4] but do little to aid human understanding. Unfortunately, semantic inference in automatic systems is beset with difficulties: the overhead in both modelling and processing tends to lead to benefits that are either simplistic to humans or have problems that are intractable to machines. The reason for this seems to be that semantics are the domain of human imagination, not of machine logic. This suggests a return to a division of labour between humans and machines in which each party does what is most natural to it. Humans are good at reasoning and associative thinking when provided with quality information, while machines are good at consistent, repetitive implementation and rather poor at reasoning. This is the starting point for this work.

Semantic modelling of information pre-dates the Web by several years in fact. Topic Maps, discussed in this work, were originally invented as a form of electronic book-index, enhanced with associative thinking [5]. They have since fallen into the shadow of efforts surrounding the semantic web research, but wrongly in the opinion of the author. Topic maps appear simpler than the general ontology languages of the semantic web and they are designed for human appraisal rather than machine inference. Topic maps also have a simple relationship with the theory of promises, which this paper aims to make use of.

2 Promises and Topics

The concept of promises was introduced into IT management in 2005 as a fresh approach to the problems of conflicts in policy based management [6]. The heuristics and extended motivation for the model were described later in [7].

Promises provide a model for abstracting the intention behind operations from the operations themselves, and have attractively simple algebraic and semantic properties. The core of promise theory is the *autonomy* of agents that are able to make promises, and the subsequent notion of *voluntary cooperation*, replacing the problematic notion of obligation. This leads to many theoretical and practical simplifications. The reference implementation of promise theory as a technology is Cfengine [8], which forms an integral part of the discussion below. Every statement made in the essentially declarative cfengine language can be understood as being part of a promise.

This declarative manifesto, backed up by practical guarantees of outcome reachability [9], has an important implication for automated management: the complete separation of intent from action means that what remains for the human is to model *intent*. This is a form of knowledge management, and thus the focus moves from implementation to knowledge.

Knowledge management is a field of research in its own right, and it covers a multitude of issues both human and technological. Most would agree that knowledge is composed of facts and relationships and that there is a need both for clear definitions and semantic context to interpret knowledge properly; but how do we attach *meaning* to raw *information* without ambiguity? This is an ad hoc association which follows human social conventions, and is therefore poorly suited to machine reasoning.

Knowledge has much in common with configuration: what after all is knowledge but a configuration of ideas in our minds, or on some representation medium (paper, silicon etc). It is a coded pattern, preferably one that we can agree on and share with others. Both knowledge and configuration management are about describing patterns.

Previous models of management have been based on pure data modelling and the assumption 'guaranteed' change in accordance with the data [1, 10]. What makes Topic Maps attractive compared to other more complex ontology tools is that they are intended for human reasoning (something humans are very good at), not for machine inference (which is something machines have rarely been very good at). Moreover, although they sound like very different animals, Topic Maps and Promises are in fact homomorphic and complement one another in a neat, symbiotic relationship.

The reasoning may be summarized as follows: a simple knowledge model can be used to represent a simple policy configuration model; conversely, a simple model of policy configuration can represent indeed manufacture a knowledge structure, and there is a natural promise engine that can implement this mapping: cfengine.

The Topic Map and Promise models are compatible because they appeal to the same basic world-view: principles for reduction of knowledge into atoms and the autonomy of concepts that automatically avoids overlap and conflict. Both models effectively use the idea of autonomy of entities and a simple context based data model that allows them both to represent their subjects, as well as one another, in a homomorphic way.

3 The Promise Model

3.1 Promise Theory

Promises are a modelling framework (see [6]) that presents a decentralized view of behaviour in systems. Promise theory describes the intentions and attributes of system artefacts (i.e. anything from system components to ideas), which are autonomous in the sense that they can change independently.

A promise is the announcement of an intention [7] (usually expected to represent a possible future) and it requires verification to confirm eventual compliance. Promises are not events but conditions (states) that persist in the memory of recipients who are in the scope of the promise. A promise is more than an intention, since an intention need not be announced, and it is less than a commitment since a commitment often involves an investment or action plan for keeping the promise.

Promises are made by a promiser 'agent' to a promisee 'agent', i.e. they are directed relationships each labelled with a promise *body* which describes the substance of the promise. A promise with body $+b$ is understood to be a declaration to "give" behaviour from one agent to another (possibly in the manner of a service), while a promise with body $-b$ is a specification of what behaviour will be received, accepted or "used" by one agent from another (see table 1). A promise *valuation* $v_i\left(a_j \xrightarrow{b} a_k\right)$ is a subjective interpretation by agent a_i (in a currency of its choice) of the value of the promise in the parentheses; this can be used for ranking of importance, for example. The value can be negative if it is pure cost. Usually an agent can only evaluate promises in which it is involved.

A promise body b has a *type* which describes the nature or subject of the promise, and a *constraint* which explains what restricted subset of the total possible degrees of freedom are being promised. Since any dynamical, systematic behaviour is a balance between degrees of freedom (avenues for change) and constraints, this is sufficient to describe a wide variety of phenomena.

Promise theory is mainly about the analysis of epochs in which promises are essentially fixed. If basic promises change, we enter a new epoch of the system in which basic behaviours change. Thus promise theory is mainly about steady-state behaviour about which one can accumulate lasting knowledge.

Table 1. Summary or promise notation

Symbol	Interpretation
$a \xrightarrow{+b} a'$	Promise with body b
$a' \xrightarrow{-b} a$	Promise to accept b
$v_a(a \xrightarrow{b} a')$	The value of promise to a
$v_{a'}(a \xrightarrow{b} a')$	The value of promise to a'

3.2 Promises in Cfengine

Cfengine 3 is the reference implementation of promise theory as a technology for configuration. It allows promises to be expressed as a language and kept by software automation. Cfengine's representation of a promise has the following generic form:

```
promise-type:

   context-classifiers::

      promiser-object -> { list of possible promisees },

         comment => ‘‘Expression of intention’’,
         body-attribute-1 => value-1,
         ...
         body-attribute-n => value-n;
```

Such declarations are grouped into ‘promise bundles’. Such a statement encodes a single promise, for example:

```
files:

   linux||solaris::

      "/etc/shadow" -> { "ISO17799 team", "other promise" },

         comment => "Check integrity of the shadow file",
         changes => record_hash_changes(),
         perms   => my_perms("root","0600");
```

This partially disclosed promise is directed at a team of humans and at another promise that depends on this one. It concerns the file `/etc/shadow`. The scope or context of the promise is the set of all agents that belong to the classes `linux` or `solaris`, and the body of the promise contains the properties that the file should have: namely a particular owner, a particular set of access rights to the file and a static hash signature.

Cfengine views this initial declaration as a *promise proposal*, which is intended to apply to any host or agent in scope. The declaration of the suggested promise is typically made at some central location where management is centralized.

Other hosts that are not represented in the scope are supposed to ignore the promise proposal, but hosts that lie in these classes will normally take this promise proposal at face value and try to keep it, as if it were a command, although there is no way of actually forcing them to do this. Indeed this voluntary behaviour represent another kind of retractable promise, namely one to accept these suggestions and implement them in the first place. At every step, the cooperation by agents is voluntary, but the effect is to set up an entirely prosaic workflow.

There are many different types of promise that one can make in cfengine. The engine is extensible and different agents are provided to keep different kinds of promises. The component `cf-agent`, for instance, can make promises to configure the resources of computers. With the advent of cfengine 3, a knowledge management component was introduced called `cf-know`, in which topic map relationships could be promised. Cfengine forms the basis by which one might integrate the management of declarative knowledge about configuration promises with the intentions and implementations of the declarations.

4 The Topic Map Model

4.1 Introduction

Let us now examine the topic map model. Topic maps were originally designed for creating generalized electronic book-indices. They pre-date the World Wide Web by several years yet they work effectively as a *semantic web* of subject references and document pointers. The basic model is effectively described in terms of the TAO: Topics, Associations and Occurrences [11].

A topic is representation of any subject one wishes to discuss, abstract or physical e.g. an item of 'abstract knowledge', which might have a number of exemplars. It might be a person, a machine, a quality, etc. Topics may be classified into *topic-types* so that related things can be collated and unrelated things can be separated, e.g. types allow one to distinguish between `rmdir` the Unix utility and rmdir the Unix system-call. Each typed topic can further point to a number of exemplars called *occurrences*, which might include documents, database entries, physical manifestations and other information references that exemplify or are about the topic. Occurrence references are like the page numbers in an index. Unlike an ordinary index, a topic map has a rich (potentially infinite) variety of cross reference types.

A book index typically has 'see also' to refer from one topic to another. Topic Maps allow one to define any kind of *association* between topics. For instance,

```
topic_1 ``is a kind of'' topic_2
topic_1 ``is improved by'' topic 2
topic_1 ``solves the problem of'' topic_2
```

and so on.

The topic map model thus has three levels, in order: types, topics and occurrences. These all label different levels of granularity. Types are exemplified by topics, which in turn are exemplified by occurrences (though in a different way). The primacy of topics in this hierarchy stems from their ability to form networks. Thus, while topic-type classifications and occurrences are disjoint entities, topics willfully connect in a space of associative interconnectedness.

The classic approach to information modelling is to build a hierarchical decomposition of non-overlapping objects. Entity relation models and object oriented class hierarchies force all occurrences of data to lies in a rigid information model.

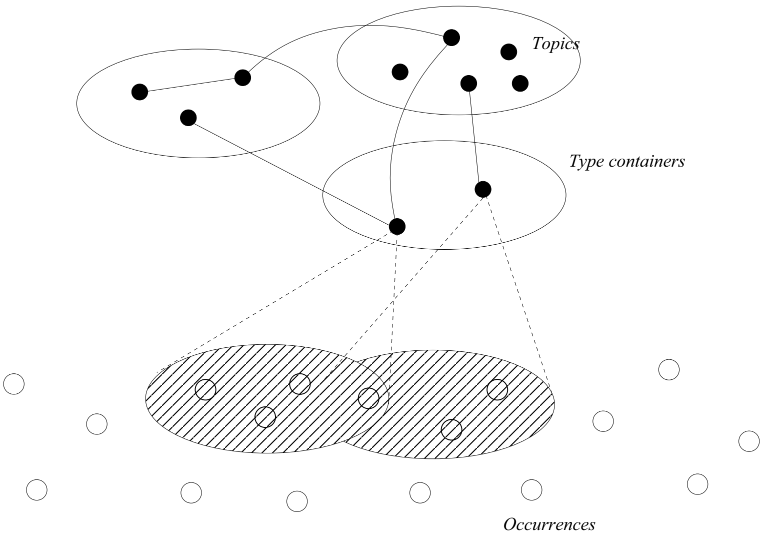


Fig. 1. Topic maps as a concept transducer: abstract topics live inside abstract classifiers or types, but can network independently. Topics 'shine different lights' on the concepts they represent called occurrences.

The data are forcibly manipulated into non-overlapping containers which often prove to be overly restrictive (cf. the need for aspect orientation and 'friends' in Object Orientation etc).

Each topic allows us to effectively 'shine a light' onto the occurrences of information that highlight the concepts pertinent to the topic somehow (see fig. 1). Formally topic maps use the term 'occurrence-types' to label individual topic's relationships to (viewpoints on) their occurrences; *occurrence relationship types* are not necessarily sub-types of the topics. Topics and occurrences are not classified into Draconian disjoint sets by their types: indeed, this is what allows topic maps to exceed simple hierarchical data modelling. Topic maps (networks of associated topics) bridge the world of concepts and exemplars by working as a 'concept transducer' that shines multiple lights onto the real world of occurrences.

The topic map model is an ISO standard that is quite rich in possibilities [5]. As a framework, the model has a simplicity to address the real problems of information complexity, but it lacks an operational road-map for usage. The same can be said about cfengine, which also provides a framework without a road-map. The key observation to integrating topic with promises is this: topic maps can represent the knowledge declared in a set of promises. They can model the relationships between the parts and types of a promise, they can point to occurrences of the promise made by multiple parties and they can discuss the abstract intentions of the promises with references to other literature. Promises on the other hand can describe how to build a topic map, its configuration and its maintenance. There is thus a natural duality between topic maps and promises.

4.2 The Cfengine Topic Map Model

Cfengine is a 'self-healing' management automation system, meaning that it can adapt to change and repair errors without human intervention. it was created by the author in 1993 [12]. It currently consists of a number of components that run on each individual (i.e. autonomous) computer in a network, and each computer typically voluntarily collects promise suggestions from a single point of management. The components are all able to make promises, e.g. `cf-agent` can make (and keep) promises about configuration changes, while `cf-monitord` can make promises about system data collection, etc. These components integrate to form maintenance loops.

Cfengine's knowledge agent `cf-know` makes promises about knowledge relationships, using the model of topic maps. It is not a generic topic map language: it provides a configuration language for managing a knowledge base that can be compiled into a topic map. The full ISO standard topic map model is sufficiently rich to capture a general topic index, indeed it is almost too rich to be a useful tool for system knowledge management. However, this is where powerful configuration management can help to simplify the process: encoding a topic map is a complex problem in configuration, which is exactly what cfengine is for. Cfengine's topic map promises have the following form:

```
topics:

  topic_type_context::                                # canonical container

    "Topic name"                                      # short topic name

            comment => "Use this for a longer description",
        association => a("forward assoc to","Other topic","backward assoc");

     "Other topic";

occurrences:

  Topic_name::                                 # Topic

    "http://www.example.org/document.xyz"      # URI to instance

      represents => { "Definition", "Tutorial"}; # sub-types
```

A topic declaration involves a *type*, in topic map parlance, which maps to a scope or class context in cfengine. The topic-type is itself a reference to a container topic, since a type is also a topic. Topics of given types form disjoint sets; however topics of the same name may exist in several types, e.g. Cfengine (the software) or Cfengine (the company). Each topic effectively promises to have a name and a number of associations with other topics. In the cfengine mapping, this is made explicit. The type of a topic is simply the cfengine *class* canonicalization of the topic name used as a scope; in practice this means converting non alphanumeric characters into underscores and adding the double-colon.

The distinction between topic and topic-type is to some extent immaterial in a topic map; at one level, a type relationship is just another kind of association between topics. However, type is used differently: its describes disjoint *classifiers* (cfengine is an instance of software, or cfengine is an instance of company), where as associations are used more generally at the conceptual level to link topics of any type into a network without boundaries (cfengine 'is used to implement' promises, or promises 'may be used to configure' a topic map).

Cfengine's rendition of Topic Maps is simplified even further. It does not implement the full ISO standard, but rather a subset that is necessary and sufficient to be isomorphic with promises. Promise theory adds a clear structure to the topic map ontology, which is highly beneficial as experience shows that weak conceptual models lead to poor knowledge maps. The result is a language for making simple topic maps which (although it does not support the entire ISO standard) is both simpler and adds powerful features allowing variable expansion, re-use and more structured bundling of data.

5 Modelling Configuration Promises as Topic Maps

We can model topic maps as promises within cfengine; the question then remains as to how to use topic maps to model configurations so that cfengine users can navigate the documented promises using a web browser and be able to see all of the relationships between otherwise isolated and fragmentary rules. This will form the basis of a semantic Configuration Management Database [13] (sCMDB) for the cfengine software. The key to making these ends meet is to see the configuration of the topic map as a number f promises made in the abstract space of topics and the turning each promise into a meta-promise that models the configuration as a topic with attendant associations. Consider the following cfengine promise.

```
bundle agent update
{
files:

 any::

   ‘‘/var/cfengine/inputs’’ -> { ‘‘policy_team’’, ’’dependent’’ },

            comment => ‘‘Check policy updates from source’’,
              perms => true,
               mode => 600,
          copy_from => true,
        copy_source => /policy/masterfiles,
            compare => digest,
       depth_search => true,
              depth => inf,
           ifelapsed => 1;

}
```

This system configuration promise can be mapped by cfengine into a number of other promise proposals intended for the `cf-know` agent. Suppressing some of the details, we have: Note that in this mapping, the actual promise (viewed as a

```
type_files::

  "/var/cfengine/inputs"
      association => a("promise made in bundle","update","bundle contains promise");
  "/var/cfengine/inputs"
      association => a("specifies body type","perms","is specified in");
  "/var/cfengine/inputs"
      association => a("specifies body type","mode","is specified in");
  "/var/cfengine/inputs"
      association => a("specifies body type","copy_from","is specified in");

   # etc ...

 occurrences:

   _var_cfengine_inputs::

    "promise_output_common.html#promise__var_cfengine_inputs_update_cf_13"
       represents => { "promise definition" };
```

real world entity) is an occurrence of the topic ‘promise’; at the same time each promise could be discussed as a different topic allowing meta-modelling of the entity-relation model in the real-world data. Conversely the topics themselves become configuration items or ‘promisers’ in the promise model. The effect is to create a navigable semantic web for traversing the policy; this documents the structure and intention of the policy using a small ontology of standard concepts and can be extended indefinitely by human domain experts (see [14]).

We end up with the following mappings, which because they are based on a model that is both simple and rigid is guaranteed to lead to densely connected networks. The two concepts may be compared as tuples:

$$\begin{aligned} \mathit{Promise} : \langle P_r, P_e^*, B\rangle, &\quad B = \langle L, V\rangle^* \\ \mathit{Topic\ map} : \langle T, A^*, O^*\rangle, &\quad A = \langle L, T_A\rangle^* \end{aligned} \tag{1}$$

For promises: P_r is the promiser, P_e^* is a number of promisees (all of which are autonomous entities). B is the body of constraints that describes the promise's intent. This in turn consists of pairs of associations between names (l-values) L of constrainable properties and the values V to be adhered to. For topic maps: T is a topic, which is an autonomous, standalone entity. A is a body of associations to other topics in topic-space, and O^* is a number of associated occurrences of the topic in the documentation/information medium. A body of associations is a set of pairs of association labels or types L and topics $T_A \in T$. There is no principled difference between associations between topics and occurrence relations, except the sets or spaces to which the end-points belong. In mapping to promises we would use associations to map concepts discussed in promises, and occurrences to map to the instances of rules in a policy.

Table 2 shows this mapping of concepts in words.

Table 2. Mapping between promises and topic map concepts

Promise theory	Topic maps
Promise of type topics	Topic
Promise of type occurrences	Occurrences
Promiser P	Topic name
Promise context	Topic-type or classifier
Promise body	Associations
Promisees	Special associations

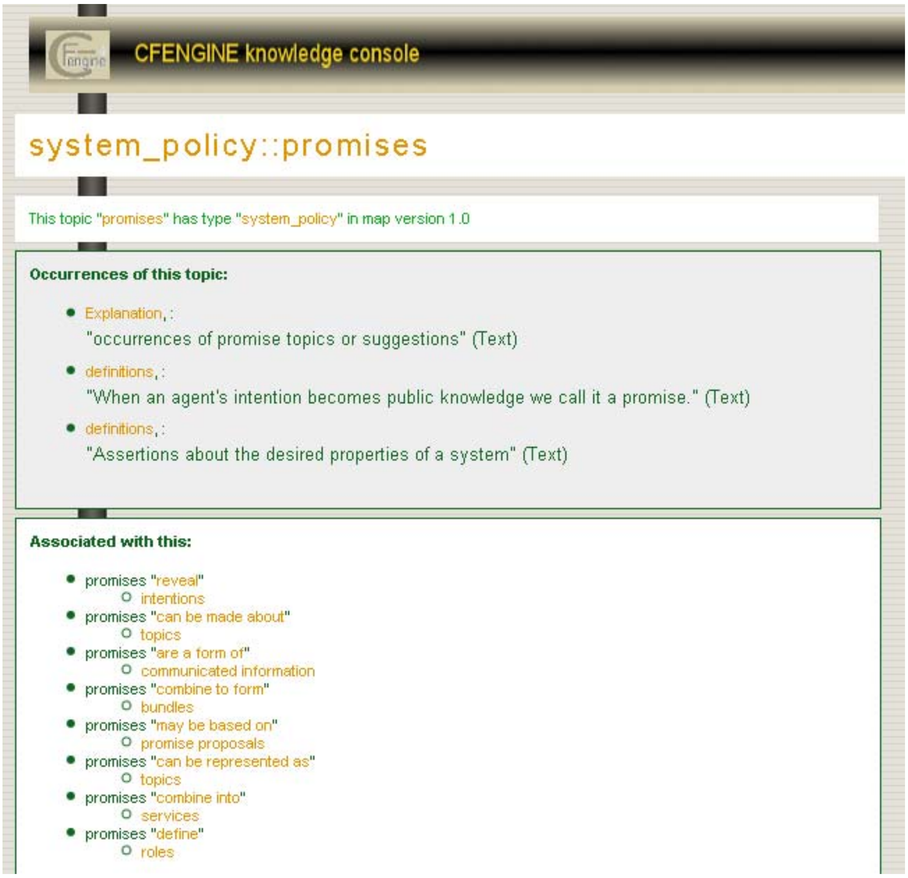
6 Knowledge about Promises

Cfengine is able to construct the topic map and the promise graph and perform analyses of these. Figure 3 shows a simple top-level example of a navigable information structure computed by cfengine from a few topics relating to this paper (with association labels removed). Imagine the same idea as a discussion of the meaning intended in policy.

Figure 2 shows part of a page rendered by `cf-know` about the subject of promises. It should various interpretations that are promised in the configuration, and their associations to related topics. Further down this page (not shown) are links to topics of *type* promise. Following one of these links takes us to fig. 3, which is a configuration promise that is actually encoded as policy for `cf-agent`. The upper graph shows the thirty or so most closely associated topics (by any association), and the lower (truncated) graph picks out only *promisee*, and `depends_on` constraints, thus automatically generating a possible impact analysis for changes to any of these promises.

Graphical representations of information open up all kinds of analyses and allow imaginative humans to see possibilities in a way that only confuses machine reasoning systems. One challenge in knowledge management is that of reducing information complexity. Even in a policy specification, the rule sets (promises) can run to thousands of lines and might contain dependencies and relationships that are not obvious to the reader. With the Topic-Map-Promise alliance, one can automatically generate a topic map from a promise-based policy and then annotate it and link it to other information bases, effectively forming a set of adaptive container classes. The chief advantage of topic maps is that they are non-hierarchical and their categories are adaptive and context dependent in the links between topic and occurrence – this is the same as cfengine's promise model and reduces the depth of information structures. Moreover there will be no categorization conflicts in either promises or topic maps by design. This sets them apart from obligation systems and Object Oriented data models.

Nevertheless, any information model that has typed elements is two dimensional and must trade complexity in the number of types and sub-types (depth) against the number of topics in each type category (breadth) and this is a fundamental problem of all information systems. However, there is a way out: the free networking of associations in topic maps means one can form standard one-dimensional pathways (routing solutions) through the associative network, which



system_policy::promises

This topic "promises" has type "system_policy" in map version 1.0

Occurrences of this topic:

- Explanation, :
  "occurrences of promise topics or suggestions" (Text)
- definitions, :
  "When an agent's intention becomes public knowledge we call it a promise." (Text)
- definitions, :
  "Assertions about the desired properties of a system" (Text)

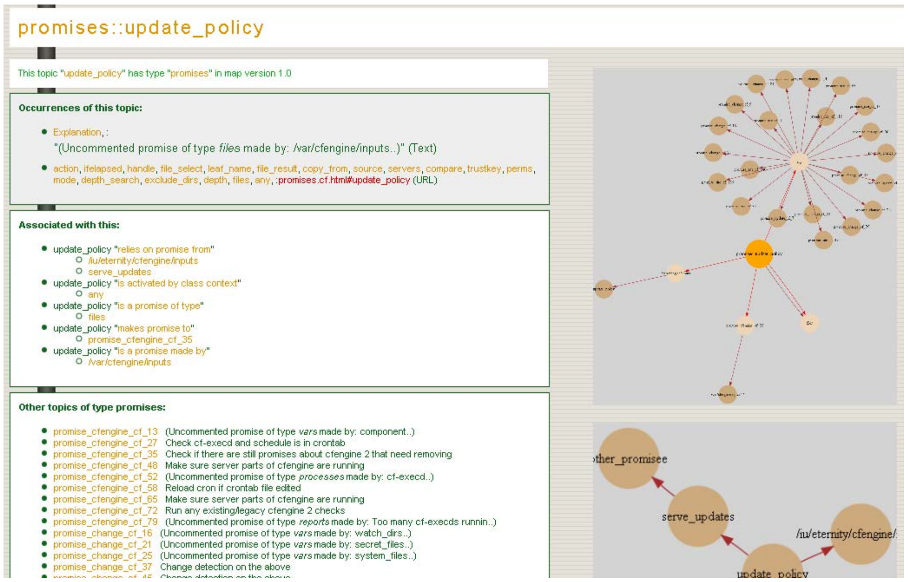
Associated with this:

- promises "reveal"
  - intentions
- promises "can be made about"
  - topics
- promises "are a form of"
  - communicated information
- promises "combine to form"
  - bundles
- promises "may be based on"
  - promise proposals
- promises "can be represented as"
  - topics
- promises "combine into"
  - services
- promises "define"
  - roles

Fig. 2. An excerpt of a topic page about promises, showing several occurrences of interpretation of the concept, and their associative links to related subjects

we might call ‘stories’. In ref. [15] we develop this idea to find minimum distance connections between topics that are modelled for human consumption. Thus the notion of topic maps extends to include ‘typed stories’ based on the idea of transitivity rules for semantic associations. The homomorphism with promises further implies that there must now be a corresponding structure in promise theory: it is *processes* or *work-flows*. Space forces us to refer to this elsewhere [15].

A final possibility of this work is a fundamental redesign of archaic models like the ‘CMDB’. Cfengine’s answer to the Configuration Management Database (CMDB) is not a traditional inventory system like most present day solutions, but rather a knowledge-based semantic web of information that links database records (occurrences) to manuals, papers and enterprise level policies through the *concepts* they employ. Without the promise concept such a task is very difficult indeed, but using Promise Theory and Topic Maps such a modern information base can now be built.



promises::update_policy

This topic "update_policy" has type "promises" in map version 1.0

Occurrences of this topic:

- Explanation, :
  "(Uncommented promise of type *files* made by: /var/cfengine/inputs..)" (Text)
- action, ifelapsed, handle, file_select, leaf_name, file_result, copy_from, source, servers, compare, trustkey, perms, mode, depth_search, exclude_dirs, depth, files, any, :promises.cf.html#update_policy (URL)

Associated with this:

- update_policy "relies on promise from"
  - /iu/eternity/cfengine/inputs
  - serve_updates
- update_policy "is activated by class context"
  - any
- update_policy "is a promise of type"
  - files
- update_policy "makes promise to"
  - promise_cfengine_cf_35
- update_policy "is a promise made by"
  - /var/cfengine/inputs

Other topics of type promises:

- promise_cfengine_cf_13 (Uncommented promise of type *vars* made by: component..)
- promise_cfengine_cf_27 Check cf-execd and schedule is in crontab
- promise_cfengine_cf_35 Check if there are still promises about cfengine 2 that need removing
- promise_cfengine_cf_48 Make sure server parts of cfengine are running
- promise_cfengine_cf_52 (Uncommented promise of type *processes* made by: cf-execd..)
- promise_cfengine_cf_58 Reload cron if crontab file edited
- promise_cfengine_cf_65 Make sure server parts of cfengine are running
- promise_cfengine_cf_72 Run any existing/legacy cfengine 2 checks
- promise_cfengine_cf_79 (Uncommented promise of type *reports* made by: Too many cf-execds runnin..)
- promise_change_cf_16 (Uncommented promise of type *vars* made by: watch_dirs..)
- promise_change_cf_21 (Uncommented promise of type *vars* made by: secret_files..)
- promise_change_cf_25 (Uncommented promise of type *vars* made by: system_files..)
- promise_change_cf_37 Change detection on the above

Fig. 3. A topic page about a policy item rendered by `cf-know`, showing relationships between neighbouring concepts and the subset of these that gives dependency or change-impact analysis

7 Conclusions

This work shows that there is a two-way mapping between promises and topic maps that enables a simple formal representation of human understanding to be codified. This may be used to annotate policy with meaning and intention, and navigate it efficiently without hierarchical complexity. Using a Promise Theory framework, system policy can be expressed directly in the form of low level *intentions* whose implementation can be promised by cfengine for any initial state of the system. Automation keeps the promises and humans think about why the promises were made – a simple division of labour which makes the best use of each's abilities.

A promise theoretic grounding will always generate a well-formed topic map because the model is directly comparable to that of topic maps. The main difficulty with topic maps is finding a sufficient number of meaningful associations to make a dense enough network for easy 'routing' of thought. Here the lack of a rigid hierarchy is essential, and an effective way to find meaningful trains of thought is to weaken associative logic not simply reason about it as a logical system [15].

The approach proposed here discourages the use of traditional data model management schema, i.e. *inter-occurrence relationships* like *hyperlinks* and the *entity relation* models, and especially object hierarchies. Rather than dealing with hundreds or even thousands of tables in the Common Information Model

(CIM), Operational Support Systems or commercial CMDBs, and searching for meaning by brute-force matching of data, we can deal with smaller conceptual neighbourhoods that point more meaningfully from high level intentions to low level implementation. Greater abstraction means fewer things to deal with. Further aspects of this approach will be reported elsewhere.

Acknowledgment. I am grateful to Jan Bergstra, Alva Couch and Steve Pepper for helpful discussions. This work is supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Strassner, J.: Knowledge Engineering Using Ontologies. In: Handbook of Network and System Administration. Elsevier Handbook, Amsterdam (2007)
2. Trastour, D., Bartolini, C., Priest, C.: Semantic web support for the business-to-business e-commerce lifecycle (2002)
3. Serrano, J.M., Serrat, J., Strassner, J., Foghlú, M.O.: Management and context integration based on ontologies behind the interoperability in autonomic communications. In: SIWN International Conference on Complex Open Distributed Systems (CODS 2007), vol. 1, pp. 435–442 (2007)
4. Serrano, J.M., Serrat, J., Meer, S.V.D., Foghlú, M.O.: Ontology-based management for context integration in pervasive services operations. In: Bandara, A.K., Burgess, M. (eds.) AIMS 2007. LNCS, vol. 4543, pp. 35–48. Springer, Heidelberg (2007)
5. Pepper, S.: Topic Maps. In: Encyclopedia of Library and Information Sciences. CRC Press, Boca Raton (2009)
6. Burgess, M.: An approach to understanding policy based on autonomy and voluntary cooperation. In: Schönwälder, J., Serrat, J. (eds.) DSOM 2005. LNCS, vol. 3775, pp. 97–108. Springer, Heidelberg (2005)
7. Bergstra, J., Burgess, M.: A static theory of promises. Technical report, arXiv:0810.3294v1 (2008)
8. Burgess, M.: The Cfengine reference manual, `http://www.cfengine.org/docs`
9. Burgess, M.: Configurable immunity for evolving human-computer systems. Science of Computer Programming 51, 197 (2004)
10. Debusmann, M., Keller, A.: Sla-driven management of distributed systems using the common information model. In: Proceedings of the VIII IFIP/IEEE IM Conference on Network Management, p. 563 (2003)
11. Pepper, S.: The tao of topic maps. In: Proceedings of XML Europe Conference (2000)
12. Burgess, M.: A site configuration engine. Computing systems (MIT Press: Cambridge, MA) 8, 309 (1995)
13. Brenner, M., Garschhammer, M., Sailer, M., Schaaf, T.: CMDB - yet another MIB? On reusing management model concepts in ITIL configuration management. In: State, R., van der Meer, S., O'Sullivan, D., Pfeifer, T. (eds.) DSOM 2006. LNCS, vol. 4269, pp. 269–280. Springer, Heidelberg (2006)
14. `http://research.iu.hio.no/topicmaps/tm.php`
15. Couch, A., Burgess, M.: Compass and direction in topic maps (Oslo University College preprint) (2009)

A Policy-Driven Network Management System for the Dynamic Configuration of Military Networks

Wei Koong Chai[1], Kin-Hon Ho[2], Marinos Charalambides[1], and George Pavlou[1]

[1] Networks and Services Research Lab, Department of Electronic and Electrical Engineering, University College London, Torrington Place, London, WC1E 7JE, UK
[2] Department of Computer Science, City University of Hong Kong, Kowloon Tong, Kowloon, Hong Kong
{w.chai,m.charalambides,g.pavlou}@ee.ucl.ac.uk, kinhonho@cityu.edu.hk

Abstract. Military networks constantly evolve to accommodate state-of-the-art technological developments across both military and commercial systems. Operating and maintaining such complex networks is no longer a trivial task. This paper presents a policy-based network management system for military networks which allows non-technical personnel (e.g. the military commander) to guide the network to behave towards specific objectives. Through policies, the system can optimize an IP-based multi-class military network with different or combinations of objectives, as requested by the decision makers. We show how the system can dynamically produce the required network configurations given specific requirements and illustrate its practicality using case studies.

Keywords: Policy-driven management, traffic engineering, military networks.

1 Introduction

In this information age, communication and information services have become vital components in security and defense agencies. Military networks are continually being enhanced with latest communication technologies to improve their flexibility, efficiency, resilience and security. Furthermore, it is highly desirable to provide service differentiation and quality of service (QoS) to ensure timely and safe delivery of mission critical information while maintaining acceptable performance to other less critical traffic. For example, during a battle, some parts of the military network may be vulnerable to attacks, reducing thus the usable available bandwidth. In this case, the network has to be re-engineered in order to sustain acceptable QoS for high-priority traffic. Failing to deliver the committed QoS may adversely affect the communications between military sites. These sophistications have made the management of the network a task requiring specific networking knowledge which decision makers (e.g. military commanders) may not possess. Although they may not have deep understanding on networking techniques, their non-technical requirements can be mapped to network-level policies, which when enforced, can achieve the high-level military objectives.

An effective way to optimize the usage of network resources is to control traffic routing and subsequently support QoS. Traffic engineering (TE) is the process of specifying the manner on how traffic within a given network should be routed in order to optimize its performance [1] by balancing the load distribution or minimizing the bandwidth consumption in the network. In the context of military networks, a commander makes decisions following complex thought processes that take into account the criticality of the mission as well as current and predicted operation situations. The policies to be applied will capture the intent of the commander in terms of how the network management system should behave. Network configuration parameters are derived based on these policies and applied to the network in response to the commander's decisions.

In this paper, we concentrate on IP-based networks whereby each network link is assigned a link weight and traffic flows are routed along shortest paths to destinations. The shortest path is defined as the route between two nodes with the least total sum of link weights. Traffic routing can be controlled by setting appropriate link weights in the network by taking into account the overall traffic demand, so as to satisfy TE objectives such as improving load balancing while achieving acceptable QoS. The network is also assumed to be supporting Differentiated Services (*DiffServ*) [2] whereby several traffic classes may exist and each requires different treatment. Within this framework, critical information can be prioritized and treated differently.

In general, it is hard to find an efficient algorithm for achieving an optimal solution for the link weight setting problem. We therefore design and implement a heuristic algorithm based on the Tabu search method for solving the problem. The algorithm virtualizes the network into separate network planes so that traffic of different priorities can be forwarded with different QoS and produces a set of link weights for each virtual network plane.

In this paper, we propose a policy-based network management system that applies policy-driven TE techniques to optimize the military network such that top-level military objectives are achieved. This work has been part of a collaborative project for military networks. The paper is organized as follows: Section 2 presents the overall architecture of the proposed policy-based network management system and the inter-relationships among the components of the system. Section 3 details the implementation of the system including the specification of the link weight optimization algorithm. We evaluate our system in section 4. Case studies are presented to illustrate how our system can be used in realistic military scenarios. We provide the related work in section 5. Finally, we summarize our contributions and conclude the paper in section 6.

2 Decomposition of the Network Management System

2.1 Overall System Architecture

In this section, we present an overview of our proposed policy-driven network management system. We illustrate in Fig. 1 a decomposition of the system which is composed of two subsystems: *Policy-based Management (PBM)* and *Network Dimensioning (ND).*

Network-level policies achieving military objectives are entered in the PBM subsystem, stored in a repository and subsequently enforced, influencing the functionality of the ND subsystem in order to address the continuously changing high-level objectives of the decision makers. Based on the requirements specified in the policies, the ND subsystem then optimizes the network by executing the optimization algorithm (detailed in section 3.1).

A Java-based Graphical User Interface (GUI) is also implemented to provide a clear and controllable representation of the outcome produced by the ND subsystem. Users can view how each traffic class is being routed across any source/destination pair and the load of each link before and after the optimization. Statistical information given in plots can be instantly generated to facilitate users in understanding the effect of the optimization. The GUI also allows users to manipulate the network topology (e.g. zooming and re-arranging the nodes) for a clearer view of the targeted network.

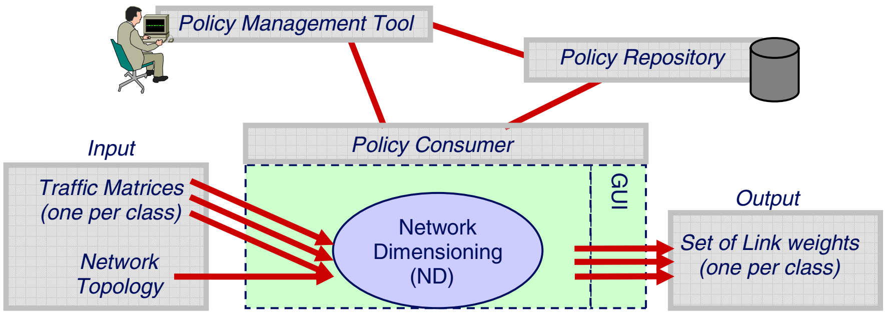

Policy Management Tool
Policy Repository
Policy Consumer
Input
Traffic Matrices (one per class)
Network Topology
Network Dimensioning (ND)
GUI
Output
Set of Link weights (one per class)

Fig. 1. Overview of the policy-driven network management system

2.2 Design of Traffic Engineering Components

The ND subsystem is developed with both existing and future TE constraints in mind. The core of the ND subsystem is the link weight optimization algorithm which incorporates the multi-topology (MT) concept for handling multi-traffic class scenario. It provides a means to configure class-based routing for different types of traffic. The physical military network is virtualized as separate network planes (or virtual topologies). Each traffic class uses a specific routing table for that virtual topology. In our algorithm, MT is used to isolate the routing of each *DiffServ* Per-Hop Behavior (PHB) by providing different link weight settings for individual PHBs. Hence, packets of different PHBs can be routed independently from one another. This advanced feature is supported by configuring multi-topology protocols such as M-OSPF [3]. Coupled with the link weight optimization algorithm, an optimal solution can thus be computed for the multi-QoS class scenario.

Fig. 2 illustrates the concept of MT-TE and serves as an example of how MT provides intra-domain path diversity across three virtual topologies between a single source/destination pair. With default link weight as 1 for all links, all traffic flows are routed via the middle path of the topology, causing congestion. However, using MT-TE, our ND subsystem can compute a set of dedicated link weights for each traffic class. Based on individual link weight settings, traffic flows of each class may follow a different path. Congestion in the middle path is thus alleviated.

The behavior of the ND algorithm can be directly influenced by the policy directives generated by the PBM subsystem. For instance, the optimization algorithm may use a different optimization objective for achieving the desired performance represented by different policies. Besides the input from the PBM subsystem, the algorithm also requires two other inputs: the network topology (including link capacities) and the expected traffic demand (in the form of estimated traffic matrices for each traffic class).

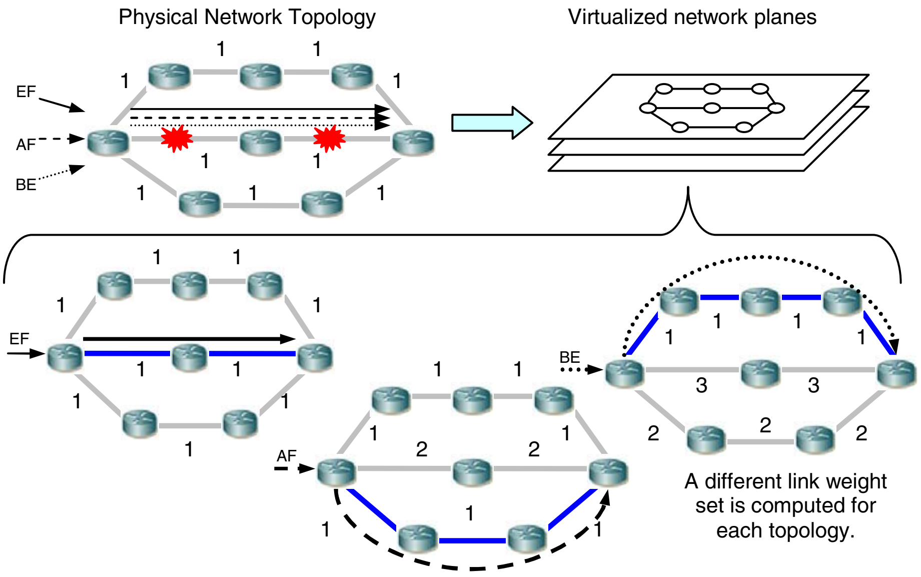

Physical Network Topology
Virtualized network planes
EF
AF
BE
A different link weight set is computed for each topology.

Fig. 2. MT-TE: The physical network virtualized as separate logical topologies with each having a different link weight setting. Congestion at the shortest path (middle route) is avoided.

2.3 Design of Policy Components

Policy-based management [4] provides the ability to (re-)configure networks so that desired QoS goals are achieved. The approach facilitates flexibility and adaptability as policies can be dynamically changed without modifying the underlying implementation. This is particularly useful in military scenarios where the high-level objectives of a commander can be encoded into policies and used to derive new configurations on demand. The key components of Fig. 1 are briefly described below.

- *Policy Management Tool (PMT)*: The PMT [5] provides the policy creation environment through which a network administrator can enter new policies. The latter are of the form *if <condition> then <action>*, where the conditional part can be a compound expression encapsulating network state and events. The *<action>* expression can be a set of actions that specify the way in which the optimization algorithm should run to achieve the high-level objectives. The tool allows the user to store newly created policies, or view policies already stored in the repository.

- *Policy Repository (PR)*: The PR is a centralized component based on an Lightweight Directory Access Protocol (LDAP) implementation that stores policies after they have been translated into object-oriented representation. Once a new policy is stored, activation information is passed to the Policy Consumer in order to retrieve and enforce it when the relevant conditions are met.
- *Policy Consumer (PC)*: The PC [6] is the most critical component of the policy architecture and is responsible for enforcing policies on the fly while the network is operating. We integrate the functionality of both the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP) of the IETF policy framework [7] into the PC. Before enforcing a policy, the consumer communicates with the PR and downloads the relevant policy objects. These are subsequently used to generate a script that implements the policy, which is interpreted into management operations when the policy is enforced. The latter can be achieved either statically through the PMT, or dynamically based on network events, and in both occasions management operations involve setting optimization attributes of the ND algorithm.

3 Policy-Driven IP Traffic Engineering

3.1 IP Traffic Engineering

3.1.1 Problem Formulation

We formally define the IP traffic engineering (i.e. link weight optimization) problem. A network is modeled as a directed graph $G = (V, A)$ where V and A represent the set of nodes and links respectively. Each link $a \in A$ has a capacity denoted by $c(a)$. We have a traffic matrix D that for each pair $(s,t) \in V \times V$ represents the demand $D(s,t)$ in traffic flow between source node s and destination node t. With each pair of (s,t) and each link a, we associate a variable $f_a^{(s,t)}$ telling how much of the traffic flow from s to t goes over a. Variable $l(a)$ represents the total load on link a, i.e. the sum of the flows going over a. Furthermore, we denote the utilization of link a by $u(a) = l(a)/c(a)$.

With the above notation, the IP TE problem can be formulated as below:

$$Minimize \quad \Phi \tag{1}$$

subject to

$$\sum_{x:(x,y)\in A} f_{(x,y)}^{(s,t)} - \sum_{x:(y,z)\in A} f_{(y,z)}^{(s,t)} = \begin{cases} -D(s,t) & \text{if } y = s, \\ D(s,t) & \text{if } y = t, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$f_a^{(s,t)} \geq 0 \qquad \text{a} \in \text{A; s,t} \in \text{V} \tag{3}$$

Constraints (2) are flow conservation constraints that ensure the desired traffic flow is routed from s to t.

In our policy-based network management system, we implemented three different ND optimization objectives for policies to be mapped to.

Optimization Objective 1 (OBJ-1): Minimize the maximum link utilization:

$$\Phi = \underset{\forall a \in A}{\mathrm{Max}} \; \mathrm{u(a)} \tag{4}$$

Maximum link utilization is defined as the highest utilization among all the links in the network. Minimizing this value ensures that traffic is moved away from congested to less utilized links and is balanced over the links.

Optimization Objective 2 (OBJ-2): Minimize the average link utilization:

$$\Phi = \frac{\sum_{\forall a \in A} u(a)}{|A|} \tag{5}$$

The goal of this optimization objective is to minimize the overall network link utilization. It tries to minimize the total bandwidth consumption in the network by shortening the routes to be used for traffic delivery.

Optimization Objective 3 (OBJ-3): Minimize the weighted link utilization:

$$\Phi = w \cdot \underset{\forall a \in A}{\mathrm{Max}} \; \mathrm{u(a)} \; + (1-w)\frac{\sum_{\forall a \in A} u(a)}{|A|} \tag{6}$$

This optimization objective minimizes the weighted sum of the maximum and average link utilization. It allows policy users to specify the value of w in order to adjust the importance and balance of the two optimization objectives.

3.1.2 A Heuristic Multi-QoS Class Link Weight Optimization Algorithm

We propose a Neighborhood Search Algorithm (NSA) based on the Tabu Search technique for solving the problem efficiently with respect to performance and computation complexity. The NSA is an important tool to solve hard combinatorial optimization problems efficiently. The basic steps of NSA can be summarized as follows. Consider a starting solution x. NSA explores the solution space by identifying the neighborhood of x, $N(x)$. The neighbors of x are solutions that can be obtained by applying a single local transformation (or a move) on x. The best solution in the neighborhood is selected as the new current solution. This neighborhood searching iterates until the stopping criterion is satisfied. The algorithm returns the best visited solution.

During neighborhood search, NSA can move the current solution to the best neighbor that either improves or worsens the quality of the solution. To avoid cycling, a special memory list is used to store previously visited solutions for a certain number of iterations. A neighbor solution is rejected if it is already in the list. To increase effectiveness, an intensification or diversification technique is used to force the algorithm to explore parts of the solution space that have not been searched yet.

Our NSA incorporates following aspects:

1) Neighborhood search: the best move in the neighborhood is identified as follows:

- ***Step 1.*** Identify two sets of links – those whose utilizations are within a small percentage of the maximum link utilization (i.e. heavily utilized links) and those whose utilizations are within a small percentage of the minimum link utilization (i.e. lightly utilized links). Consider the most utilized link in the first set.
- ***Step 2.*** Increase the weight of the link by a random value in an attempt to move traffic away from that link and reduce its load. Randomly select a link from the lightly utilized link set and decrease its weight by a random value in an attempt to attract more traffic over this link from the highly utilized links.
- ***Step 3.*** Run Dijkstra's shortest path first algorithm for the current link weights to re-calculate the routes for the traffic. Then re-calculate the objective.
- ***Step 4***. Select the next most utilized link and repeat steps 2 to 5 until all the links in the heavily utilized link set have been considered.
- ***Step 5***. Among all feasible solutions, choose the one with the minimum maximum link utilization and consider it as the current solution.

2) Tabu list: The tabu list memorizes the most recent moves, operating as a first-in-first-out queue. Its size depends on the size and characteristics of the problem. In our problem, the tabu list consists of the links whose weights have been recently changed and the amount of increase/decrease applied to the corresponding link weight.

3) Diversification: The goal of diversification is to prevent the searching procedure from indefinitely exploring a region of the solution space that consists of only poor quality solutions. It is applied when there is no obvious performance improvement after a certain number of iterations. In other words, we ensure that the algorithm is not restricted to a local optimum but will explore further to find a global optimum solution. For a diversification, several links are picked up from each of the lightly and heavily utilized link sets. The weights of the selected links from the former set are decreased while the weights of the selected links from the latter set are increased. Note that any solution produced by the diversification is acceptable if it is feasible.

4) Stopping Criterion: the search procedure stops if either the pre-defined maximum number of iterations is reached or there is no pre-defined performance improvement for the objective function after a certain number of consecutive diversifications.

3.1.3 Optimization for Multiple Traffic Classes

The ND subsystem allows the optimization of multiple traffic classes. We implemented this feature in the following way. First, the traffic classes are prioritized according to their importance. In *DiffServ*, the priority of traffic classes follows the order of *EF*, *AF* and *BE*. Next, the ND algorithm is executed for the highest-priority traffic class. The outcome (i.e. the set of link weights) is assigned to that traffic class over the corresponding virtual network plane. The residual bandwidth in the network, which has not been used by any traffic class, is recorded. The ND algorithm repeats the procedures above for optimizing the next highest-priority traffic class while taking into account the residual bandwidth. It stops when all the traffic classes have been processed.

3.2 Policies for Optimizing Military Networks

To effectively utilize the ND algorithm and adapt the network configuration according to some objectives, we have defined a set of military policies that can influence the output of the algorithm and demonstrate its capabilities. These policies are derived from sample high-level military objectives that can be requested by a commander as follows:

Commander Objective 1 (CO-1):
Under normal conditions setup network to either:

a. avoid service quality degradation
b. accommodate as many services as possible

Commander Objective 2 (CO-2):
During rescue missions sustain quality of associated services

Commander Objective 3 (CO-3):
During battle readiness level accommodate as many services as possible without compromising quality.

Each of the above objectives is mapped to the policies of Table 1 which when triggered, instruct the ND subsystem to execute the algorithm with the necessary parameters and produce a new network configuration. It should be noted that we distinguish between off-line (e.g. P1) and on-line policies (e.g. P2, P3). The former are evaluated statically before the network is deployed, whereas the latter are evaluated dynamically while the network is operating.

Table 1. Network optimization policies

Policy ID	Policy Condition	Policy Action
P1	event(initNetwork)	optimize(*objective*)
P2	event(congestion) && event(rescueOp)	optimize(maxLinkLoad)
P3	event(battleReadiness) && maxLinkLoad < 90%	decrWeight(20%)

Policy P1 is used to generate a link weight setting for the initial deployment of the network. It is an off-line policy that executes the algorithm according to the ND optimization objective, which is specified as a parameter in the policy action. Minimizing the maximum link load spreads the traffic throughout the network thus achieving CO-1a, whereas minimizing the average link load forces traffic to follow the shortest path thus achieving CO-1b.

Policy P2 is a special case of P1 but is instead triggered by the run-time event of network congestion during a rescue operation. Since priority in such missions should be given to supporting services (e.g. video feed from the rescue location), the policy will result to a congestion-resolving configuration, despite the fact that other lower-priority services may suffer longer delays.

The last policy (P3) is used to achieve CO-3 and, instead of optimizing individual ND objectives, it takes the balanced approach described in the previous section, where a weight w acts as the bias between the two objectives. This policy is triggered when the network is required to switch to battle readiness mode. The additional condition concerning the maximum link load forces the dimensioning algorithm to iteratively decrease w (initially set to 1) by 20% until the resulting configuration does not overload any network link. The results of enforcing these policies are described in the section 4.3, demonstrating that their dynamic nature can provide the network with self-optimization capabilities.

4 System Evaluation and Analysis

To evaluate the capability of the proposed policy-based network management system, we carry out a systematic evaluation study of two scenarios. In the first scenario, we evaluate the effectiveness of our ND algorithm on synthetic network topologies and traffic matrices. This aims to ensure that the algorithm performs as expected. After that, we provide three case studies to illustrate how our policy-based network management system can be applied to practical military situations.

4.1 Evaluation Study Setup

As mentioned, there are two inputs to the ND algorithm: the network topology and the traffic matrix. In our evaluation study on the synthetic network, we generate the traffic matrices based on the gravity model [8]. We manipulate the level of traffic demand via the traffic intensity multiplier *k* in the model. The larger the value of *k*, the higher the traffic load to the network. The network topology is generated based on the exponential random graph model [9]. The tunable variables in this model are the network connectivity multiplier, α and the maximum distance between any pair of network nodes, *L*. In this model, the probability of a link connecting a pair of nodes increases linearly with α and decreases exponentially with the distance between them.

4.2 Evaluation on a Synthetic Network

We first present our evaluation results for the ND algorithm in finding the optimal link weight set for a network topology with a range of traffic load conditions so that the traffic flows are routed to avoid overloading of network links. A 30-node network is generated with $\alpha = 0.5$ and $L = 50$ unit distance. All links have the same capacity of 8 Mbps. In this section, we configure the network to support only one PHB. The default weights for all links are set to 100. We varied the traffic load by tuning the variable *k* in the Gravity model. We optimized the network based on *OBJ-1* for this evaluation (i.e. minimizing the maximum link utilization).

We observe from Fig. 3 a general increasing trend of the highest link load when *k* is increased. This is because the network load is directly proportional to *k*. In all cases, our algorithm manages to find a set of link weight that reduces the highest link load. When $k > 25$, the highest link load before optimization is over 100%, implying severe network congestion. For cases where $30 \leq k \leq 40$, the ND subsystem manages to find a link weight setting that avoid link overloading in the network (i.e. reduce highest link load to below 100%). However, it fails to find a feasible solution that avoids link overloading when $k > 40$. This is caused by the existence of a critical links which traffic flows to some destinations cannot avoid. In such cases, simply optimizing traffic routing may not solve the congestion problem. Additional physical resources (e.g. new links or increased capacity to the critical link) are necessary in order to accommodate the traffic load.

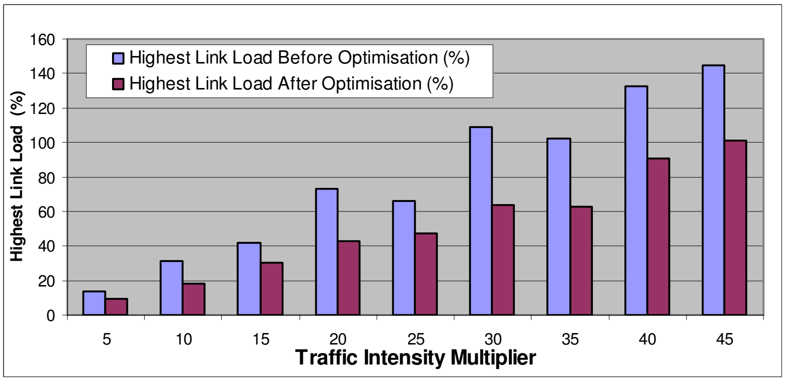


Fig. 3. Highest link load for different traffic load is always reduced after optimization

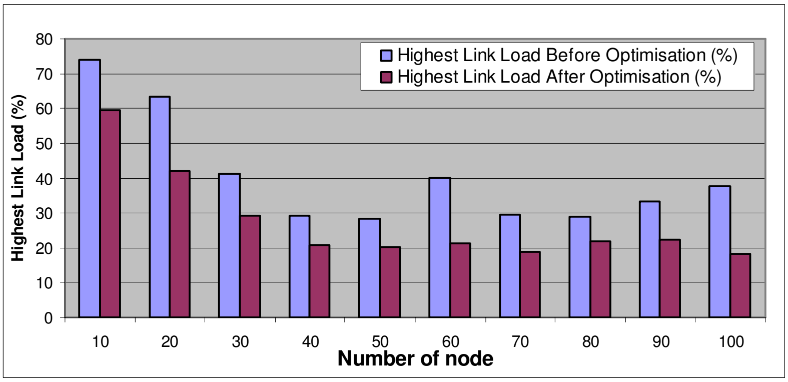


Fig. 4. Highest link load for different network sizes is always reduced after optimization

Next, we evaluate the effects of different network sizes with $\alpha = 1.0$, $L = 100$, link capacity = 8 Mbps and $k = 30$. Fig. 4 shows the optimization results. In all cases, the highest link load is decreased after the optimization.

4.3 Case Study

In section 4.2, we have shown the effectiveness of the ND subsystem in optimizing network resource utilization. Now, we illustrate how the proposed policy-driven network management system can be applied to practical military networks. For the case studies presented here, we take a sample military network topology of 14 nodes and estimated traffic matrices. For confidentiality reasons, we appropriately scale the traffic matrices and do not reveal the topology details in the paper.

4.3.1 Case Study 1: Initialization of the Military Network

A military network with *DiffServ* capabilities has just been deployed and a daily communication load is forecasted for each PHB. The network must be dimensioned to spread traffic across the network in a balanced manner to ensure acceptable service to all PHBs (i.e. CO-1a). We achieve this by applying policy P1 that instructs the ND subsystem to optimize the traffic routing based on *optimization objective 1.*

if event(initNetwork) ***then*** *optimize(OBJ-1)*

We show the effect of applying this policy to the network in Fig. 5. We observe that the highest link load is reduced from 151.25% to 78.75%. Before the optimization, there are seven links that are being utilized and link 3-4 and 6-10 are overloaded. After optimization, more links are being used to carry the traffic flows (i.e. 12 links) and all the link loads are well below 100% with the highest reaching approximately 80% (i.e. link 8-9). This implies that some traffic flows are being routed via longer paths traversing some links which are otherwise unused so that no link(s) are being overloaded. In other words, the new link weight setting manages to spread the load in a more evenly manner. The downside of this is that the overall link load is increased from 22.765% to 24.948%. Nevertheless, this increment is negligible.

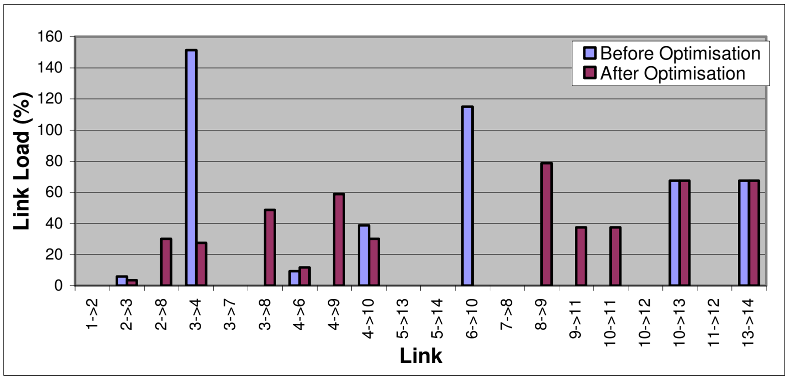


Fig. 5. The link load before and after optimization

4.3.2 Case Study 2: Rescue Mission

A military helicopter carrying an important person crashes at a politically sensitive site. The exact coordinates of the site are unknown. A rescue mission to recover the personnel is initiated. An unmanned surveillance vehicle that provides a video feed back to the headquarters is deployed to determine the crash site. Meanwhile, a link within the network fails causing the quality of the video to deteriorate significantly because of congestion. The quality of the video feed must be immediately restored. Policy P2 is applied to the network with the new information of the failed link.

if event(rescueOp) && event(congestion) ***then*** *optimize(OBJ-1)*

We show in Fig. 6 screenshots from our system. Note that the failed link (i.e. link 4-10) has already been removed from the network topology. Before applying policy

P2, we can clearly see that link 3-4 and 6-10 are overloaded (i.e. traffic flowing through these links suffers QoS degradation). After the optimization, these links are no longer overloaded while several links carry increased traffic (e.g. link 3-8 and 10-11). Some traffic flows have been diverted to other routes to alleviate congestion.

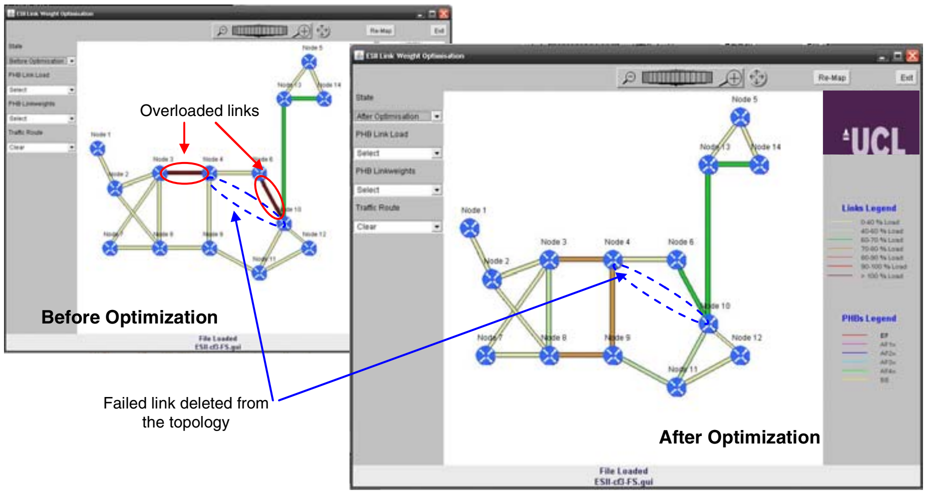


Fig. 6. Re-optimizing the network to sustain QoS after link failure

4.3.3 Case Study 3: Dynamic Configuration of the Military Network

New intelligence gathered indicates possible attacks. The network is to be set to battle readiness mode (i.e. preparing to accommodate new missions and thus having increased network load). This requires the network to have minimal average load without causing QoS degradation to in-progress services. With the weight w of *optimization objective 3* initially set to 1.0, policy P3 is applied to the network as follows:

***if** event(battleReadiness) **&&** maxLinkLoad < 90% **then** decrWeight(20%)*

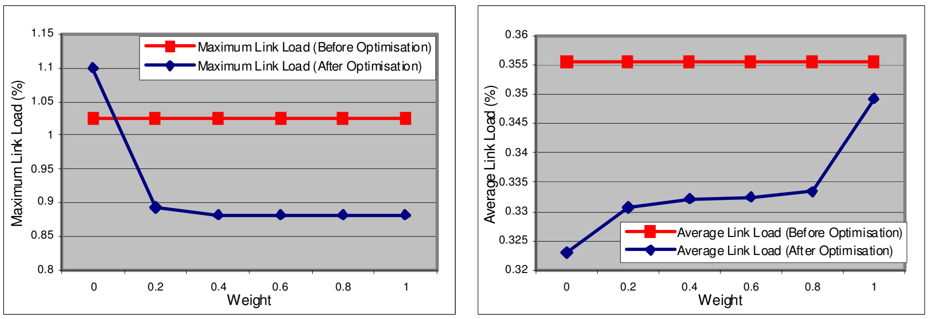


Fig. 7. Optimization with different weights; (left) maximum link load, (right) average link load

We present the results of the optimization via *OBJ-3* by varying the weight from 0.0 to 1.0 in Fig. 7. The plots show how the weight can tune the optimization results from one objective to another. When the weight is 0.0, we are essentially optimizing the network via *OBJ-2*. Conversely, when the weight to 1.0, we are actually using *OBJ-1* as the ND optimization objective.

In this case study, the objective is to prepare the network for new incoming mission traffic. At the same time, in-progress services are not to be disrupted by congestion. Hence, the policy runs the ND algorithm iteratively with decreasing weight until the maximum link utilization exceeds 90%. From our results, we can see that the optimal weight setting is 0.2 where the average network is decreased without having any link higher than 90% utilization.

5 Related Work

Military networks are by nature highly complex and dynamic. Compared to commercial networks, the response times for military networks are often much shorter and critical. Such networks also require high level of automation. As such, policies, especially dynamic ones, are much sought-after for efficient enforcement of network configurations. In fact, [10] has illustrated the potential of a policy-based management system for military networks. Previous work on policy-based network management system (e.g. [5] and [11]) mostly targeted regular civilian or commercial networks, disregarding the specific characteristics of military networks. We built upon past work and developed a full policy-based management system focusing on the requirements of military networks.

Routing management system is being used by network and service providers to effectively manage routing in their networks. The objective is to optimize the operational IP performance such as minimizing bandwidth consumption or improving load balancing. There are several such systems in literature. [12] has proposed a system that optimizes routing in pure IP networks by appropriately adjusting the link weights of the OSPF protocol. [13] proposed a similar system for IP/MPLS networks. Instead of optimizing link weights, explicit paths between each pair of nodes are determined. However, these systems required the participation of network operators to manually configure their optimization behaviours and have not taken into account system autonomy and self-management which are two features much needed in military networks.

6 Conclusions

In this paper, we presented the results of our work from a collaborative research project regarding military networks. A policy-based network management system with user-friendly graphical interface targeting military requirements and scenarios was described. The system is driven by policies to optimize the network for different objectives. It is also capable of dealing with conflicting optimization objectives via a weight acting as a tuning parameter. We also developed and validated a link weight optimization algorithm employing the Tabu Search method to influence the dynamics of routing traffic within the network. We incorporated the concept of multiple virtual topologies in our algorithm for handling multi-QoS class scenarios. Our results suggest that the pure IP-based *DiffServ* network can be optimized for various TE objectives through the intelligent assignment of link weights. With the exception of heavily loaded conditions, there will generally be multiple feasible link weight solutions. We should also mention that this optimization and subsequent configuration is fairly robust as the traffic matrices in those networks can be fairly accurately predicted, unlike in operational ISP networks in which end-user behaviour can vary dramatically. Finally, we demonstrated our system via three realistic military scenarios applying policies to achieve specific high level objectives. We illustrated the application of both static and dynamic policies to the military network.

References

1. Awduche, D., et al.: Overview and Principles of Internet Traffic Engineering, IETF RFC 3272 (May 2002)
2. Blake, S., et al.: An architecture for Differentiated Service, RFC 2475 (December 1998)
3. Psenak, P., et al.: Multi-Topology (MT) Routing in OSPF, IETF RFC 4915 (June 2007)
4. Verma, D.C.: Policy-Based Networking, pp. 155–165. New Riders Publishing (2000) ISBN 1-57870-226-7
5. Flegkas, P., et al.: Design and Implementation of a Policy-based Resource Management Architecture. In: Proc. IEEE/IFIP Integrated Management Symposium (March 2003)
6. Flegkas, P., Trimintzios, P., Pavlou, G.: A Policy-based Quality of Service Management Architecture for IP DiffServ Networks. IEEE Network Magazine Special Issue on Policy Based Networking 16(2), 50–56 (2002)
7. Yavatkar, R., Pendarakis, D., Guerin, R.: A Framework for Policy Based Admission Control, Informational RFC 2753 (January 2000)
8. Zhang, Y., et al.: Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads. In: Proc. ACM SIGMETRICS (2003)
9. Zegura, E., Calvert, K., Bhattacharjee, S.: How to Model an Internetwork. In: Proc. IEEE INFOCOM (April 1996)
10. Chadha, R.: Beyond the Hype: Policies for Military Network Operations. In: Proc. Int'l. Conf. on Systems and Network Communications (ICSNC) (October 2006)
11. Trimintzios, P., et al.: Service-driven Traffic Engineering for Intradomain Quality of Service Management. IEEE Networks Magazine, 29–36 (May/June 2003)
12. Fortz, B., Rexford, J., Thorup, M.: Traffic Engineering with Traditional IP Routing Protocols. IEEE Communications Magazine 40(10), 118–124 (2002)
13. Xiao, X., Hannan, A., Bailey, B., Ni, L.M.: Traffic Engineering with MPLS in the Internet. IEEE Network Magazine 14(2), 28–33 (2000)

A Strategy for Multi-Agent Based Wireless Sensor Network Optimization

Ahmad Sardouk, Rana Rahim-Amoud, Leïla Merghem-Boulahia, and Dominique Gaïti

ICD/ERA, FRE CNRS 2848, Université de Technologie de Troyes,
12 rue Marie Curie, 10010 Troyes Cedex, France
{ahmad.sardouk,rana.amoud,leila.boulahia,dominique.gaiti}@utt.fr

Abstract. The multi-agent approach has been proposed in the literature as a solution for data gathering, and routing in Wireless Sensor Networks (WSNs). In these propositions, the knowledge of an agent is generally limited to a single parameter such as the energy of the sensor node and/or to the address of its next hop in a routing protocol proposition. In this paper, we propose a strategy for the agent to make a more appropriate decision to cooperate or not in a data gathering session. This strategy uses, in addition to the energy of the node, several parameters from the local view of the agent as the position of the node within the network, the network density, and the information importance degree. Through successive simulations, this strategy has proved its ability to manage cleverly the power consumption of the sensor nodes and hence to extend the WSN life time.

Keywords: Wireless sensor networks, Multi-agent systems, Energy-efficiency, Autonomic sensor network.

1 Introduction

The rapid technological progress in wireless sensor networks is attracting more and more applications. This domain was known by its importance for military applications where sensors are scattered behind the enemy lines to collect information. However, nowadays the WSN is proposed for medical use to monitor the patients with special diseases and prevent them from spreading. Many other applications, such as tracking and large scale environmental monitoring, require this improvement in WSN.

However, the sensor nodes have limited power defined by their batteries. The radio entity of the sensor node drains this battery in transmission, reception and even in idle state. Hardware advances in communication and battery technology will lead in overcoming some of the power problems. Thus, the constructors of sensor nodes are giving a special interest to the development of processing and memory capacity of sensor nodes. An example of this development could be seen in SunSpot [10] sensor nodes. They offer new sensor nodes with 180MHz of processing, 512KB of RAM and 4MB of memory with an implemented java virtual machine. However, problems will not be completely resolved and software improvements are still needed. Indeed, a sensor node should be able to manage its battery, to judge the importance of the gathered information, to reduce the communication of non useful or non important information, etc. In other terms, a sensor node should be autonomous and should be able to make decisions while taking into consideration its state and the state of its environment.

In this context, the multi-agent systems (MASs) emerged as an important tool to build autonomic networks. An MAS is composed of a set of agents able to cooperate with each other to exchange information and execute tasks in order to achieve a global objective.

In one of our previous papers [8], we have proposed an energy-efficient communication architecture based on the multi-agent approach. In the proposed architecture, only an agent having important information cooperates with its neighbours in order to create a data gathering session. This session summarizes the data of multiple sensor nodes into one message allowing to significantly saving energy.

In order to manage more cleverly the power consumption of the nodes, in this paper, we enhance the architecture defined in [8] by proposing an agent strategy which is a step towards the autonomic WSN. By using this strategy, the agent, implemented in a sensor node, will take into consideration several parameters before deciding to cooperate or not in a data gathering session. These parameters are obtained from the agent environment or by exchanging information with its neighbours. In this paper we define the following four parameters: (1) The energy of the node, (2) the position of the node within the network, (3) the network density in the coverage zone of the node, and (4) the importance degree of the gathered information. These parameters will be explained and discussed in details in section 4.

The rest of this paper is organized as follows. In section 2, we present briefly the multi-agent systems. In section 3, we review the agent-based solutions proposed in the literature. Next, section 4 contains the main contribution of this paper: the agent strategy. Simulations setup and results analysis are discussed in section 5 and 6 respectively. Finally, section 7 concludes the paper and introduces the future work.

2 Multi-Agent Systems

According to [2], an agent is a physical (robot) or virtual (real time embedded software) entity having trends and resources, able to perceive its environment, to act on it and to acquire a partial representation of it (called the local view of the agent). An agent is also able to communicate with other peers and devices and has a behaviour that fits its objectives according to its knowledge and capabilities. Furthermore, an agent can learn, plans future tasks and is able to react and to change its behaviour according to the changes in its environment.

An MAS is a group of agents able to interact and to cooperate in order to reach a specific objective. Agents are characterized by their properties that determine their capabilities. Different properties are defined like autonomy, proactive-ness, flexibility, adaptability, ability to collaborate and to coordinate tasks, mobility, etc.

According to its role within its environment, the agent acquires some of these properties. Multi-agent approach is well suited to control distributed systems. WSN are good examples of such distributed systems. This explains partly the considerable contribution of agent technology when introduced in this area.

3 Agent-Based Propositions in WSN

Agents and MAS are two interesting concepts for a great number of researchers in different domains. In WSN, researchers have been interested in these two concepts in order to propose energy-efficient approaches especially in routing and data gathering problems. In this section, we present some of the proposed solutions and we underline the local view of the agent in each proposition. In fact, the decisions of an agent are based on its local view. This view presents the information of its current sensor node information as the remaining power in the node, the list of its neighbours, etc.

Since WSN is limited in energy, designing an energy-efficient routing protocol appears as a key point to extend the network life time. Therefore, authors of [5] propose an intelligent agent routing scheme over WSN. This scheme is based on building an abstract tree structure where the sink is the root. When an event happens, the information can be passed back to the sink through the tree structure. The authors propose to introduce an agent at each level of the tree. This agent transfers the information of a lower level to an upper level. At each level of the tree, they select the closet (in term of distance) sensor node to the upper level to implement the agent in this proposition, the local view of the agent is limited to the address of its upper level node and the distance to it (deduced from the received power).

Authors of [3] propose another approach based on the use of mobile agent. They propose to carry each data packet sent from a source node by an autonomous mobile agent. This agent is responsible for making the appropriate route decisions, in an energy-efficient manner, based on its local view. The authors define a forwarding table in each sensor node. This table contains the list of possible next hops with their remained power and estimated power (or cost). The estimated power is the transmission power needed to reach the next hop. In this proposition, the energy of the next hop appears as the main information in the local view of the agent.

In mobile agent propositions [1] [4], authors propose to send the processing code to the sink. This code is a part of a message called mobile agent, which contains also the list of source nodes. The mobile agent passes through each of the source nodes defined in the list, processes their data locally and concatenates them into the data field of the message. This technique provides a gain in power by (1) eliminating the local redundant information as the mobile agent processes data locally, and (2) concatenating multiple nodes information into one message, which means one message overhead for multiple source nodes information. The authors suppose a fixed list of source nodes in each mobile agent. The agent, thus, will just focus on finding its route to process and concatenate the data from its list of source nodes. Hence, the local view of the agent, when it arrives to a node, will be restricted to the local raw data and the address of the next sensor node (next source node in the list).

In our previous work [7] [8], we have proposed an information importance based communication architecture (IBC). Thus, the data gathering session starts when an agent (sensor node) detects important information. This agent invites its one hop neighbours to cooperate in order to gather the maximum possible of information and to create a cooperation message summarizing these collected information. However, the neighbour agent, who is at the same time the first hop on the path to the sink for the agent in question (source node), will not response the cooperation request. Indeed, once the cooperation message is ready, this neighbour agent (called intermediate agent) will receive the message and will invite its one hop neighbours' agents to co-operate. The intermediate agent will gather the information of its one-hop neighbours and extend the initial cooperating message. This message will be then sent to the next intermediate agent. The new intermediate agent will, in its turn, repeat the same scenario. This scenario will be repeated until reaching the sink node.

The example in Fig. 1 illustrates the IBC approach where we can see that there is a static agent (smile face) implemented in each node. We suppose that agent A has important information, hence it cooperates with its one hop agent neighbours (E, I and B) to gather their information. Then, it sends the resulting cooperation message to its first hop on the path to the sink, which is agent B. B, C, and D are the intermediate agents, each one of them cooperates with its one hop neighbours and concatenates data into the main cooperation message. We would like to underline here that the neighbour agents decide to cooperate or not following the importance of their gathered information. In this proposition, the local view of an agent was limited to the importance of the gathered information.

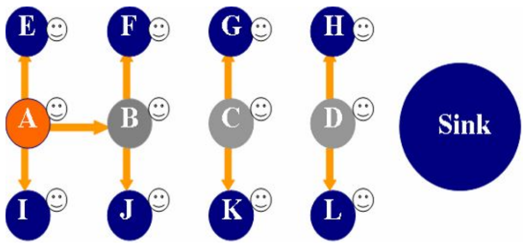


Fig. 1. Agents based communication example

The authors of [6] propose the use of agents for a power management in wireless sensor network. Unfortunately this work does not give results; however, it studies the possibility and importance of using the agents in WSN. The authors examine some of the potential decisions an agent can take regarding intelligent power management. They give the example of a node, in the middle of the network, which routes further nodes messages if it has a sufficient battery level. This work emphasizes the importance of using different characteristics of agents in WSN.

The approaches presented in this section propose the use of MASs in WSN in order to decrease the power consumption. However, in each proposition, only a few number of parameters of the agent local view have been considered by the agents in order to make decisions. In the majority of these propositions, the authors have mainly focused on the available power. However, some other parameters would be important for the longevity of a WSN such as the position of the node, the network density, etc.

In the next section, we detail our agent strategy and the parameters that it takes into account in order to make appropriate decisions.

4 Agent Strategy

In the literature, an agent is generally selfish which means that it cooperates with its neighbour agents when it can and needs. This feature appears to be very important in WSNs as an agent cooperates only if there is a gain. The gain is a relative point that depends mainly, in our domain, on the sensor node life time and the relevance of cooperating with other agents. In the present work, we propose a strategy for the agent to compute the relevance of cooperation. For this strategy, we define the main parameters that may influence the relevance of the participation in cooperation. These parameters are as follow:

- The energy (E);
- The network density (D);
- The position of a sensor node within the network (P);
- The information importance (I).

We will like to note also that the relevance (R) will be computed with an approximation of a constant (C). In addition, each one of the defined parameters has its importance factor or its priority. We express the defined parameters by the equation (1) to compute the relevance of cooperation.

$$R = E \times \alpha + \frac{1}{D} \times \beta + \frac{1}{P} \times \theta + I \times \omega + C \tag{1}$$

Where α, β, θ, and ω are the importance factor for the energy (E), the density (D), the position (P), and the information importance degree (I) respectively.

4.1 Energy

The energy is an important parameter in a resource limited network such as the WSN. It is generally seen as the most important parameter. Indeed, the remaining battery level appears to be the most important thing in this parameter but it is not the only one. In order to better use the energy of a node, we define an administrator power strategy (APS) parameter. This parameter allows the network administrator to extend its WSN life time. By multiplying the available battery level by a percentage, the agent will reject some cooperation requests that it would be accepted if the administrator strategy was disabled. This rejection allows sensor nodes to save more energy, hence it extends the whole network life time.

Otherwise, the importance of the administrator power strategy could be emphasized also in the case of multi-application sensor network where the administrator will be able to define the importance of each application. Hence, the value of (E), is given by equation (2):

$$E = A \times APS \tag{2}$$

Where E is the energy parameter in equation (1), A is the available or remained power in the battery and APS is the administrator power strategy parameter, which is a percentage defined by the administrator depending on the application. As presented in the equation (2), the energy E could be just the remaining battery level if no administrator power strategy has been defined (APS=1). In addition, lower is the APS longer is the life time of the WSN.

4.2 Network Density

The network density varies from a deployment to another one and from a node to another one within the same deployment depending on the node distribution. This parameter appears as an important parameter to take into consideration. Indeed, more an agent has neighbours, less is the importance of its participation in a cooperation; that is why in the equation (1), we take the inverse of the density (D). To illustrate the importance of this parameter, let us take the example of a tracking application where the position of the desired object can be defined by at least three agents. Finding these three agents in a dense network is an easy task. However, if ten agents participate in this task, in place of three, we will have an undesired loss of power and time.

The density in this case is a relative parameter computed by each agent. There are two main reasons behind that: the first one is that each agent has a local view and there is no agent with a global network view. The second reason, which is more important, appears in the fact that for a specific task, we need cooperation between the agents of the local task zone and not farther agents. For simplicity's sake, we propose the following equation (3) to compute this density (D).

$$D = \frac{N_{real}}{N_{theoretical}} \tag{3}$$

Where $N_{theoretical}$ is the theoretical number of nodes and it is given from the ideal distribution of the nodes or the grid distribution (see Fig. 2(a)). $N_{theoretical}$ corresponds to the number of nodes within the radio range of a reference node (RN). The RN is a node in the centre of the area to eliminate the special cases of border nodes.

N_{real} is the real number of neighbour nodes, which means the number of its one hop neighbour nodes appearing on its MAC layer. It should be equal, in the ideal case, to all the neighbor nodes within the radio range of the node. In Fig. 2(b), we show an example of randomly distributed nodes to give an idea about real network densities.

4.3 Position within the Network

The third parameter is the position of the agent node (P) in the network. We define three types of node positions: (1) normal, (2) edge and (3) critical. The normal position is the position inside the network where the node has multiple neighbours. The edge node is a node in the border of the network and/or having a view of the network limited to one and only one neighbour. A node is considered in a critical position if it connects two parts of the network. That means, if the node runs out of battery, it may divide the network or multiple nodes behind it will become unreachable and in the best case they will require a longer route to communicate their data to the sink. This longer route is expensive in term of energy as the number of hops is increased. Fig. 2(b) presents a random deployment of 100 nodes and we have marked some of the nodes that are in critical position.

The strategy should allow an agent in a critical position to decrease its power consumption to maintain the maximum possible the connection between the two parts of the network. Thus, the value of the importance factor of this parameter should be equal or higher than the energy or the information importance degree factor.

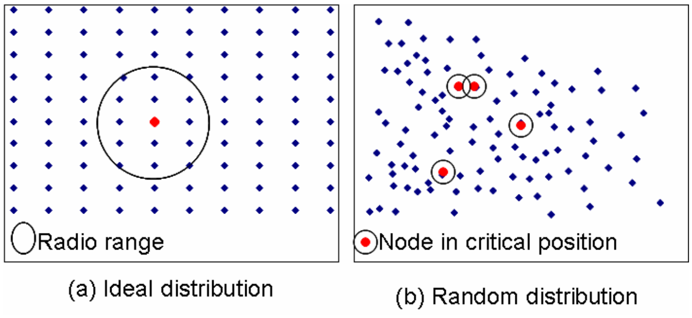


Fig. 2. Network topology

4.4 Information Importance Degree

The last parameter (I) is the information importance degree that depends heavily on the desired application. This parameter could be computed by a local processing in the node. This processing allows the agent to estimate the importance of the gathered information.

For example, in a tracking application, if the detected object is the desired one or in the case of a visual application if the captured picture contains an animal face (supposing we are searching for new species in a forest), the agent will judge this information as important.

In other domains as in environmental monitoring (Humidity, temperature, etc.), the agent saves the last gathered information to compare it with the new gathered one. If the difference between the two is greater than a predefined threshold, this information will be considered as important. Conversely, the agent drops the old information and saves the recent one and marks the information as unimportant. The same technique could be used also in tracking when the object stays in the sensor zone during two or more gathering cycles.

5 Simulation Setup

To evaluate the relevance of our proposed agent strategy, we have carried out a set of simulation tests. These simulations compare the performance of our previous work [8] (IBC) where no agent strategy has been used, with a Strategy Based Communication (SBC), which is a communication based on our agent strategy. The IBC, presented in section 3, is a communication architecture based on the importance of the information. We have implemented these two approaches on GlomoSim [11] which is a scalable simulation environment for wireless and wired network systems.

In our simulation setup, as presented in Table 1, we summarize the different simulation parameters that we have used during the evaluation of our proposition. We have run our simulation over a 1000mx1000m square with a random distribution of nodes during 1000 seconds. We have limited the radio range and the data rate of each node to 87m and 1Mbps respectively as suggested in [9]. The transmission and reception powers' parameters, which influence directly the radio range, have been chosen carefully from the ranges defined in the sun SPOT system technical document [10].

In order to test the scalability of the agent strategy and its relevance across different network densities, the simulations are done for a number of nodes varying from 100 to 900 nodes with an interval of 200.

The local processing time is inspired from the work realized in [1][1], where the processing code is put in a message sent by the sink. This message is the mobile agent. Indeed, transferring this code from the message to the node and placing it in the appropriate place of the memory will take some time. We have estimated this time to 10 ms. The authors of [1] have fixed the processing time to 50 ms which means that 40 ms will be sufficient in our proposition as a local processing time.

Table 1. Basic simulation parameters

Simulation Parameters	Values
Network size	1000mx1000m
Node distribution	Random
Radio range	87m
Throughput	1Mbps
Size of sensed data	24 byte per node
Sensed Data Interval	10 seconds
Simulation time	1000 seconds
Local processing time	40ms

Table 2. Agent strategy equation parameters

Agent Strategy Parameters	Values
Threshold of R	0.7
α, θ, ω	0.25
β	0.1
C	Random [0,0.15]

The agent strategy parameters are presented separately in section 4. The importance factors of these parameters α, β, θ, and ω are fixed to 0.25, 0.10, 0.25, 0.25 respectively. The constant C is a random variable between 0 and 0.15. These values reflect the importance of their correspondent parameters. By giving the same value to α, θ, and ω, we give the same importance to the energy, position and information importance degree, in the calculation of the relevance value. The density has been given a lower priority compared to them as we suppose that it does not influence

[1] This work is presented in section 3.

directly the performance of the whole WSN. Based on these factors and through several simulations, we have found that the majority of cooperation relevancies (R) were between 0.6 and 0.8. Thus, we have chosen to fix threshold of R to 0.7. These parameters are resumed in Table 2.

5.1 Power Consumption Computation Model

The power consumption is a main performance criterion in our work. It represents the average value of energy consumed by each sensor node during the simulation to transmit, receive, and process the data. For the transmission and reception, we use the equation (4), defined in [9]. E_{TX} is the power consumed during transmission and E_{RX} is the power consumed during the reception. Both of them are computed following the data length and distance of transmission (radio range of the node) (l,d):

$$\begin{aligned} E_{TX}(l,d) &= lEc + led^{s} \\ E_{RX}(l,d) &= lE_{c} \end{aligned} \qquad \text{where } e= \begin{cases} e_1 & s=2, \quad d<d_{cr} \\ e_2 & s=4, \quad d>d_{cr} \end{cases} \tag{4}$$

Where E_c is the base energy required to run the transmitter or receiver circuitry. A typical value of E_c is 50 nJ/bit for a 1-Mbps transceiver; d_{cr} is the crossover distance, and its typical value is 86.2 m; e_1 (or e_2) is the unit energy required for the transmitter amplifier when $d < d_{cr}$ (or $d > d_{cr}$). Typical values of e_1 and e_2 are 10 $pJ/bit{\cdot}m^2$ and 0.0013 $pJ/bit{\cdot}m^4$ respectively.

For the local processing consumption, we use the rules defined in [9]. The authors evaluate this energy based on the number of instructions and the frequency of the processor. In IBC and SBC, we use the processor defined in the sun SPOT technical document [10], which sets the processor frequency of their sensor nodes to 180 Mhz. According to [9], a processor with such frequency consumes approximately 0.8 nJ per instruction.

6 Results and Analysis

In this section, we present the simulation results to highlight the performance of our proposition. We show the advantages of the proposed agent strategy scheme by comparing IBC to SBC (which is an amelioration of IBC). It is important to bear in mind that in SBC we base our agent strategy only on the information importance.

We focus mainly on the efficiency of our proposition in terms of power consumption and scalability in different network densities. As presented in the simulation setup section, we have varied the number of nodes from 100 to 900.

In Fig. 3, we plot the average power consumption per node in IBC and SBC. The results show that SBC decreases the power consumption comparing to IBC. In addition, it is clear that the saved power is more important for the higher number of nodes. These results prove that our agent strategy is significantly better designed for scalable or dense networks than the IBC approach. Indeed, for a number of nodes varying from 100 till 900, the power consumption obtained by using the agent strategy is in average reduced by a factor of 1.5 which means an important amount of saved power.

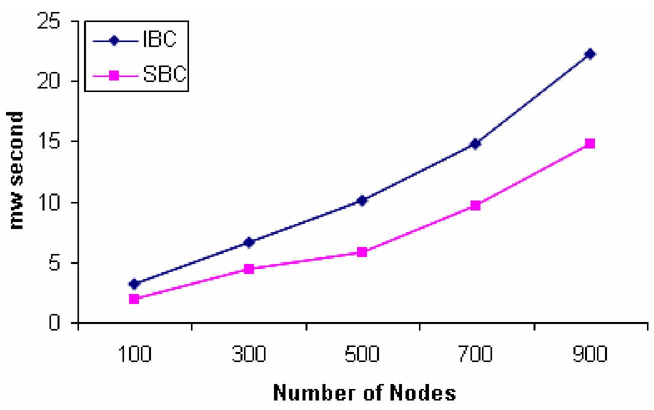


Fig. 3. Average power consumption per node

To show the effects of the administrator power strategy (APS) on the life time of the WSN, we ran a set of simulations with four different values of APS: 25%, 50%, 75%, and 100%. These percentages, as presented in section 3, define for the agent the amount of energy that it can use to evaluate the relevance of cooperation.

Fig. 4 shows that when APS is equal to 75% (Fig. 4), the power consumption per node is divided by two approximately comparing to APS=100% (no strategy). Hence, the network life time has been multiplied by two approximately.

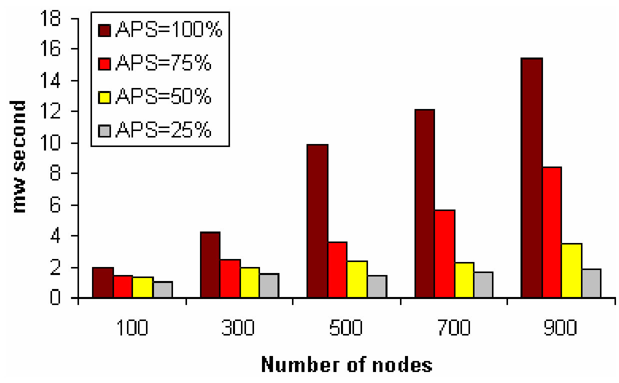


Fig. 4. Average power consumption per node for different administrator strategies

In addition, the results confirm that lower is the APS longer is the life time of the WSN. We can also observe that the network scale and density do not degrade the optimization of the network life time. The definition of this parameter depends highly on the type of the application and the WSN administrator strategy.

Fig. 5 compares the power consumption per node for the nodes in critical positions in both approaches. As we can observe, the agent strategy decreases the average power consumption of these nodes in an important manner. It shows also that more the network is dense more the amount of decreased power is important. We can observe also, that for 700 nodes, the agent strategy divided by two the consumption of these nodes and for 900 nodes, this optimization remained important where the agent strategy divides the consumption by more than 1.5. In addition, in a non dense network, the power consumption has been divided by a factor varying from 1.5 to 2. Hence, we can deduce from these curves that the agent strategy offers a better power management for nodes in critical positions independently from the network scale and density.

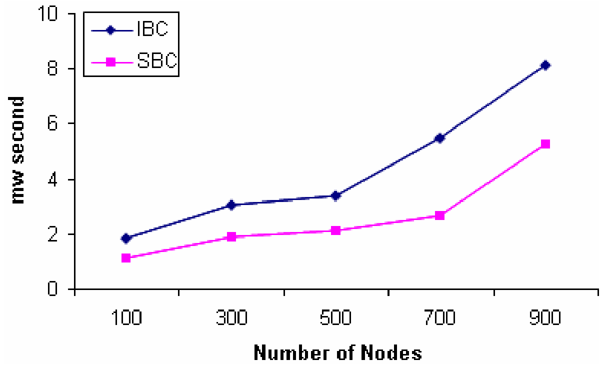


Fig. 5. Average power consumption per node in critical positions

7 Conclusion

In this paper, we had presented an agent strategy to allow the agent implemented in the sensor nodes to cleverly manage the power consumption of the sensor nodes. This optimizes the cooperation of agents in wireless sensor networks. This optimization is based on allowing the agent to take into consideration multiple parameters existing in its local view. In addition, to the available sensor node battery, this strategy deals with the position of the sensor node within the WSN and specially the critical position of a node (when a node connects two parts of the network). It takes into consideration the density of the network around the sensor node and the information importance degree. This strategy allows computing the relevance of a cooperation of an agent with other agents.

The importance of this strategy has been studied comparing to IBC through several simulations. The results illustrate the performance of our strategy. Indeed, SBC has proved an important optimization in term of average power consumption per node and an important management of the power of nodes in critical positions.

As a future work, we think that a mathematical model could be interesting to study the possibility of using a variable cooperation relevance threshold. Then, this agent strategy will be studied in the case of multiple applications over the same physical WSN. We aim also to explore the possibility of using an agent based routing protocol.

References

1. Chen, M., Kwon, T., Leung, V.C.M., Yuan, Y.: Mobile Agent Based Wireless Sensor Networks. Journal of Computers 1, 14–21 (2006)
2. Ferber, J.: Multi-Agent System: An Introduction to Distributed Artificial Intelligence. Addison Wesley Longman, Harlow (1999)

3. Gan, L., Liu, J., Jin, X.: Agent-Based, Energy Efficient Routing in Sensor Networks. In: AAMAS. ACM, USA (2004)
4. Li, Y., Xu, J., Zhao, B., Yang, G.: A New Mobile agent architecture for wireless Sensor Networks. In: Industrial Electronics and Applications, ICIEA, pp. 1562–1565. IEEE, Los Alamitos (2008)
5. Liu, H., Liu, C.: An Intelligent Agent Routing over Wireless Sensor Networks. In: IEEE Proceedings of MASS 2006, pp. 358–366. IEEE, Los Alamitos (2006)
6. Marsh, D., O'Hare, G.M.P., Ruzzelli, A., Tynan, R.: Agents for Wireless Sensor Network Power Management. In: ICPP Workshops, pp. 413–418. IEEE Computer Society, Los Alamitos (2005)
7. Sardouk, A., Merghem_Boulahia, L., Gaiti, D.: Agents Cooperation for Power-Efficient Information Processing in Wireless Sensor Networks. In: Networking and Electronic Commerce Research Conference, Italy (2008)
8. Sardouk, A., Merghem_Boulahia, L., Gaiti, D.: Agent-Cooperation Based Communication Architecture for Wireless Sensor Network. IFIP Wireless Days/Ad-hoc and Wireless Sensor Networks. EAU (2008)
9. Sohraby, K., Minoli, D., Znati, T.: Wireless Sensor Networks, Technology, Protocols, and Applications. Wiley-Interscience, Hoboken (2007)
10. Sun Microsystem, Sun Labs: Sun^{TM} Small Programmable Object Technology (Sun SPOT) Theory of Operation (2008)
11. U.P.C. Laboratory and W.A.M. Laboratory: Glomosim: A scalable simulation environment for wireless and wired network systems. In: The 3rd International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks (2005)

Flow Monitoring in Wireless MESH Networks

Cristian Popi and Olivier Festor

MADYNES - INRIA Nancy Grand Est - Research Center
615, rue du jardin botanique
54602 Villers-les-Nancy, France
{popicris,festor}@loria.fr

Abstract. We present a dynamic and self-organized flow monitoring framework in Wireless Mesh Networks. An algorithmic mechanism that allows for an autonomic organization of the probes plane is investigated, with the goal of monitoring all the flows in the backbone of the mesh network accurately and robustly, while minimizing the overhead introduced by the monitoring architecture. The architecture of the system is presented, the probe organization protocol is explained and the performance of the monitoring framework is evaluated by simulation.

1 Introduction

Community, as well as operator based wireless mesh networks are gaining market shares at a fast pace in the area of Internet access. Easy deployment and the absence of any need of centralized coordination make them a promising wireless technology [4]. The centerpiece of such a network is the wireless distribution system, named the backbone of the network. Such a backbone comprises a mesh of fixed nodes which play the role of either access points, or routers (when they relay packets for other routers), or both of them. The nodes in the backbone communicate in a multi-hop fashion over the wireless links established between them and based on technologies like WiFi or WiMax. The mesh structure provides robustness and availability: if one or multiple nodes fail, the packets will be rerouted on different paths by the "alive" nodes.

While these networks provide flexibility, their heterogeneous nature (for the case of community networks) comes with a range of challenges and difficulties; performance, interoperability and security being some of them. Second, mesh elements may operate under severe constraints such as reduced bandwidth, significant signal quality fluctuation due to environment conditions: obstacles, interferences, or hidden hosts. When designing a monitoring infrastructure for a wireless mesh network, one should therefore provide for situations like broken links, out of range nodes (weakening of the radio signal), or failed nodes. The solution should adapt to the new situation by choosing a different node for monitoring, or a different export path for the monitored information, for instance.

Flow monitoring is a well established approach to monitor the behavior of a network. On one hand, it is used for user monitoring, application and service profiling. On the other hand, certain flows may need to be treated differently from others (ie. separate queues in switches or routers) for traffic shaping, fair queueing, QoS; monitoring the flows in the network is therefore essential in order for such (traffic) handling decisions to be taken.

In this paper we investigate an algorithmic and protocol mechanism that allows for an autonomic organization of the probes plane, that scales well and is robust to node and link failure. The monitoring of the flows is done by the nodes in the mesh backbone. We make the assumption that flows are generated from and directed to mobile nodes (clients of the network) or to the other networks (Internet) via a node acting as a gateway. All the segments of a flow traverse the same path. The paths of flows going through the backbone, pass by one ore more mesh routers, which are all potentially probes. If more than one probe monitor the flow (case of MeshFlow in [8]), than the cost of monitoring increases. This becomes important especially in what concerns the network overhead at export time. This work looks into finding a mechanism for node self-organization such that all flows are monitored only once in their passage through the mesh backbone, while adapting to topology changes generated by loss of links or nodes. The solution relies on the knowledge of routing information for each flow on any mesh node, by requiring that each node distribute its routing table records to the other nodes in the backbone. For this, the underlying routing protocol must be a pro-active one, in which the routing tables are built based on topology knowledge and before any packet needs to be sent across the network (ie. OLSR [7], or DSDV [13]).

The rest of the paper is structured as follows: section 2 introduces related work. Section 3 presents the architecture of the flow monitoring framework. The protocol for node self-organization is shown in section 4 and routing information distribution and the decision making process in section 5. Section 6 provides the evaluation of the proposed monitoring system, and the conclusions end the paper in section 7.

2 Related Work

[10,12] and [5] all present active approaches towards monitoring the state of the network. In [10], K. Kim introduces a scheme for accurate measurement of link quality in a wireless mesh network. The proposed architecture, EAR (Efficient and Accurate link-quality monitoR) uses distributed and periodic measurement of unicast-based unidirectional data probes by dynamically choosing one of three schemes: passive, cooperative and active to measure link quality. In [12], a greedy solution to probe station placement in a network is provided, to detect all node failures in the network. At each step of the algorithm, a node is added to the probe station set such that the set of shadow nodes (the set of the nodes in the networks that can not be reached by k independent paths from the probe stations, where k is the maximum number of node failures) is minimized. Like in our work, adaptability mechanisms are offered.

A self-configurable cluster-based monitoring system for a wireless mesh network is presented in [15]. Monitoring nodes are organized into a cluster-based hierarchical structure in a dynamical way; topological monitoring information, gathered from passively listening to OLSR (Optimized Link State Routing Protocol [7]) traffic is pushed up the hierarchical structure to a topology monitoring entity (which corresponds to a cluster head). In a flow monitoring scenario, flows would be monitored by all the nodes on their paths. We look rather at how we measure available data across the system by coordinating the participant nodes, such that flows are monitored by only one node.

In [16] the problem of sampling packets in a cost-effective manner is proposed, by solving the two conflicting optimization objectives: maximization of the fraction of IP flows sampled vs. minimization of the total monitoring cost. The solutions of the posed problems determine the minimal number of monitors and their optimal locations under different sets of constraints. In [6] Chaudet et al. also found that in terms of cost (of deployment and running of probes), it may be worthwhile to monitor only a part of the traffic (ie. 95%) (this reduces the number of probes needed to almost half in the examples presented in their simulations). These last two approaches are theoretical and they require a priori knowledge about the topology of the network, and the traffic flowing through the network. We aim to build a system that automatically and dynamically organizes the monitoring probes plane to capture at a minimum cost a high percentage of the traffic passing the network without knowing the flow graph in advance.

[8] proposes a monitoring architecture for IP flows in a wireless mesh network. MeshFlow records are being created on every mesh router in the path of a packet. By aggregation of these records a complete transportation path of packets can be deduced. Our proposal minimizes the export overhead by choosing one probe only on the path of a flow to monitor it.

In [14], Gonzalez et al. build a network monitoring scheme, A-GAP, with accuracy objectives. To reduce the overhead of monitored information between the monitoring nodes and the management station the authors introduce a filter scheme by which a monitoring node does not send updates for small variations of its state or partial aggregate state computed on it. The filter is dynamically computed on each node based on a discrete-time Markov chains stochastic model. The results lead to the conclusion that accepting small errors in monitoring accuracy, overhead reduction is gained.

3 Architecture

The routers are all possible probes, that is they all have flow counting capabilities. In order for the probes to make decisions on which one monitors a flow, a global vision of the routing entries of all the nodes in the backbone is required on every node. This allows a probe that sees a flow passing through its interface to trace the flow's path. A prerequisite for monitoring a flow is that a probe P that sees a flow F on one of its interfaces has to know the flow's entry and exit points in the backbone, as well as the next hop towards the exit point for each node on the path of the flow. In accordance with this, the functional architecture of the monitoring system is presented in figure 1. The routing plane builds up the routing table of the mesh nodes (with the help of a pro-active routing protocol). It then provides the routing table entries of all nodes to the monitoring

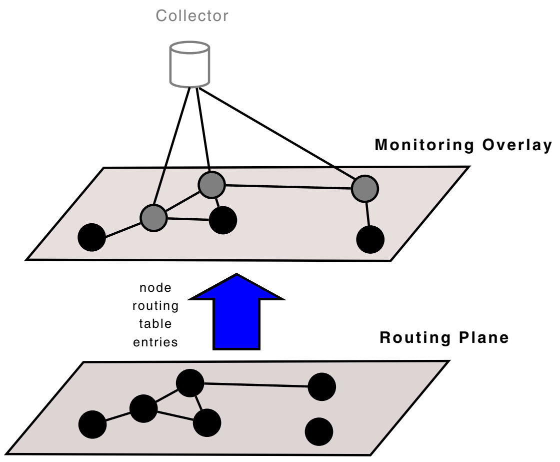


Fig. 1. Wireless Mesh Network flow monitoring architecture

overlay, which uses this information to organize the nodes into monitoring or non-monitoring probes. Two components come into the decision making process when organizing the nodes for monitoring: the routing information received from the routing plane to locally build the path of a flow on a node, and the metrics that allow to differentiate between nodes located on the path of the flow. These metrics are distance (in number of hops) of the node from the collector (to which the node is configured to send flow records), connectivity degree and up link quality. Nodes with better distance, higher connectivity or up link quality are the ones elected to monitor the flow.

The building blocks of a node's architecture are: the neighbor discovery service, the Multi Point Relay (MPR) manager, the global routing service and the flow monitoring service.

The **neighbor discovery service** gives information about the one hop and the two-hop neighbors. It relies on periodically polling neighbor nodes via Hello messages; this enables the tracking of alive nodes and of changes in the topology.

The **Multi Point Relay manager** computes, locally, for each node, the set of Multi Point Relay nodes (MPR) which are used to flood management messages from a node into the network. The MPRs (an OLSR concept [7]) selected by a node are the 1-hop neighbors of this node that allow it to reach all of its symmetric 2-hop neighbors. The use of MPRs to flood information into the network is an optimization of a classical flooding algorithm. This service relies on the underlying neighbor discovery service and is used, in our monitoring architecture, by the global routing table service for flooding local routing table information.

The **global routing table service** makes a view of the routing tables of all nodes available to any node by optimized flooding of messages. The messages flooded into the network by a node k contain tuples of this form: $\{NextHop_i(k), Node\_ID\_List\}$, with $i = 1, N_k$, where N_k stands for the number of entries in k's routing table, and $Node\_ID\_List$ is a list containing all the destinations that are reached from k via a next hop node $NextHop_i(k)$. Apart from routing information, other information local to a node is spread into the network: the distance (in number of hops) from k to the collector (the node, or station which collects all flow export messages sent by probes, and which can be reached independently by all probes by configuring them with the IP address of the collector), the connectivity degree of k (which is the number of direct neighbors it has), and the up link quality (the quality in bandwidth of the path from k to the collector).

The **flow monitoring service** takes decisions per flow on whether a node monitors the flow, based on the information provided by the global routing table service. The source and destination of the flow are extracted from packet headers to deduce the entry and exit points in the backbone. From the routing pieces of information of all nodes in the backbone available locally on each node, the entire path of the flow is constructed. A comparison of the metrics of all the nodes on the flow's path (which are provided locally as well) with the node's own metrics, gives the node the means for deciding if it monitors the flow. The flow monitoring service also handles the flow cache management and flow exportation.

We exemplify the proposed monitoring system, and how it works. In figure 2, there are 3 traffic flows which pass the backbone of the mesh network formed of nodes A, B, C and D. The collector is a station attached to node C. Therefore, node C is the better positioned probe, only 1 hop away from it. Node B is 2 hops away, and nodes A and D, 3 hops. After having exchanged routing table entries, and their distance from the collector, the probes (A, B, C and D) are now capable of deciding what flows to monitor. For instance, for flow F_{1I} nodes

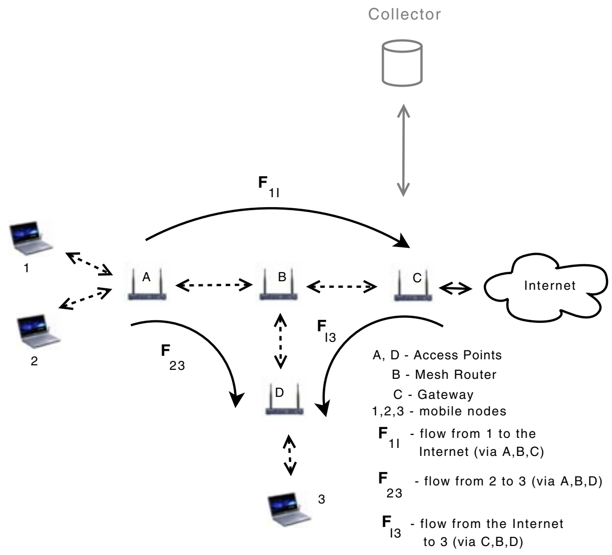

Collector
F1I
1
A
B
C
Internet
2
F23
FI3
D
3
A, D - Access Points
B - Mesh Router
C - Gateway
1,2,3 - mobile nodes
F1I - flow from 1 to the Internet (via A,B,C)
F23 - flow from 2 to 3 (via A,B,D)
FI3 - flow from the Internet to 3 (via C,B,D)

Fig. 2. Wireless Mesh Network – flow example

A, B and C all share the knowledge that the entry point is A, the exit point is C, and B is the intermediary node, and also know the distance of each of them from the collector. Since C is the closest, probes A and B decide, locally, not to monitor F_{1I}. Node D, does not have any decision to take for flow F_{1I}, since this flow does not pass any of its interfaces. The same applies to flow F_{I3}, which is monitored by C, and F_{23}, monitored by B. In the figure, the collector has been chosen to be attached to node C. Therefore node C is privileged to monitor all flows originating or destined from/to the Internet. This might mean a gain in the probe selection cost which is due to the fact that most flows are destined or originated to/from the Internet ([9] states that practically all traffic is to/from a gateway). It also means that node C can become a bottleneck, slowing down normal traffic in the network. The problem would be agravated for a large topology; in that case, it would be best to chose it's location away from the gateway.

The monitored flows, are exported by each of the probes to the collector for storing/analyzing. The export paths from the probes to the collector are common with the traffic paths in the backbone of network. In figure 2, for router A to export monitored flow information to the collector, the multi-hop path via B and C is used, in competition with regular traffic. Since bandwidth provisioning can be an issue, it is important to coordinate the probes' monitoring decisions such that a flow is not counted multiple times on its path through the backbone and that the path from the probe that monitors the flow to the collector is the most convenient. In the provided example, all flows are monitored only on B and C.

4 Probe Organization

Probe organization in a flow monitoring context means organizing the nodes such that monitoring decisions can be made within the confines of the cost associated with the number of monitoring nodes per flow defined in section 1. Probe organization in a non-reliable environment also means adapting the monitoring system to changes in the topology, which, although not so frequent as in mobile ad-hoc networks, still occur due mainly to node start/shut-down/failure or to unstable links (ie. caused by interference or obstruction).

For the nodes to share a global view of the routing tables of all the backbone nodes, a distribution system is needed, to help disseminate the routing table entries. Normal flooding (see OSPF [11]) is not optimal, especially where we send routing table information, which for a proactive routing protocol includes all backbone nodes as destinations. We suppose a mesh architecture, where clients are hidden behind NATs installed on each router (see the Roofnet project [1]); routing tables on backbone nodes therefore only have entries for all mesh routers and a default entry for the gateway. The number of records to be sent per node is roughly equal to the number of nodes in the backbone. Data organization optimizations can be made to reduce this number, by including lists like: $\{Next\_hop_i : Destination\_Node\_ID\_List\}$ in a flooded message sent by node k, instead of $(Destination\_Node_i : Next\_hop)$ pairs, where

$Destination\_Node\_ID\_List$ represents a list of all the nodes in the backbone that can be reached from k via a $Next\_hop_i$ node. In addition, k's neighbors need not be sent explicitly, if they are already a next-hop for some destination in the network.

In order to reduce the number of control messages we use the concept of Multi Point Relay (MPR) employed by the OLSR routing protocol to flood topology control messages. The MPR Set selection scheme is that of OLSR ($Hello$ messages are used). For flooding the network with routing entries, routing control message (RC) are sent by every node and broadcast via the MPRs, containing the entire routing table of the sender.

Whenever a node joins, or fails, the topology changes. The system self-adapts, in the sense that the one and two-hops neighbor lists stored inside each node change their components and the MPR Set is recomputed. Similarly, a new global routing table vision on each node is recomputed with the change of routing tables.

Configurable parameters allow the tuning of the granularity of topology changes as seen by the monitoring system. The $Hello\_Period$ is the duration of time between two successive $Hello$ messages sent by a node. The RC_Period is the duration of time between two successive RC messages. We determine the optimum values for these parameters through simulation with the NS-2 simulator (see section 6).

5 Flow Monitoring Decision

A node floods the network with Route-control (RC) messages (UDP messages with the TTL set to 255) via its computed MPR set. An RC message contains a node's routing table records and a sequence number specific to the sending node. The sequence number is used by the node to rule out the possibility of outdated information.

The RC_Period is the duration of time between two successive RC messages sent by a node. After a node A receives an RC message from a node B, it updates the routing table record information for B , and keeps this information for a RC_Store_Period time duration. If it doesn't receive a RC message from B during this time it will consider B is lost and will discard all routing information from it. This does not affect the accuracy of the monitoring system, since if B is lost, or there is no path from B to A, there cannot exist a flow that passes both A and B.

5.1 Data Representation and Manipulation

The content of an RC message encloses a list of all ($Next\_hop$, $Destination\_List$) pairs contained in the routing table of the RC message sender, and the metrics it advertises (distance from the collector, degree of connectivity and up link quality). A receiving node stores this information in a tuple ($Node\_ID$, $\{Next\_Hop\_Node_1, Dst\_List_1\}, ...\{Next\_Hop\_Node_n, Dst\_List_n\}$). Dst_List_i represents the list of destination nodes reachable from $Node\_ID$ via $Next\_Hop\_Node_i$.

```
Input: $Entry\_Point$, $Exit\_Point$
Output: Flow path: the sequence of Node IDs that can potentially monitor the flow
1 $Current\_Node = Entry\_Point$;
2 $Destination\_Node = Exit\_Point$;
3 $Path = \{Current\_Node\}$;
4 while $Current\_Node$ is not $Destination\_Node$ do
5     $Next\_Node = \text{search\_next\_node}(Current\_Node, Destination\_Node)$;
6     $Current\_Node = Next\_Node$;
7     $Path = Path + \{Current\_Node\}$;
  end
```

Algorithm 1. Algorithm for building the path of the flow through the backbone of the wireless mesh network

The procedure for determining the path of a flow passing through a node's interface is shown in algorithm 1. In line 5, the function search_next_node() returns the next-hop node for the current node towards the destination. Flow end points corresponding to mesh clients are mapped to the routers they connect to. If the flow has one of its end points outside the wireless mesh network (ie, other networks), then either the entry or exit points of the flow are mapped to the gateway. Since the current node holds routing information for all nodes in the backbone, there is exactly one entry in its table that yields the pair (destination: next_hop), no matter what node in the backbone the destination is.

Once the complete path of a flow is obtained, the node compares its metrics to those of the nodes on the paths. The distance from the collector is the number of hops a message sent from the node to the collector has to travel, and it is computed by periodically using traceroute to the collector. The connectivity degree is the number of one-hop neighbors of a node. This can be easily deduced from the size of its one-hop neighbor list. In case both collector distance and connectivity degree are identical for two nodes, an up link quality metric, which estimates the quality of the first hop link towards the collector from the node is used to choose between the nodes. As soon as a node finds that there is another node on the flow's path with better metrics, it gives up comparing, and does not monitor the flow.

6 Evaluation

In this section, we evaluate the effectiveness of our approach. We first describe simulation settings and parameters and then show the results of the evaluation.

6.1 Simulation Setup and Scenario

We simulated a mesh network using the NS-2 simulator [2]. For the MAC layer we used the 802.11b with RTS/CTS (Request to Send/Clear to Send). The

RTS/CTS mechanism is enabled to effectively realize multi-hopping in the backbone. For the routing protocol, we chose OLSR, and use the UM-OLSR patch [3] for ns-2. We conducted our experiments on a variety of network topologies with sizes of between 20 and 50 nodes, and then for scalability reasons, 100 nodes. Nodes in all topologies are fixed, to simulate the static routers in the wireless mesh network backbone.

The traffic pattern is modeled such that 80% of the traffic is directed to or coming from the gateway nodes ([9] states that practically all traffic is to/from a gateway). TCP flows are used to emulate user traffic, with two-way TCP agents randomly disposed over the backbone nodes, and application-level data sources (traffic generator) attached to the agents. We use two-way TCP because it implements SYN/FIN connection establishment/teardown.

6.2 Accuracy

The accuracy measures how close to the proposed goal the monitoring system is, in terms of the number of monitored flows, and the number of monitoring probes per flow.

We first measured the accuracy for a stable mesh network (topology did not change during one experiment and the routing tables on all of the nodes were stable). We conducted 5 experiments for a network size of 20 nodes, and another 5 for a network size of 50 nodes. As expected, all flows were counted exactly once. The average number of probes for different topologies that monitored flows was 43% of all probes. The minimum was 34% and the maximum 66% (the values are averaged over the two network sizes).

Next, we compute the accuracy for an instable mesh network topology. We simulated this by adding 5 nodes, one node every 5 minutes to the low-density network (size=20). The total number of TCP flows sent in this experiment was 3000. The experiment lasted 30 minutes. We repeated it 5 times for different topologies (same network size). The first node was added after 5 minutes. The average non-monitored flow percentage was of 5.3%. This is explicable because of the chosen policy: not to monitor flows, when the full flow path can not be reconstructed. We have intentionally set OLSR in our simulator to be more reactive to topology changes than the monitoring overlay (*TC_Interval* is set to 5 seconds and *RC_Period* to 7 seconds). Instability occurs immediately after the change in topology, and before all the nodes have a knowledge of the new routing information. Figure 3 shows an example of a scenario where this situation occurs. Node D joins, the links A-D and D-E are active, and the routing protocol chooses to route packets from A to F, through D and E, instead of through B and C. Flows from A to E were monitored by node C (as it is the closest to the collector) before the change in topology. As it takes some time to propagate all routing information to all nodes (the *RC_Period* is set to 7 seconds), A, D, E and F only know the next hop for the flow A-E, but not the whole path, therefore they do not monitor any flow that passes through D.

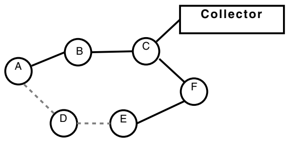


Fig. 3. Topology Instability Scenario - with route for flow A-F changed from A-B-C-F to A-D-E-F

Table 1. Protocol accuracy vs overhead for different pairs of (*Hello_Period*, *RC_Period*)

Hello and RC Period Pair	Accuracy (%)	Overhead (packets/node/sec)
(2,5)	99.2	10.3
(2,7)	93.7	9.8
(3,7)	93.5	6.3
(5,10)	86.9	2.1
(5,15)	62	1.5

6.3 Accuracy vs. Management Overhead

The management overhead measures the number of coordination packets sent by the probes. The only packets sent by the monitoring overlay are *Hello* and *RC* messages.

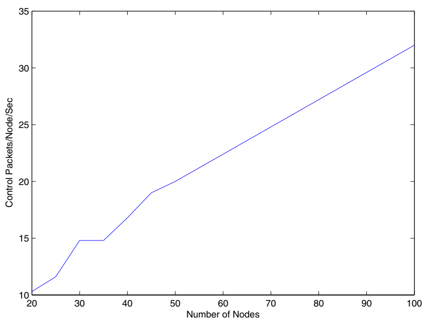
We consider 20 and 50 node mesh networks. We try different values for the *Hello* and *RC* message periods. In all experiments, randomly, nodes fail for a duration of maximum 1 minute (only one node at a time) and then come back alive. This way we simulate more realistically periodic link failure or quality variations. The experiments are all 5 minutes long. The table 1 lists the average values of the accuracy and number of control messages over a 5 minute period for network sizes of 20 and 50 nodes.

The results show that the monitoring system is pretty accurate for a value of *RC_Period* of 10, only missing around 13% of the flows. If the interest is to obtain most of the traffic, decreasing the values of *Hello_Period* and *RC_Period* gives a good monitoring accuracy, but it triples the number of packets sent per node. Further increasing the two parameters yields an accuracy which may no longer be acceptable (62% of flows counted).

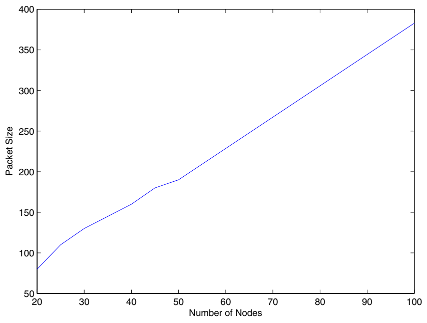
We next see how increasing the number of nodes in the backbone affects the control plane.

6.4 Scalability

The amount of control traffic and message size were measured as the network size increased. We performed a set of experiments, each one 5 minutes long, where we varied the number of nodes from 20 to 50 (in increments of 5) and then 100. Figure 4 shows the effect of increasing the number of nodes in the



(a)



(b)

Fig. 4. (a) the average number of control packets per node per second as node size increases; (b) the average packet size as node size increases

topology to the number of control packets conveyed in the network and their size for a (*Hello_Period*, *RC_Period*) pair of (2,5) seconds.

Both the number and the size of packets vary linearly with the number of nodes. While the number of control messages can be reduced provided the accuracy of monitoring is lowered, the *RC* message size depends on the routing table size of the nodes; with a pro-active routing protocol, each node holds entries for all nodes. Therefore, *RC* messages, even though optimized to include a node only once (see section 4) still holds at least all backbone nodes' identities.

7 Conclusions and Future Work

In this article we have proposed a novel flow monitoring system for Wireless Mesh Networks with the goal of counting all of the flows passing through the backbone on only one mesh node, aiming to reduce export overhead. We build a monitoring overlay, in which a probe is able to decide if it monitors a flow that passes through its interfaces. We have described a protocol for probe organization that aims to reach the proposed goal. The protocol is based on a simple and minimal overhead routing table information flooding by all the probes. By sharing a global routing table view, the probes can trace the paths of the flows within the backbone, and therefore know what other nodes compete for monitoring the flows it sees. The metrics that make the difference in the decision process are chosen to minimize the flow record export: the distance from the collector, the connectivity degree and the up link quality.

We have evaluated the monitoring system to see its accuracy and the impact of the introduced management overhead. We determined the values of the intervals for sending control messages (*Hello* and *RC* messages) for a random number of topologies with 20 and 50 nodes, that optimize the number of management packets while keeping the accuracy of the monitoring system within acceptable limits.

In the future we plan to implement the proposed monitoring system on a small scale 802.11 mesh testbed, that we have set up in the lab. Secondly, since the monitoring system is based on the metrics advertised by the participating nodes, we investigate the situation where one or more nodes maliciously send faulty metrics, to either avoid monitoring (to save processing and communication bandwidth) or to monitor all traffic flows with the intention of eavesdropping, for instance. Another track of action is to find means to dynamically compute the values of the control message intervals, to adapt to the topology dynamics of the network. Recommended values (like the ones computed by us) are too general to fit the specificity of a community wireless mesh network (which depends a lot on conditions like geography or sources of interference).

Acknowledgement

This paper was partially supported by the French Ministry of Research project, AIRNET, under contract ANR-05-RNRT-012-01.

References

1. MIT Roofnet - 802.11b mesh network implementation, `http://pdos.csail.mit.edu/roofnet/doku.php?id=design`
2. The Network Simulator NS-2, `http://www.isi.edu/nsnam/ns/`
3. UM-OLSR patch for ns-2, `http://masimum.dif.um.es/?Software:UM-OLSR`
4. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. Computer Networks 47(4), 445–487 (2005)
5. Bejerano, Y., Rastogi, R.: Robust Monitoring of Link Delays and Faults in IP Networks. In: INFOCOM (2003)
6. Chaudet, C., Fleury, E., Guérin-Lassous, I., Rivano, H., Voge, M.-E.: Optimal positioning of active and passive monitoring devices. In: CoNEXT, Toulouse, France (October 2005)
7. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). RFC Experimental 3626, Internet Engineering Task Force (October 2003)
8. Huang, F., Yang, Y., He, L.: A flow-based network monitoring framework for wireless mesh networks. IEEE Wireless Communications 14(5), 48–55 (2007)
9. Jun, J., Sichitiu, M.L.: The nominal capacity of wireless mesh networks. IEEE Wireless Communications 10(5), 8–14 (2003)

10. Kim, K.-H., Shin, K.G.: On accurate measurement of link quality in multi-hop wireless mesh networks. In: MobiCom 2006, pp. 38–49. ACM Press, New York (2006)
11. Moy, J.T.: OSPF Version 2. Request for Comments 2328, Internet Engineering Task Force (April 1998)
12. Natu, M., Sethi, A.S.: Probe station placement for robust monitoring of networks. J. Netw. Syst. Manage. 16(4), 351–374 (2008)
13. Perkins, C., Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In: ACM SIGCOMM 1994, pp. 234–244 (1994)
14. Prieto, A.G., Stadler, R.: A-GAP: An adaptive protocol for continuous network monitoring with accuracy objectives. IEEE Transactions on Network and Service Management (IEEE TNSM) 4(5) (June 2007)
15. Sailhan, F., Fallon, L., Quinn, K., Farrell, P., Collins, S., Parker, D., Ghamri-Doudane, S., Huang, Y.: Wireless mesh network monitoring: Design, implementation and experiments. In: Globecom Workshops, 2007 IEEE, November 2007, pp. 1–6 (2007)
16. Suh, K., Guo, Y., Kurose, J., Towsley, D.: Locating network monitors: Complexity, heuristics and coverage. In: Proceedings of IEEE Infocom (March 2005)

Visualization of Node Interaction Dynamics in Network Traces

Petar Dobrev, Sorin Stancu-Mara, and Jürgen Schönwälder

Computer Science, Jacobs University Bremen, Germany
{p.dobrev,s.stancumara,j.schoenwaelder}@jacobs-university.de

Abstract. The analysis of network traces often requires to find the spots where something interesting happens. Since traces are usually very large data-sets, it is often not easy and time intensive to get an intuitive understanding of what happens within a given trace. Through the use of suitable data visualization techniques, it is possible for humans to identify noteworthy spots or characteristics of a trace much faster. Particularly interesting properties of a certain class of network traces are node interaction dynamics, that is how the traffic matrix between nodes evolves over time and the pattern of messages exchanged between nodes. This paper presents some tools visualizing node interaction dynamics that were developed to assist network trace analysts.

Keywords: Network measurement, network visualization, SNMP, NetFlow.

1 Introduction

The collection of network traces is necessary in order to understand how network protocols are used in operational networks and to validate simulation models that are frequently used to evaluate new protocols. While the collection of network traces is technically relatively straightforward, it turns out that the analysis of the collected data is challenging due to the volume of network traces and the difficulty to identify the spots of a trace where interesting observations can be made. The later problem can be tackled by generating suitable visualizations so that the perception capabilities of humans can be exploited to identify spots effectively.

Most network trace analysis tools provide graphical displays of traffic over time, showing traffic breakdowns in different time resolutions. While useful to get a first overview, such traffic plots are not sufficient in order to develop a deeper understanding of the exchanges captured in a network trace. For understanding certain traces, it is essential to develop an understanding of the interaction dynamics. A simple example are network or port scans where the scanning strategy reveals some insights about the tools an attacker might have used.

Another example are traces covering a specific type of traffic that is expected to exhibit a certain behaviour. Some of our previous work has been related to the collection and analysis of network management traces, and in particular SNMP traces [1,2]. Initial results were published in [3] where we showed some simple static visualizations of the SNMP topologies discovered in our traces. While the simple static visualizations proved to be useful (one trace showed some unexpected anomalies due to dynamic address assignments), the static topology visualizations do not provide insights about the interaction dynamics. In particular, it is not possible to determine whether a data collection engine spreads the data retrieval traffic over a polling cycle or sends a burst of polling requests at the start of each cycle. Furthermore, it is not possible to see whether there are topology changes or if there are patterns of topology changes. (Note that topology changes on the management plane are usually caused by devices or links failing or returning back to their normal operational state.) To address these questions, we need to visualize the node interaction dynamics in a way such that it is possible to observe a pattern on a trace spanning days, even though the messages are exchanged with a round-trip time measured in the order of microseconds.

We started our investigation of node interaction visualization techniques by asking the following two questions:

– To what extent can existing tools help in visualizing node interaction dynamics?
– Is it possible to develop generic solutions that can be used for different types of traces?

The rest of the paper is structured as follows. We first review in Section 2 the state of the art in SNMP trace analysis and in network data visualization. We then describe an experiment to use the network animator (NAM) for the visualization of node interaction dynamics of SNMP traces in Section 3. Section 4 discusses a second experiment where we used a Java graph drawing library to visualize topology changes in SNMP traces. We briefly describe how we applied the tools to NetFlow traces in Section 5 before we draw our conclusions in Section 6.

2 Related Work

The Simple Network Management Protocol (SNMP) [4] is widely deployed to monitor, control, and (sometimes also) configure network elements. Even though the SNMP technology is well documented, it remains relatively unclear how SNMP is used in practice and what typical SNMP usage patterns are. In order to get a better understanding how SNMP is utilized by network management applications, an effort was started to collect SNMP traces from operational networks [3]. The Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF) produced an RFC [1] explaining the motivation behind the SNMP trace collection effort and specifying two trace storage formats that ease the exchange of traces between tools.

Static properties of several SNMP traces have been analyzed in [3] and a specific graphical representation has been introduced to visualize static SNMP flow topologies. Subsequent work to understand periodic pattern in SNMP traces, carried out at the University of Twente, made it obvious that a common terminology and a clear set of definitions are needed in order to produce results that are well defined and comparable. The common terminology resulting from this work has been published in [2]. Researchers at the Federal University of Rio Grande do Sul created an online system visualizing SNMP traces. Their system produces SNMP flow topology graphs similar to the graphs introduced in [3], plus MIB tree and time series graphs [5].

Most network data visualization work has focused on developing effective mechanisms to visualize large static data-sets. In the rest of this section we will review related work on topology visualizations and on NetFlow data visualizations.

Early work on the visualization of network topologies was done in the 1990s. The authors of [6] and [7] developed basic radial and geographic representations. More recent work often focuses on the visualization of Autonomous Systems (AS) routing topologies [8,9]. Linear IP address spaces are sometimes displayed in two dimensions using space filling curves [10,11,12], as this technique allows to show aggregations with common prefixes. To visualize network activities, three dimensional cube visualizations have been suggested, mapping source and destination addresses and port numbers [13,14]. These cube visualizations indicate interaction dynamics, although they can be hard to read due to the two-dimensional projection on computer displays.

NetFlow data is usually associated with time series displays such as those generated by RRDtool[1]. Radial layouts for visualizing NetFlow data were suggested in [15], while hierarchical network maps resembling treemaps were proposed in [16]. The work described in [17] proposes to display edge bundles on top of hierarchical network maps. The Isis system described in [18] provides visual flow data representations designed to make temporal relationships apparent and to allow for visual classification of events to reveal traffic structure. Most of these visualization systems focus on the visualization of static properties of NetFlow data. While the Isis system supports the interactive exploration of traces, it does not aim at animating node interaction dynamics.

3 Experiment #1: NAM

In our first experiment, we visualize the exchange of SNMP messages between managers and agents in order to highlight how SNMP monitoring engines distribute the polling load over time. While doing the conceptual design of the trace visualizer, it became apparent that *what* is being displayed should be decoupled from the *how* to display it. For describing what should be displayed we had to write our own tools because the SNMP trace format is relatively new and there are no powerful processing tools readily available yet. The comma separated values (CSV) SNMP trace format defined in [1] is very easy to parse in almost any programming language. Once the traces are parsed the flexibility of the programming language allows for any filter to be described and any output

[1] `http://oss.oetiker.ch/rrdtool/`

format to be used. We have used the programming language Python [19] in our experiment for it's rapid prototyping features. For visualization, we needed a tool that uses an open format and is able to display network topologies (graphs) and messages traversing the topology. We have chosen the popular network animator `nam` due to its ability to visualize the exchange of messages between nodes and our familiarity with this tool.

We first describe the network animator used for the experiment. We then describe how we convert SNMP traces into input files for the network animator before we present and discuss the visualization results.

3.1 Description of `nam`

The network animator[2] (`nam`) is distributed as part of the `ns2` simulation software package. It is mainly used to visualize traces generated by the ns2 simulator. The `nam` tool is implemented in a mixture of C++, Tcl/Tk [20], and an object-oriented extension of Tcl. In general, the animation software is not easy to modify due to the high learning curve involved in understanding the interplay of the different programming languages involved.

The network animator reads a simulation trace file (also called a `nam` trace file), computes a graph layout for the recorded network topology, and afterwards it animates the messages passing over the links connecting the nodes. The graphical user interface provides controls to pause the animation, to change the animation speed, to zoom in and out, and to recalculate the graph layout or to manually reposition nodes.

The `nam` input file uses a relatively simple textual line-oriented format. We only describe a subset of the features here that we have used in our experiment. A `nam` trace file starts with a header containing version information, the list of nodes, and the list of links connecting the nodes. The version line has the following format:

```
V -t * -v 1.0a5 -a 0
```

The character `V` indicates that this is a version definition and the version number `1.0a5` is passed as the argument to the `-v` option. Nodes are defined using lines of the following format:

```
n -t * -a %d -s %d -S UP -v circle -c black -i black
```

The character `n` indicates that this is a node definition. The `-t *` parameter defines that this event does not have a time attached. The `-a` and `-s` parameters define the address and the id of the node. The `-S` parameter defines the node status while the `-v` parameter specifies the shape of the node and the `-c` and `-i` parameters specify the color of the node. Links connecting nodes are defined using lines of the following format:

```
l -t * -s %d -d %d -S UP -r %d -D %f -c black
```

[2] `http://isi.edu/nsnam/nam/`

The character l indicates that this is a link definition while the -t * parameter again indicates that this definition does not have a time attached. The -s and -d parameters specify the id of the source and destination nodes. The -S parameter defines the link status and the -c parameter the link color. The -r and -D parameters specify the data transmission rate of the link and the delay associated with the link. Finally, the color header lines specify the color coding of message types:

```
c -t * -i %d -n %s
```

The character c indicates that this is a color definition. The parameter -t * again indicates the time of the event. The parameter -i specifies the id of the color while the -n parameter carries a color name (according to the X11 color database). Following the nam file header, network message events are encoded using the following format:

```
h -t %f -s %d -d %d -e %d -c %s -i %d -a %d
```

The character h indicates a hop event. The parameter -t specifies the time of the event while the parameters -s and -d specify the source and destination node id of the message. The parameter -e carries the message size while the -c parameter carries a conversation identifier and -i a message id. The color of the message is specified using the -a parameter. Next to hop events, we use receive events. They have the same format in the nam trace file, except that the character h is replaced by the character r.

3.2 Conversion Algorithm for SNMP Traces

In order to use the network animator, we wrote a program to convert SNMP traces into the nam trace format. The program reads SNMP traces in the CSV format defined in [1]. In order to generate the nam header with the complete node and link list, our program needs to read the whole CSV file before nam output can be produced. In order to be able to share visualizations with other researchers and the public, we took care that our converter hides information such as IP addresses and absolute time-stamps.

For each transport address in the SNMP trace file, a node with a unique id is generated. The id is not derived from the transport address contained in the trace file. The links are inferred by determining all pairs of transport addresses that exchanged SNMP messages and translating this to source/destination node ids. For SNMP traces, the number of nodes and links encountered is usually small [3]. Finally, we generate for each SNMP message two nam trace events, namely a hop event and a receive event. The SNMP message events are color coded, allowing network analysts to distinguish easily between different request message types and responses.

An important issue is the time scale. Our visualization is designed to display traces covering weeks. However, in a real scenario, packets travel the network in fractions of a second. Any decent time scale that will allow a user to view the entire data-set will make the SNMP messages almost invisible. For this reason we tweak the network properties in the following way: we change the delay assigned to a link to 60 seconds. This is unrealistic, but it helps to make packets visible for longer time periods since it allows users to select faster than real-time timescales and still be able to see SNMP messages being sent. This change also helps to obtain a general overview over the network because the user actually sees all the data in the last 60 seconds. Another issue is the data rate of the links. It is used by the network animator to display the size of the message (in conjunction with the delay). For this reason, a 150 byte message on a 1Mbps link is too small to be displayed. Therefore we have also decreased the rate of the links down to 1kbps.

3.3 Results and Discussion

The generation of topology layouts is left to the network animator. Our conversion program does not provide any hints to the network animator concerning the graph layout.

Figure 1 shows a typical layout calculated by NAM. In this trace, two manager applications monitor an almost disjoint set of network devices. (Note that this layout is different from the layouts described in [3] since it is based on the network addresses of the nodes involved and does not distinguish different SNMP flows to the same endpoints.)

Figure 2 shows screenshots from a visualization of trace l06t01 (see [3] for more details about this trace). The left part (a) shows a manager polling devices by sending requests at about the same time to many devices. The devices respond with roughly the same delay causing subsequent requests to be sent out

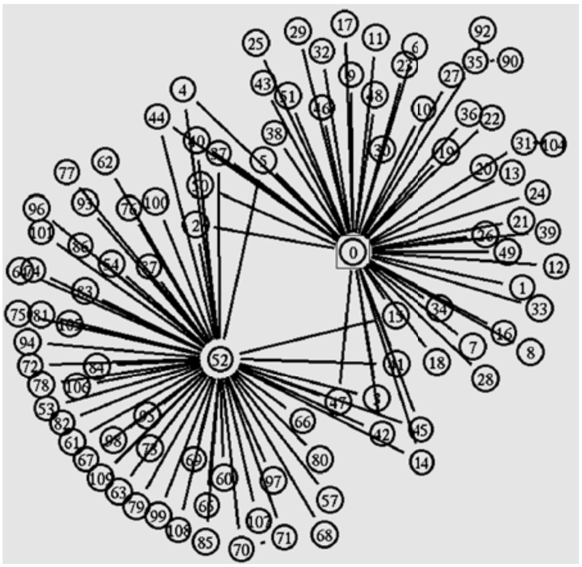
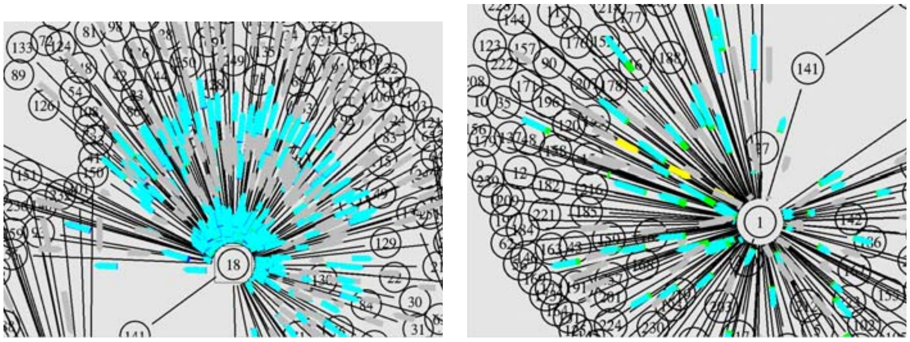


Fig. 1. Network layout calculated by `nam`



(a) Message waves (b) Dispersed messages

Fig. 2. SNMP interaction dynamics visualized in `nam` (trace l06t01)

at roughly the same time as well. This leads to message "waves" in the visualization. The right part (b) shows a manager distributing the messages well over time, thereby avoiding bulky request/response waves. Part (b) also shows notification messages originating from one agent to inform the manager of some events.

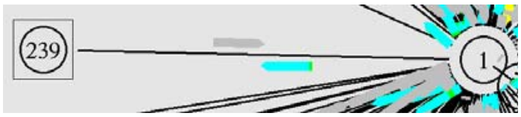


Fig. 3. SNMP message interaction details

Figure 3 shows some details highlighting the visualization of request/response interactions over a single link. Due to the adaptation of the link properties (delay and data rate), a request and the corresponding response usually appear traveling on a link at the same time.

4 Experiment #2: NetViz/JUNG

In our second experiment, we aim at visualizing the changes of the manager / agent topology over time by calculating topology-change dynamics graphs. A topology-change dynamics graph is a undirected graph showing the relation between nodes based on the recorded activity. If a message between node X and node Y has been seen in the last t_{link} seconds, there is an edge between the nodes in the graph. If no activity has been seen on the link for a period of t_{link} seconds, the edge between the nodes is removed. Once a node remains with no active links attached to it (no edges on the graph), a timeout timer is fired and if no link attaches to it for t_{node} seconds, the node is removed from the graph. In general, one can expect that the relationship between SNMP managers and agents is rather static. Changes usually occur if there is a (notable) event in the network (devices powering up / down, notifications emitted that do not happen regularly). To understand a trace, it is therefore a good approach to find such topology changes and to further analyze them.

4.1 Description of NetViz/JUNG

In order to create topology-change dynamics graphs for traces that can be very large in size, we decided to take a two step approach similar to the way the network animator works. We have implemented a graph animation tool called NetViz that reads an ordered timestamped list of events as input (addition of a vertex, addition of an edge, removal of a vertex, removal of an edge) and runs the animation. An additional program is used to read trace files and to produce the intermediate file containing a timestamped list of events. One benefit of this approach (besides increased modularity) is that animation scripts are usually several orders of magnitude smaller than the original trace files they are derived from.

The animation file format is very simple. Each line has the following basic format:

```
<timestamp> <action> <identifier>
```

The `<timestamp>` of the first line contains an absolute timestamp while all subsequent timestamps are relative. The `<action>` is one of `Va`, `Vr`, `Ea`, `Er` (vertex add/remove, edge add/remove) and the `<identifier>` is a unique identifier for the vertex / edge to be added / removed.

For the visualization of the topology, we used the Java Universal Network / Graph Framework[3] (JUNG), a software library for the modeling, analysis, and visualization of data that can be represented as a graph or network [21]. The library supports a number of different graph layout algorithms, clustering algorithms, and user interface controls such as lenses. The library makes it easy to write Java [22] programs that dynamically update the graph by adding nodes and edges while the displayed layout is dynamically recomputed. We added a simple graphical user interface (GUI) with a few controls around the JUNG framework.

4.2 Conversion Algorithm for SNMP Traces

The conversion program reads an SNMP trace in the CSV format defined in [1] and internally constructs a graph. Nodes are identified by their IP addresses and edges are identified by the IP addresses of the endpoints, independent of the direction of message exchanges. To each node and edge, there is a time-to-live counter attached, initialized with the values t_{node} and t_{edge}, passed as command line arguments. The timer for an edge starts running from the moment it is added to the graph, while the timer of a vertex starts running when there are no more edges connecting this vertex to other vertexes. The default value for t_{node} and t_{edge} was set to 600 seconds.

[3] `http://jung.sourceforge.net/`

4.3 Results and Discussion

The graph visualizer reads the animation file and draws the graph as it changes over time. The placement of the nodes on the canvas is done according to a relaxation algorithm, which considers each edge as a spring and finds the node configuration with minimum potential energy. When vertexes or edges are inserted or removed, the new configuration is relaxed until it reaches a minimum again.

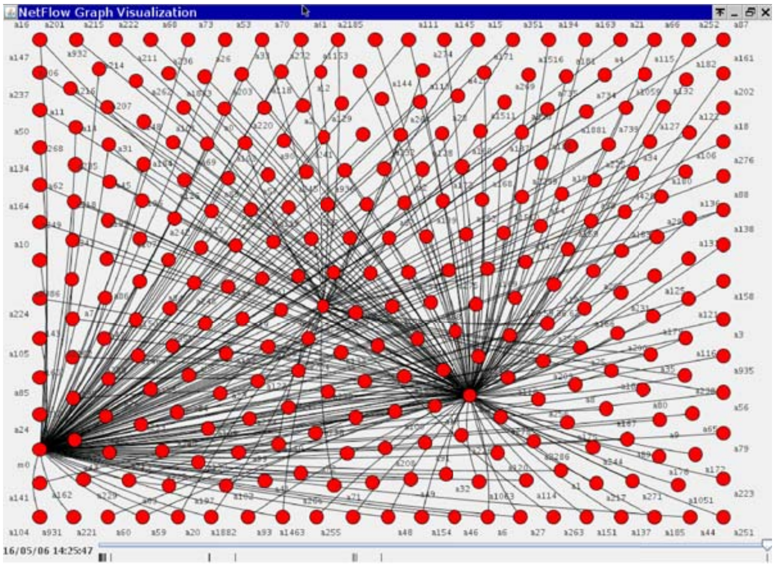


Fig. 4. Dynamic network layout calculated by NetViz/JUNG (trace l06t01)

The GUI of the application, shown in Figure 4, provides a label with human readable time of the replay, based on the timestamps, which are in seconds since the Epoch, and a timeline with the regions of activity marked on it. Since in most of the cases the configuration of the graph remains intact for most of the time and just has several peaks of activity over the whole time period, running the animation at a constant speed is somewhat boring as nothing is happening most of the time. To deal with this, we introduced an idle speed at which the animation runs when there is no activity. Once it gets close to some activity, the animation switches back to normal speed, so that the user can see what changes exactly are taking place on the graph. The normal and idle speed can be passed as command line arguments, or otherwise default to 300 real-world seconds per 1 second animation time for normal speed and 6000 real-world seconds for idle speed.

The application allows for interaction with the canvas by moving it with the mouse (pulling with the hand tool) and zooming in and out with the mouse wheel. Further controls to move the timeline and to pause the animation are possible but have not been realized yet at the time of this writing.

Figure 4 shows a typical situation of a few SNMP managers polling periodically a set of SNMP agents. The nodes are identified by labels indicating whether they act as a manager (labels starting with m) or an agent (labels starting with a). The timeline shown at the bottom of the window indicates when changes happen in the topology. Figure 5 shows a zoom-in on the timeline of the configuration of Figure 4. The topology changes at the beginning of the trace are usually caused by adding nodes and edges until a stable situation has been established and as such usually do not indicate events of special interest. (An option to address this could be to suppress topology change events in the first $t_{startup} = max(t_{node}, t_{edge})$ seconds).

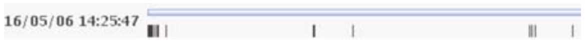


Fig. 5. Enlarged timeline display of Figure 4

Our experience using this tool and, in particular, the adaptive replay mechanism has been very positive. In fact, the activity timeline itself gives valuable insight into the traces since changes to the topology-change dynamic graphs likely indicate events of some significance. A shortcoming of the current implementation is, however, that the removal of nodes and edges can lead to rather drastic changes of the graph layout. To deal with this, we are experimenting with strategies where nodes and edges first become invisible for a while so that a node / edge disappearing and reappearing shortly later does not cause instabilities in the graph layout.

5 Application to NetFlow Traces

The tools described in Sections 3 and 4 were originally developed to visualize SNMP traces. It turned out that it was straightforward to modify the translators so that they can parse NetFlow [23] traces and produce intermediate files for the visualization tools. Of course, flow traces collected at backbone routers need to be filtered and / or aggregated appropriately for the tools to produce meaningful node interaction visualizations.

For the `nam` conversion, some adaptations are required since flow traces only record the start and end of a flow, plus the number of bytes and packets exchanged. In order to enhance the visualization, we decided to choose the size and density of the visualized packets solely based on the duration of a flow. The source emits packets as long as the flow is present. The packets travel slowly in order to enhance the visualization. This also has the nice effect that very short flows will still be visible, so it is possible to go faster through an animation without losing a lot of details.

We are currently interested to understand flow patterns generated by personal hosts and specific applications. In particular, we like to answer the question to what extend flow patterns can be used to identify machines or even users. We assume that flow pattern generated by a certain host or user can serve as a fuzzy fingerprint and we are trying to explore techniques to identify hosts or users based on such fuzzy fingerprints. In order to evaluate whether this is feasible, we started an effort to collect flow records on end user devices such as notebooks. These traces are reasonably small in size and with some limited filtering one can achieve useful visualizations of interaction dynamics in order to identify characteristic patterns.

Figure 6 shows a screenshot of `nam` visualizing the traffic generated by a notebook during one day. The screenshot shows some Web browsing activity of the user of the notebook at a certain point of time. By moving through the timeline, it is possible to easily identify periods of high activity and sites visited frequently. To further improve the visualization results, it is useful to pre-process the NetFlow traces by aggregating IP addresses so that, for example, different IP addresses of Google servers appear as a single node.

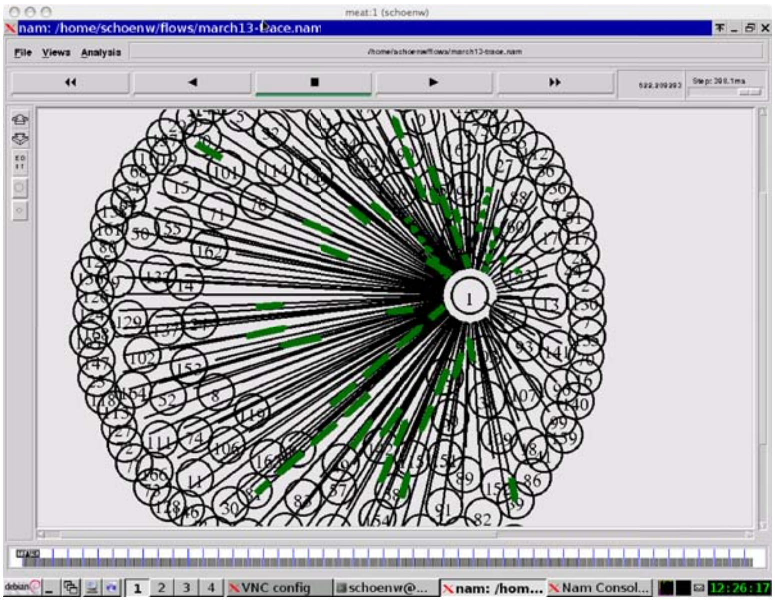


Fig. 6. Visualization of NetFlow traces using `nam`

6 Conclusions

Most existing visualizations of network traces focus on the static properties of the traces. Since our goal is to understand the dynamics recorded in network traces, we did some experiments to create visualizations of node interaction dynamics using readily available tools or libraries.

We found that with relatively little effort, it is possible to adapt tools or libraries to produce meaningful visualizations of interaction dynamics. However, effective integration of the tools / libraries remains a major problem. To be effective, a network analyst must be supported by multiple different visualization views of the same data and user interactions must be reflected on all views. To achieve this, much more development work is needed.

Our experience with `nam` is two-fold: On the one hand, it was very easy to get started since the `nam` intermediate file format is easy to generate and debug. However, the graph layout algorithm used by `nam` is not producing very satisfying results for larger graphs (e.g., ineffective use of the canvas space and many overlapping nodes) and the zooming capabilities help to to deal with this shortcoming only to some extend. Furthermore, we found that the speed of the animation does not scale well for larger trace files. As the `nam` tool is written in Tcl/Tk (with some parts in C/C++), it is not easy to extend it since the effort of learning how the `nam` tool has been implemented is relatively high.

Our experience with the JUNG graph drawing framework has been mixed as well. The graphs produced by the JUNG library generally look nice and using the Java library is rather straightforward due to good documentation and many readily available examples. While the performance of the graph layout algorithms is generally good, we did experience performance difficulties when we tried to animate more complex graphs. This is mainly a Java limitation since drawing on a canvas is relatively costly and does not exploit the capabilities of modern graphics hardware.

While we started with SNMP traces with very specific properties (dominant regular polling traffic with a relative stable communication matrix), it turned out that some of the tools we created could be adapted easily to deal with other trace formats. The decoupling of visualization software from specific data sources through intermediate file formats has proven to be a big win here. Unfortunately, some other openly available visualization tools we looked at do not support such intermediate formats and integrating their visualization capabilities or adapting these tools to different trace formats requires much more involved programming efforts.

Based on the experience we have gained by using `nam` and NetViz/JUNG for the visualization of node interaction dynamics, we are currently implementing a new visualization tool called `snam` that utilizes the Open Graphics Library (OpenGL) [24] and is able to take advantage of the capabilities of modern graphics hardware. We plan to further develop our tool towards an integrated trace visualization environment that consists of a loosely coupled set of visualization and data conversion tools that can be orchestrated as needed for a given trace analysis task.

Acknowledgement

The work reported in this paper is supported by the EC IST-EMANICS Network of Excellence (#26854). Thanks to Nikolay Melnikov for his comments that helped to improve the presentation of this work.

References

1. Schönwälder, J.: SNMP Traffic Measurements and Trace Exchange Formats. RFC 5345, Jacobs University Bremen (October 2008)
2. van den Broek, J., Schönwälder, J., Pras, A., Harvan, M.: SNMP trace analysis definitions. In: Hausheer, D., Schönwälder, J. (eds.) AIMS 2008. LNCS, vol. 5127, pp. 134–147. Springer, Heidelberg (2008)
3. Schönwälder, J., Pras, A., Harvan, M., Schippers, J., van de Meent, R.: SNMP Traffic Analysis: Approaches, Tools, and First Results. In: Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007), May 2007, pp. 323–332 (2007)
4. Case, J., Mundy, R., Partain, D., Stewart, B.: Introduction and Applicability Statements for Internet Standard Management Framework. RFC 3410, SNMP Research, Network Associates Laboratories, Ericsson (December 2002)
5. Salvador, E., Granville, L.: Using Visualization Techniques for SNMP Traffic Analyses. In: Proc. of the IEEE Symposium on Computers and Communications (ISCC 2008). IEEE, Los Alamitos (2008)
6. Schulze, M., Benko, G., Farrell, C.: Homebrew network monitoring: A prelude to network management. In: Proc. SANS II (March 1993)
7. Pleitner, S., Brown, D.: Geotraceman: A Visual Traceroute. In: Proc. Australian Unix User Group (AUUG 1995) (September 1995)
8. Huffaker, B., Plummer, D., Moore, D., Claffy, K.: Topology discovery by active probing. In: Proc. Symposium on Applications and the Internet (SAINT 2002), pp. 90–96. IEEE, Los Alamitos (2002)
9. Boitmanis, K., Brandes, U., Pich, C.: Visualizing internet evolution on the autonomous systems level. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 365–376. Springer, Heidelberg (2008)
10. Munroe, R.: Map of the Internet. xkdc (2006)
11. Irwin, B., Pilkington, N.: High Level Internet Scale Traffic Visualization Using Hilbert Curve Mapping. In: Proc. of the Workshop on Visualization for Computer Security. Mathematics and Visualization. Springer, Heidelberg (2008)
12. Samak, T., Ghanem, S., Ismail, M.A.: On the Efficiency of using Space-Filling Curves in Network Traffic Representation. In: Proc. 11th IEEE Global Internet Symposium. IEEE, Los Alamitos (2008)
13. Lau, S.: The Spinning Cube of Potential Doom. Communications of the ACM 47(6), 25–26 (2004)
14. Oberheide, J., Goff, M., Karir, M.: Flamingo: Visualizing Internet Traffic. In: Proc. 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), pp. 150–161. IEEE, Los Alamitos (2006)
15. Keim, D.A., Mansmann, F., Schneidewind, J., Schreck, T.: Monitoring Network Traffic with Radial Traffic Analyzer. In: Proc. IEEE Symposium on Visual Analytics Science and Technology (VAST). IEEE Computer Society, Los Alamitos (2006)
16. Mansmann, F., Vinnik, S.: Interactive Exploration of Data Traffic with Hierarchical Network Maps. IEEE Transactions on Visualization and Computer Graphics 12(6) (November 2006)
17. Mansmann, F., Fischer, F., Keim, D.A., North, S.C.: Visualizing large-scale IP traffic flows. In: Proc. 12th International Workshop Vision, Modeling, and Visualization (November 2007)

18. Phan, D., Gerth, J., Lee, M., Paepcke, A., Winograd, T.: Visual Analysis of Network Flow Data with Timelines and Event Plots. In: Proc. Workshop on Visualization for Computer Security (VizSec 2007). Mathematics and Visualization, pp. 85–99. Springer, Heidelberg (2007)
19. van Rossum, G., Drake, F.: The Python Language Reference Manual. Network Theory Limited (2003)
20. Ousterhout, J.K.: Tcl and the Tk Toolkit. Addison-Wesley, Reading (1994)
21. O'Madadhain, J., Fisher, D., Smyth, P., White, S., Boey, Y.B.: Analysis and Visualization of Network Data using JUNG. Journal paper under preparation
22. Campione, M., Walrath, K.: The Java Tutorial: Object-Oriented Programming for the Internet, 2nd edn. Addison Wesley, Reading (1998)
23. Cisco Systems: NetFlow Services Solution Guide (January 2007)
24. Segal, M., Akeley, K.: The OpenGL Graphics System: A Specification. Version 3.1, The Khronos Group Inc. (March 2009)

Towards Energy Efficient Change Management in a Cloud Computing Environment

Hady AbdelSalam[1], Kurt Maly[1], Ravi Mukkamala[1], Mohammad Zubair[1], and David Kaminsky[2]

[1] Computer Science Department, Old Dominion University, Norfolk, VA 23529, USA
{asalam,maly,mukka,zubair}@cs.odu.edu
[2] Strategy and Technology, IBM, Research Triangle, Raleigh, NC 27709, USA
dlk@us.ibm.com

Abstract. The continuously increasing cost of managing IT systems has led many companies to outsource their commercial services to external hosting centers. Cloud computing has emerged as one of the enabling technologies that allow such external hosting efficiently. Like any IT environment, a Cloud Computing environment requires high level of maintenance to be able to provide services to its customers. Replacing defective items (hardware/software), applying security patches, or upgrading firmware are just a few examples of the typical maintenance procedures needed in such environments. While taking resources down for maintenance, applying efficient change management techniques is a key factor to the success of the cloud. As energy has become a precious resource, research has been conducted towards devising protocols that minimize energy consumption in IT systems. In this paper, we propose a pro-active energy efficient technique for change management in cloud computing environments. We formulate the management problem into an optimization problem that aims at minimizing the total energy consumption of the cloud. Our proposed approach is pro-active in the sense that it takes prior SLA (Service Level Agreement) requests into account while determining time slots in which changes should take place.

Keywords: Cloud Computing, Autonomic Manager, Change Management, Energy Efficient.

1 Introduction

IT companies have been looking for solutions that enable users to access computing power of supercomputers with thin client devices. The cloud computing concept has emerged as an appropriate solution that attracted the attention of large IT companies such as IBM and Google. A cloud [3, 5] can be defined as a pool of computer resources that can host a variety of different workloads, including batch-style back-end jobs and interactive user applications. A cloud computing platform dynamically provisions, configures, reconfigures, and de-provisions servers as needed. Several commercial realizations of computing clouds are already available today (e.g., Amazon, Google, IBM, and Yahoo, ...etc).

Due to the tremendous increase in energy costs in the past few years, it is expected that efficient power management will play an essential role in the success of large IT environments such as computing clouds. Power management in such environments can be very challenging when hundreds or even thousands of servers are located within a relatively small area. The impact of high power consumption is not only limited to the energy cost but it also extends to the cost of initial investment of cooling systems to get rid of the generated heat and the continuous cost needed to power these systems. In order to reduce operational cost at these centers while meeting any performance based SLAs (Service Level Agreement), several techniques have been proposed in the literature. Among these techniques, we can find hardware techniques such as processor throttling and dynamic frequency scaling and also software techniques such as virtualization and Advanced Configuration and Power Interface (ACPI) standards.

Servers in computing clouds typically go through several software and hardware upgrades. Change management is the process that determines the appropriate time in which these upgrades should take place without violating any of the underlying SLAs. Not only maintaining and upgrading the cloud should occur with minimal disruption and administrative support but it also should go hand in hand with power management to guarantee efficient utilization of the cloud resources.

In previous work published in [1], we proposed and implemented a basic autonomic manager for change management that can determine whether applying a specific change would violate any of the system SLAs. We extended our autonomic manager in [2] by integrating it with a scheduler that can simplify change management by proposing time slots in which changes can be applied without violating any of SLAs requirements. Motivated by the importance of developing energy efficient techniques for computing cloud environments, in this paper we extend our previous work by proposing a pro-active, energy-aware technique for change management in IT environments governed by SLAs requirements such as computing clouds. The remainder of the paper is organized as follows. In section 2, we describe the architecture of our cloud environment. The details of our proactive approach is presented in section 3. Section 4 offers conclusions and describes future work.

2 Cloud Architecture

Figure 1 shows one possible architecture for a cloud. In this architecture, requests from cloud clients get into the system through the cloud gateway. After necessary authentication and based on the current load on the servers, the load balancing module forwards client requests to one of the clouds servers dedicated to support this type of requests. This implies that the load balancing module at the cloud gateway should have up-to-date information about which client applications are running on which servers and the current load on these servers. One possible way to make such information available is to integrate the gateway and load

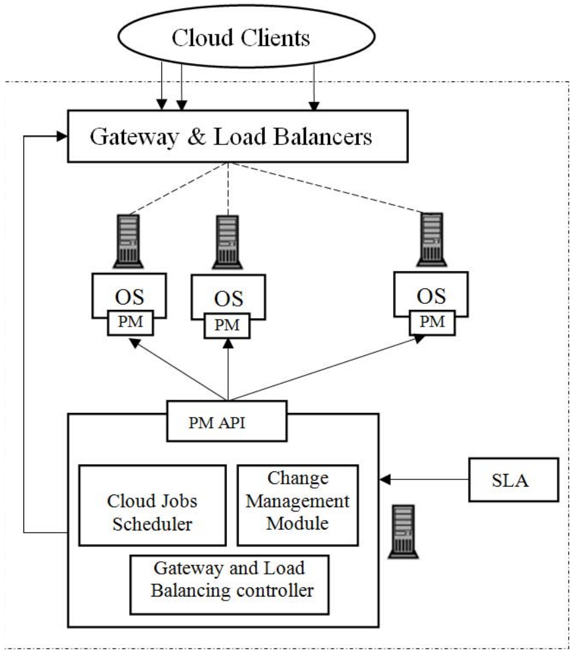


Fig. 1. Cloud Architecture

balancing controller with the job scheduler in order to be able to distribute the load on cloud servers correctly.

From statistical and historical information of client applications, SLA requirements can be mapped into MIPS. The MIPS requirements are then used to determine the number of servers and their operating frequencies in order to minimize total power consumption. Once the jobs scheduler module determines the number of servers and the optimal running frequencies, client applications are being assigned to servers based on the SLA requirements of each client. Typically, this process may involve running several instance of the same application on different servers and distributing requests of users of the same client over these servers based on their current load.

Mapping cloud SLA requirements into MIPS and then into a number of servers and their running frequencies implies that the power management (PM) module at the cloud central controller should interact with the power management module on each server. Through this interaction, the central controller can change the running frequencies of these servers when necessary. In section 3, we show how the number of servers and the running frequencies are actually determined under different cloud workloads.

3 Pro-active Change Management

A cloud consists of one or more server groups; each group has a number of servers that are identical in hardware and software configuration. Thus, all the servers in the same group are equally capable of running any client application. Our proactive approach will be presented in section 3 assuming a homogeneous cloud where all the servers are identical (e.g. one server group). It can be extended straightforwardly to heterogeneous clouds by restricting the users of each client to one server group and looking at the heterogeneous cloud as a group of homogeneous sub-clouds.

Cloud clients sign a service level agreement with the company running the cloud. In this agreement, each client determines its computing needs by aggregating the processing needs of its user applications, the expected number of users, and the average response time per user request. To be able to estimate the computing power (MIPS) needed to achieve the required response time, the client should provide the cloud administrators with any necessary information about the type of the queries expected from its users. One way of doing this is through providing a histogram that shows the frequency of each expected query. Cloud administrators run these queries on testing servers and estimate their computing needs based on response time requirements. Once a client has been running its services on the cloud for some time, the computing needs requirements can be updated from actual statistics from the cloud.

Average response time for a user query depends on many factors, (i.e. the nature of the application, the configuration of the server running the application, and the current load on the server when running the application). The focus of this paper is on clouds supporting interactive applications like web services and web applications. In the future we will address other types of clouds[1]. Under this assumption, we can approximate the minimum average response time constraint as determined by the SLA by the minimum number of instructions that the application is allowed to execute every second. This kind of approximation can be easily done for interactive applications. For example, assuming that the response time for a user query was measured to be t seconds, when the query runs solely on x MIPS server. Now, if we run the same query along with other queries on the same server, to guarantee a minimum average response time of r seconds, the computing power dedicated for the query must be at least $M = \frac{t*x}{r}$ MIPS. Hence, when a new user joins the cloud, the load balancer module should forward requests of the new user to one of the cloud servers such that it has at least M MIPS of its computing power available. Based on client SLAs and the expected number of users for each client, the cloud should pro-actively provide enough number of servers to satisfy required response time.

Advanced hardware-based power management techniques such as dynamic voltage/frequency scaling (DVS) [4] allow servers to run on a wide range of operating frequencies. The relationship between the power consumption and the running frequency takes the form $P = A + B * f^3$, where A and B are constants that depend on the hardware configuration of the server. The most important

[1] For computing intensive applications, the best strategy may be to assign these applications to one or more powerful servers running on their top speed so they can finish as soon as possible. This strategy would give servers a chance to remain idle for longer periods saving their total energy consumption.

feature of the cubic relationship shown above is that the first term in the RHS $[A]$ accounts for power consumption of components that do not scale with the operating frequency, while the second term $[B * f^3]$ accounts for changes in CPU power consumption when the operating frequency changes.

Assuming that the total computing workload of all cloud clients is L_T, and that the cloud has a large number of servers, each of them can run on a range of operating frequencies, we are interested to determine the optimal number of servers to use and the corresponding operating frequency of each server such that power consumption is a minimum. We have proven that the optimal number of servers is given by $k = \sqrt[3]{\frac{2B}{A}} L_T$, and the optimal operating frequency is $f_i = \frac{L_T}{k}$. Unfortunately, the optimal values for k and f_i depend on L_T which in most scenarios changes periodically over time, based on the type and number of user requests determined by SLAs (see figure 2). The length of the period depends on the constituting SLAs (e.g. one day, one week, or even one month). To include the dynamic nature of the total cloud load $L_T(t)$ into our model, we divide each period into what we call running segments or running slots. During the same segment, the cloud total load is fixed, however it can change from one segment to the next. Under this assumption, we can model the total cloud load as shown in the example in figure 2. In that illustration, we assumed a cloud with three different SLAs, and a periodicity of one day. Figure 2 also shows the total cloud load on the system as the sum of the computing load of individual SLAs. Furthermore, we eliminate minor changes in the total cloud load by approximating the load on the cloud by a tight upper bound envelope.

Based on the cloud total load shown in the curve in figure 2 and using the formulae shown above, we can pro-actively evaluate optimal values for k and f_i for all upcoming segments and prepare required resources ahead of time. Hence,

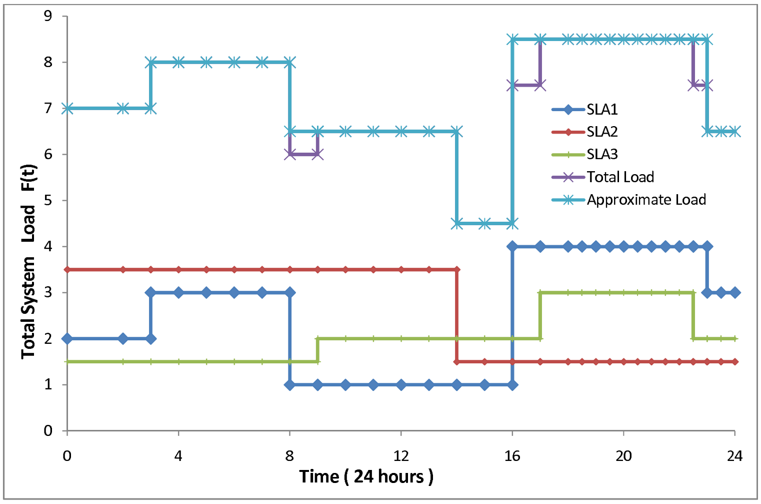


Fig. 2. Actual and Approximated Total Load Due to Several SLAs

when the running segments starts, there will be enough resources to serve users requests satisfying their response time requirements. The number of idle servers in each time segment equals the difference between the total number of cloud servers and $k(t)$. These idle servers are optimal candidates for pending change management requests.

4 Conclusions

In this paper we have described a pro-active energy-aware change management scheme for cloud computing environments that serve primarily clients with interactive applications such as web applications and web services. We used the cubic relationship between power consumption and the frequency at which a server is running to develop an analytical description of the optimal number of servers and the corresponding optimal running frequencies needed to satisfy clients' SLAs. We further describe how these formulae can be used to determine time intervals when servers become available for change management and how many such servers will be available. Thus we have provided mechanisms on how to avoid over-provisioning of clouds and how to minimize power consumption for a given set of SLAs.

References

1. AbdelSalam, H., Maly, K., Mukkamala, R., Zubair, M.: Infrastructure-aware autonomic manager for change management. In: POLICY 2007: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks, Washington, DC, USA, pp. 66–69. IEEE Computer Society, Los Alamitos (2007)
2. AbdelSalam, H., Maly, K., Mukkamala, R., Zubair, M., Kaminsky, D.: Scheduling-capable autonomic manager for policy based it change management system. In: EDOC 2008: Proceedings of the 12th IEEE International Enterprise Computing Conference, München, Germany (September 2008)
3. Boss, G., Malladi, P., Quan, D., Legregni, L., Hall, H.: Cloud computing. In: High Performance On Demand Solutions (HiPODS) (October 2007)
4. Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., Gautam, N.: Managing server energy and operational costs in hosting centers. In: SIGMETRICS 2005: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 303–314. ACM, New York (2005)
5. Erdogmus, H.: Cloud computing. IEEE Software 26(2), 4–6 (2009)

A Market-Based Pricing Scheme for Grid Networks

Peng Gao[1,2], Xingyao Wu[2], Tao Liu[2], and David Hausheer[3]

[1] Graduate University of Chinese Academy of Sciences, Beijing, China
[2] China Mobile Group Design Institute Co., Ltd, Beijing, China
{gaopeng,wuxingyao,liutao1}@cmdi.chinamobile.com
[3] Department of Informatics IFI, University of Zurich, Switzerland
hausheer@ifi.uzh.ch

Abstract. This paper presents a new market-based pricing scheme which aims to improve the link load balance in Grid networks. Simulation results show that the proposed scheme achieves a better link load balance and, thus, improves the network's robustness and stability. At the same time, the scheme increases the network's effective capacity as it enables to accommodate more new services. The results show that the proposed scheme leads indeed to a more efficient usage of network resources.

Keywords: Pricing, market model, load balance, Grid networks.

1 Introduction

Despite the fact that traffic management engineering mechanisms have been investigated both in the area of Internet-based communications [2] as well as in thea area of Grid computing (cf. [3], [4], [5], [6]), the use of economic means for traffic management in Grid networks has received little attention so far. Existing mechanisms mainly focus on traffic optimization from the end user's perspective rather than from the network provider's perspective. Hence, they may not lead to an optimal result from the network point of view. For example, a main drawback of [3] is the dynamic migration of individual, on-going services from one *node* or *link* to another without considering the concurrent Grid services' behavior, which can cause new bottlenecks in the network and – even worse – it may lead to unexpected ping-pong migration in some cases. Moreover, [1], [7], and [8] focus on balancing the load in terms of CPU and memory resources only.

In order to achieve an optimal balance of the traffic load which is benefical for the network provider as well as the network user, this paper proposes a new approach which aims to manage the traffic of Grid networks through economic means. Our contribution is a simple method for balancing the network traffic through a market-based price adjustment scheme.

The rest of this paper is organized as follows: Section 2 presents the proposed pricing scheme, while Section 3 shows and discusses the results of the simulations that have been carried out to show the improvements of the proposed scheme. Finally, Section 4 concludes this paper.

2 Pricing Scheme

The proposed economic traffic management approach is to introduce a logical pricing component in every network domain which is in charge of determining the final link price according to the link load. The network is considered as an overlay network among different participants in a Grid, denoted as peers. It is assumed that metering points are located at each peer which detect and report the current link load to the pricing component in a periodic manner. Based on the information collected, the pricing component decides on the link prices, which shall reflect the current link load fluctuation and finally lead to a balance of the load on all links of the network.

The market-based pricing scheme balances the link load via an adjustment of the link price to provide an incentive to peers to select a path or link with a light load rather than one with a high load. The link price will increase, if the link load is higher than the average level or decrease, if it is lower than the average level.

A path is composed of one or more links for each pair of peers. Each link can be used to transfer data for several other peers. Figure 1 shows an example for a specific source peer (peer 1) and target peer (peer 4) for which there are two potential paths.

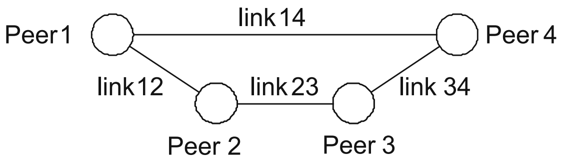


Fig. 1. Path sets for a specific connection from Peer 1 to Peer 4: Path 1 = {link 14}, Path 2 = {link 12, link 23, link 34}

A feasible path solution space can be provided by any routing algorithm. The link resource is modeled as a set $G(V,E)$ which is composed of peers and links. V is the set of the peers, $|V| = n$ is the number of peers, E is the set of links, $|E| = m$ is the number of links. A detailed step-by-step description of the price adjustment algorithm follows:

1. Assuming that the peer α competes or shares the bandwidth in the path P_α which is composed of M links ($M < m$), r_1,r_2,...,r_M is the set of resources; the resource allocation vector is represented as $x = \{x_1, x_2, ..., x_M\}$; the corresponding price of the path is $\bar{p} = \{p_1, p_2, ..., p_M\}$, in which p_1,p_2,...,p_M is the price of the links on the path. The basic link price p_{base} as well as the link price adjustment step p_{step} has to be determined by the network provider.
2. A utility function describes the satisfaction of a peer with the available resources and their link prices. It is assumed that the feasible solution space

is determined before the calculation of the utility function, i.e. the credits owned by a peer is enough to cover the data transmission. The utility function of the transferring path of peer α is defined as follows:

$$U(x) = Min\{(x_i/X) \cdot (C/c_i), i \in P_\alpha\},$$

where X and C are average remaining link throughput and average link price respectively; x_i is the remaining throughput of link i; c_i is the price of link i, P_α is the data transferring path of peer α.

3. According to the result of the routing method, the feasible solution of a set of data transfer paths is determined as $B(\bar{p}) = \{x : \bar{p} \cdot x \leq \omega\}$, where ω is the total credits of the peer or the amount of credits that the peer wishes to pay for the given service and $\bar{p}$ is the price vector of the corresponding path (or links).
4. For those target peers with enough credits it is more likely to have more than one potential path for the data transfer from their corresponding source peer. It is assumed that the size of $B(\bar{p})$ is K. Obviously K cannot be less than zero. When $K = 0$, it means that the peer doesn't have enough credit. If $K = 1$, there is only one potential path $\tilde{p}$ for the service delivery. If $K > 1$, the following procedure will help to select the final path from the potential path set. Before going into the detailed procedure, the potential paths are listed in the order of decreasing utility function:

$$\boldsymbol{U} = \{U(x_1) \geq U(x_2) \geq \ldots \geq U(x_K), x_i \in B(\tilde{p}), 1 \leq i \leq K\}.$$

Instead of simply selecting the path with the highest utility function as the final path, a *persistence factor* is introduced to avoid ping-pong effects in some cases where many peers select one specific link and shift to an another link simultaneously. In this way, each peer will choose the path with the highest utility function with a high probability and it still has the possibility to select the path listed at a lower position in the sequence.
5. The demand function is defined as $\Phi(\tilde{p}) = \{U(\tilde{p})\}$. The total resource demand of link i is the accumulation of the demand of each peer interested in the link i: $d_i(\tilde{p}) = \Sigma_{\alpha \in A} \Phi_i^\alpha(\tilde{p})$, where in $\Phi_i^\alpha(\tilde{p})$ is the resource demand of peer α in the link i under the current price vector $\bar{p}$. A is the set of the peers being interested in link i.
6. The definition of the over-demand function is that the difference between the total resource demand and the supply of link i at the current price $\bar{p}$, i.e. $Z_i(\tilde{p}) = d_i(\tilde{p}) - S_i$, S_i is the remaining capacity of the link i; For all the $i \in E$, if $|Z_i(\tilde{p})| \leq \varepsilon$, in which ε is a predetermined small enough value, the traffic control based on the market model reaches the balance in price vector $\bar{p}$. As long as the resource demand of the link changes, the link price will make a response, i.e., the link price will decrease if $Z_i(\tilde{p}) < 0$ or increase when $Z_i(\tilde{p}) > 0$.
7. After the update of the link price, step 2 to step 6 will be repeated. Finally, the iteration round with the highest total utility value will be chosen as the final result and the price of each link in that round will be set as the final link price.

Of course, in order to guarantee QoS for all services no over-demand or overload is permitted, i.e. the link load or link utilization, normalized by the link capacity, cannot be larger than one.

3 Simulation and Evaluation

In order to verify how much the proposed scheme improves the network resource utility, simulations have been carried out and results are compared with those of the shortest (direct or one hop) path scheme (OSPF).

In order to achieve convincing simulation results, Monte Carlo simulations have been carried out. The network scale is ranging up to 340 peers. It is assumed that each peer can reach any other peer through a direct path or via one relay peer. For the sake of simplicity, only paths with no more than two hops or links were considered.

The traffic requests and link loads have been generated randomly. The initial link prices were proportional to a link's current (background) load. A maximum of 100000 simulation runs were carried out in order to obtain a stable simulation result for all Monte Carlo simulation cases.

The average link load variation of the whole network was chosen to analyze the performance in a quantitative manner. Ideally, in a robust network all links have almost the same if not identical level of load, i.e. the variation of the link load sequence is zero. A larger difference between link loads will lead to a larger variation of the link load sequence. Therefore, a smaller variation value indicates a more robust and stable network.

As it can be seen from Figure 2, the market-based pricing scheme outperforms the OSPF scheme in terms of network link load balance, since it reduces the

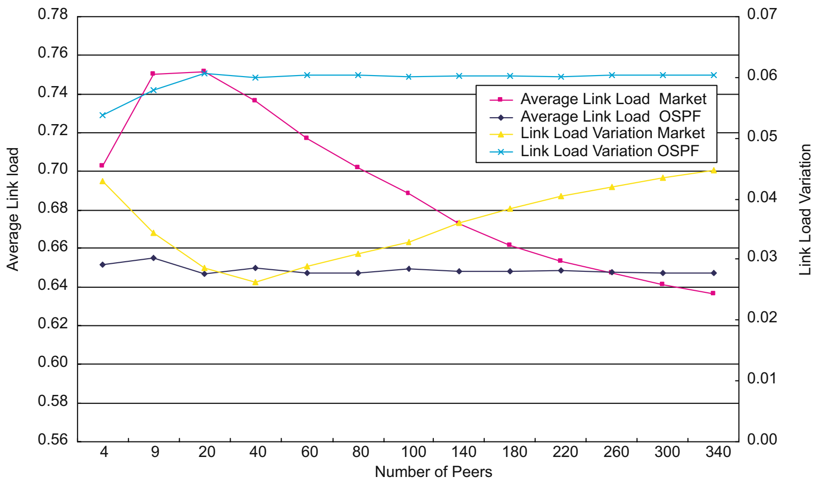


Fig. 2. Performance comparison of the market-model based pricing scheme with OSPF in terms of link load variation and average link load

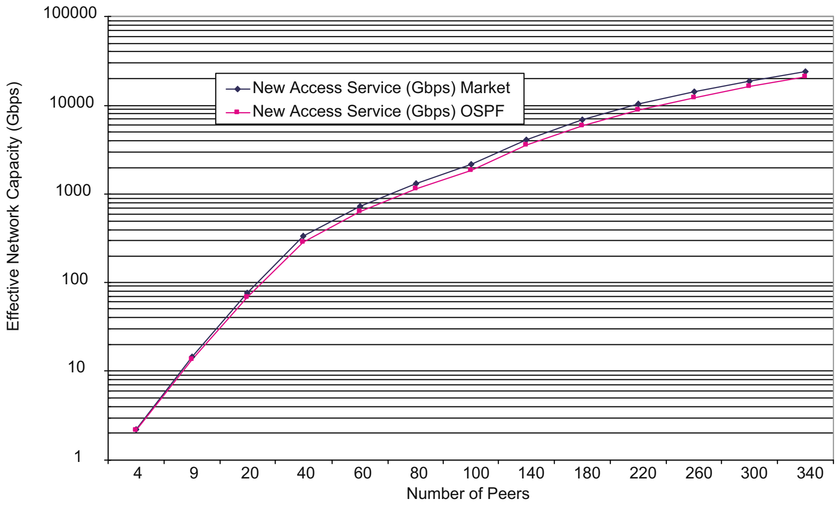


Fig. 3. Performance comparison of the market-model based pricing scheme with OSPF in terms of effective network capacity

Table 1. Time consumption of the proposed pricing scheme

# Peers	10	50	100	150	200	250	300
Time (sec.)	0.00375	0.0228	0.17	0.5706	1.4044	2.7874	4.8874

average link load variation ranging from 20% to 50% based on the number of peers in the network.

Moveover, the average link load with the proposed scheme is higher than that of the OSPF scheme when the network peers are less than 200, which shows that the price adjustment scheme improves the network resource utilization. In addition, the network's effective capacity is increased. As Figure 3 shows ten percent more services can be accommodated in the network when the number of peers is larger than ten.

Finally, the computational complexity of the proposed pricing scheme has been measured by the time consumption of the price adjustment algorithm. As expected, Table 1 shows that the computational complexity of the scheme increases with the increasing number of peers within a network.

4 Conclusion

This paper presented a market-based pricing scheme for Grid networks. Simulation results show that the proposed scheme improves the network's robustness and stability. This is due to the fact that the average link load variation is reduced compared to the OSPF scheme. Furthermore, the pricing scheme increases the network resource utilization as it allows to accommodate more services compared to the OSPF method. The average link load with the proposed scheme is higher than that of the OSPF scheme when the network peers are less than 200. Moreover, the simulation results show that ten percent more services can be accommodated in the network when the number of peers is larger than ten.

Future work aims at extending the simulations with more hops between peers and improving the scalability of the proposed scheme by dividing the network into multiple domains.

Acknowledgments. This work has been performed partially in the framework of the EU IST project EC-GIN under the contract STREP FP6-2006-IST-045256. Additionally, the authors would like to acknowledge discussions with all of their colleagues and project partners.

References

1. Allenotor, D., Thulasiram, R.K.: A Grid Resources Valuation Model Using Fuzzy Real Option. In: Stojmenovic, I., Thulasiram, R.K., Yang, L.T., Jia, W., Guo, M., de Mello, R.F. (eds.) ISPA 2007. LNCS, vol. 4742, pp. 622–632. Springer, Heidelberg (2007)
2. Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., Xiao, X.: RFC 3272: Overview and Principles of Internet Traffic Engineering, IETF Informational RFC (May 2002)
3. Batista, D.M., da Fonseca, N.l.S., Granelli, F., Kliazovich, D.: Self-Adjusting Grid Networks, IEEE ICC (June 2007)
4. Lin, F.C., Keller, P.M.: The Gradient Model Load Balancing Method. IEEE Transsactions on Software Engineering 13(1), 32–38 (1991)
5. Loh, P.K.K., Hsu, W.J., Cai, W.: How Network Topology Affects Dynamic Load Balancing. IEEE Parallel and Distributed Technology 4(3), 25–35 (1996)
6. Palmer, C., Steffan, J.: Generating Network Topologies That Obey Power Law. In: IEEE GLOBECOM (2000)
7. Sandolm, T., Gardfjäll, P., Elmroth, E., Johnsson, L., Mulmo, O.: An OGSA-Based Accounting System for Allocation Enforcement across HPC Centers. In: 2nd International Conference on Service Oriented Computing (ICSOC 2004) (November 2004)
8. Wu, S., Jin, H., Chu, J.: A Novel Cache Scheme for Cluster-Based Streaming Proxy Server. In: IEEE ICDCS Workshops (June 2005)

Consistency of Network Traffic Repositories: An Overview

Elmer Lastdrager and Aiko Pras

University of Twente, The Netherlands
e.e.h.lastdrager@student.utwente.nl, a.pras@utwente.nl

Abstract. Traffic repositories with TCP/IP header information are very important for network analysis. Researchers often assume that such repositories reliably represent all traffic that has been flowing over the network; little thoughts are made regarding the consistency of these repositories. Still, for various reasons, the traffic capturing process may have missed packets. For certain kinds of analysis, for example loss measurements, such inconsistencies may lead to the wrong conclusions.

This paper proposes an algorithm to detect such inconsistencies, using the idea of "fake gaps". A prototype has been developed, and used to test two well-known repositories: the WIDE and Simpleweb repositories. The paper shows that both repositories contain several inconsistencies.

1 Introduction

A network traffic repository contains network traffic gathered from one or more location(s), often a router or backbone. Captured traffic is stored in data files in a repository; typically such files contain data captured over a longer period, for example minutes, hours or even days. Repositories can store different types of network data, for example TCP/IP header files, netflow records or SNMP packets. In this paper the focus is on the most common type: TCP/IP header data.

Using a repository can be very convenient for a researcher, as gathering data yourself can be very time-consuming, or even impossible. A potential issue in using a repository, is the consistency of the traffic inside the repository. When traffic inside a repository does not completely correspond with the actual traffic that was transmitted and received, this can influence measurements, analysis and therefore also conclusions that researchers draw. Hence, it is critical to have information about the consistency of the network traffic repository, so it can be taken into account when analysing the data.

Issues with consistency of the data in repositories have been reported by M. Timmer in [1]. While using a repository, it appeared that not all data was recorded properly. Timmer introduces the term "fake gap" to represent those parts of a TCP flow that are absent in the repository, although they were acknowledged at the TCP level. In [2] a relatively simple algorithm has been developed to find a sudden decrease in data in small intervals, which may indicate a problem with the repository. However it may very well be a temporary network problem and therefore does not necessarily affect the consistency of the repository. Although [2] has performed some initial research, the consistency analysis never exceeded a few data files. However, as researcher, knowledge about inconsistency in a repository is essential. At this moment, there are no statistics about possible inconsistency of repositories available. In this paper we will therefore analyse two well-known repositories: the WIDE and Simpleweb repositories. A tool has been developed that analyses TCP flows. We focus on TCP, because its state-full nature allows detection of fake gaps; for UDP this is, due to its stateless nature, not possible.

In this paper, the main question that will be answered is: *How can inconsistency be detected in a TCP traffic repository?*

To answer the main question, we first focus on detecting fake gaps by introducing an algorithm. The sub question we will answer is: *How can we detect fake gaps?*

Next, we built a prototype of the algorithm to test existing repositories in order to answer the second sub question: *How consistent are today's repositories?*

It should be noted that an earlier version of this paper has been presented at the tenth Twente Student Conference on Information Technology [3]. That conference is an internal conference of the University of Twente, of which the proceedings have not officially been published by a real publisher. This paper is short version of the earlier paper, and it is the first publication for an international audience.

The structure of this paper is as follows. In Section 2 we will propose an algorithm to detect inconsistency. In Section 3 we will discuss the results of testing two existing repositories using a prototype we built and finally in Section 4 we will answer our research questions, draw conclusions and discuss possible future work.

2 Detecting Inconsistency

In this Section, we will describe inconsistency called fake gaps and introduce an algorithm for detecting inconsistency. Although there may be different ways a repository can be inconsistent, we will only consider fake gaps.

2.1 Fake Gaps

In this paper's introduction, we said a fake gap to be representing those parts of a TCP flow that are absent in the repository, although they were acknowledged at the TCP level. To explain this more precisely we first have to consider a gap. To start with an example: when Alice wants to send some data to Bob, she sends ordered packets named A B C D E F. Bob may receive this as A B C E F **D**, but knows about the correct order and can therefore recreate the original message. This is called packet reordering. At the side of Bob, after having received C, there is a gap until D is received. In this example, the gap is filled when D is received.

We call a gap a fake gap, when one or more packets in a sequence of packets are not present, but are also not retransmitted; hence the original TCP flow is not affected. Consider Alice and Bob: Alice sends the sequence A B C D E F to Bob using TCP. Bob's network administrator records all traffic sent to Bob. According to the data recorded, Bob received A B C E F. When the connection between Alice and Bob is closed, we know Bob must have received D. We can then say that the recorded TCP flow between Alice and Bob contains a fake gap. All data was transmitted and received correctly, but the recorded data does not reflect this. Hence the recorded data is inconsistent. From now on, if we are referring to the flow of data packets within a TCP connection, we will abbreviate it to a flow.

In recorded traffic data, it is likely that there are flows that start and/or end outside the recorded time period. Therefore, detecting fake gaps in these flows is difficult. To avoid this, we only take flows into account that are sufficiently recorded. All packets of a single flow, from the first SYN-packet up to a FIN- or RST-packet should be recorded. If this is the case, we call the TCP flow a *usable flow*. Note that the final FIN-handshake can be partly outside the recorded time period.

2.2 Algorithm

In Section 2.1 we concluded fake gaps prove inconsistency. To detect this the algorithm we introduce, listed as Fig. 1, first extracts all usable flows from the complete set of packets. The second part of the algorithm loops over all usable flows. For every usable flow, it checks if there is an acknowledging packet that acknowledges a packet that has not yet been seen. If so, this indicates a fake gap and an identifier of the flow together with the packet that is used to detect the fake gap, is added to a list. So the final result of this algorithm is a list of flows combined with packets directly after the fake gaps.

It is important to see that this algorithm alone cannot detect the exact amount of fake gaps within the traffic dump. It can give an indication and show which usable flows are affected. If, for example, during a small interval no packets were recorded at the recording device, multiple flows can be affected by this. Our algorithm would detect a fake gap for each affected flow. We do not consider this a limitation. The results of our algorithm should provide a starting point for deep inspection of packets and flows.

3 Analysing Repositories

We made a prototype to analyse repositories using the algorithm from Section 2.2. We analysed two existing repositories using our prototype. The first repository we used was Simpleweb [4], maintained by the University of Twente. It was included in this paper because it is publicly available and easily accessible. We analysed a subset of data covering all 6 locations the Simpleweb repository provides, totalling to 224 data files. The second repository we included in this paper is the WIDE traffic repository [5], maintained by the MAWI working group. It contains data files with traffic of several trans-Pacific lines. We used one data file from samplepoint A to test our prototype and we analysed 27 data files from

```
INIT usableFlows to {}        # all usable flows found
INIT testFlows to {}          # all flows found so far
FOR each packet in data file
    IF packet is SYN
        CREATE flow from packet
        ADD flow to testFlows
    ELSE IF packet belongs to flow in testFlows
        SET flow to flowOf(packet)
        ADD packet to flow
        IF packet is FIN or RST
            REMOVE flow from testFlows
                ADD flow to usableFlows
END FOR
INIT fakeGaps to {}
FOR each flow in usableFlows
    FOR each packet in flow
        IF packet is ACK
            IF acked packets are not in this flow
                ADD tuple (flow, packet) to fakeGaps
    END FOR
END FOR
```

Fig. 1. Pseudo-code of the fake gap detection algorithm

samplepoint F, which were all over 1 GB in size. We chose samplepoint F since this one is daily updated while the other samplepoints are discontinued or were not accessible. The full table of results created by our prototype, and the prototype itself, can be found at [6]. Our prototype calculates fake gap statistics using various intervals. That is, fake gaps from different flows within this interval are grouped together and reported as a single fake gap.

Figure 2 shows the estimated number of missing packets per repository we tested and the estimated number of fake gaps within an interval of 0.05 seconds. The left part of the graph shows the extreme values. A first observation includes the presence of fake gaps in almost all tested data files from the WIDE repository. Also note that not all test data from analysing Simpleweb was plotted, instead we plotted the 28 highest values. It can be observed from the raw test data, that there is an absence of fake gaps in 50.8% of the tested data files of the Simpleweb repository. Data files from location 2 are almost solely responsible for this. A possible explanation could be the low amount of traffic at this location.

Figure 3 plots the percentage of affected usable flows that have at least one fake gap. For the WIDE repository, on average 0.49% of the usable flows is affected by at least one fake gap. When ignoring data files without fake gaps, the average percentage of affected usable flows is 0.51%. Of the tested data files of the Simpleweb repository, an average of 0.27% of the usable flows in all data files was affected by at least one fake gap. When ignoring consistent files, the average is 0.55%. The highest value is in the file loc2-20030718-1530, where 23.72% of the flows is affected by at least one fake gap.

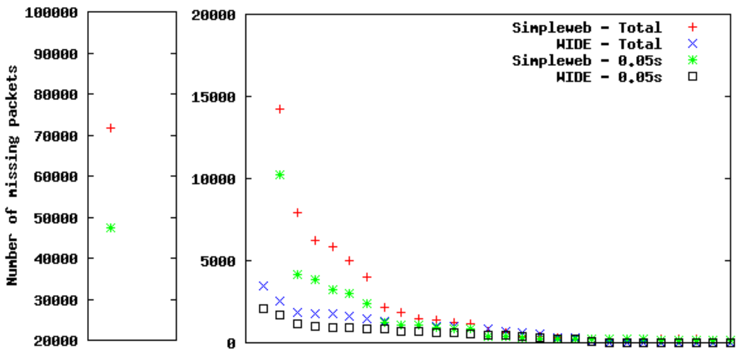


Fig. 2. Estimated number of missing packets and fake gaps

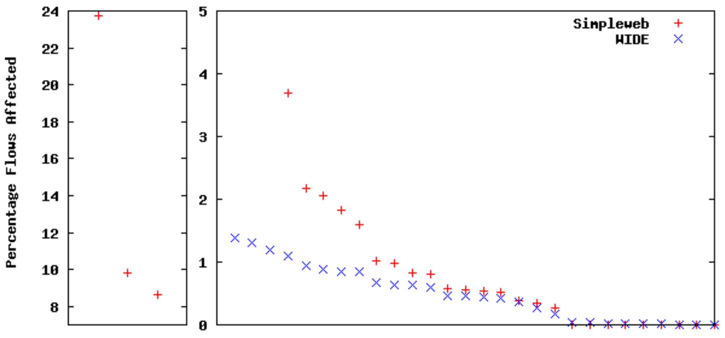


Fig. 3. Percentage of affected usable flows

4 Conclusions

This paper describes our research on the consistency of network traffic repositories. Before answering the main research question, we first look at two sub questions identified in Section 1.

The first sub question was *"How can we detect fake gaps?"*. We proposed an algorithm in Section 2, which extracts TCP flows. Then, it tries to identify fake gaps, packets that are not recorded by the recording device but were sent. The algorithm checks whether all data in the TCP flow is present by analysing TCP headers.

The last sub question, *"How consistent are today's repositories?"*, was answered in Section 3. We performed measurements on the Simpleweb [4] and WIDE [5] repositories. We showed both repositories contain inconsistencies. In the Simpleweb repository an average of 0.27% of the investigated TCP flows was affected by at least one fake gap. For the WIDE repository, this average was 0.49%. The research covered a substantial subset of data from both repositories.

We analysed 28 data files from the WIDE repository and 224 data files from the Simpleweb repository.

Going back to the main research question, *"How can inconsistency be detected in a TCP traffic repository?"*, we can now conclude detecting inconsistency is possible by using the proposed algorithm, which detects fake gaps. The knowledge that a repository is not always consistent is very important for research where it is critical to have all data recorded, like research on packet loss. For this kind of research, it is recommended to take possible inconsistency in the repository into account and, if no statistics are present, analyse the repository data before using it.

Future research could include extending the proposed algorithm to support TCP flows that are not completely present in the data file. This research can be used together with algorithms like the one described in [2], which checks for anomalies in traffic rate, to find the exact locations of fake gaps. This can, in turn, be used to draw conclusions about non-TCP traffic, thereby getting a better overview of the consistency of a network traffic repository.

References

1. Timmer, M.: How to identify the speed limiting factor of a TCP flow, `http://dacs.ewi.utwente.nl/assignments/completed/bachelor/reports/B-assignment_Timmer.pdf` (retrieved October 5, 2008)
2. Slomp, G.: Consistency of repositories. Presented at: 8th TSConIT, `http://referaat.cs.utwente.nl/new/paper.php?paperID=377` (retrieved October 5, 2008)
3. Lastdrager, E.E.H.: Consistency analysis of network traffic repositories. Presented at: 10th TSConIT, `http://referaat.cs.utwente.nl/new/paper.php?paperID=464` (retrieved February 20, 2009)
4. van de Meent, R., Pras, A.: Simpleweb/University of Twente – Traffic Measurement Data Repository, `http://traces.simpleweb.org` (retrieved October 5, 2008)
5. Cho, K., Mitsuya, K., Kato, A.: Traffic data repository at the WIDE project. In: Proc. USENIX Annual Technical Conference, p. 51 (2000)
6. Lastdrager, E.E.H.: Prototype and results, `http://www.vf.utwente.nl/~lastdragereeh/referaat`

A P2P-Based Approach to Cross-Domain Network and Service Management

Adriano Fiorese[1,2], Paulo Simões[1], and Fernando Boavida[1]

[1] Centre for Informatics and Systems of the University of Coimbra - CISUC
Department of Informatics Engineering - DEI
University of Coimbra - UC
fiorese@dei.uc.pt, psimoes@dei.uc.pt, boavida@dei.uc.pt
[2] Department of Computer Science - DCC
University of the State of Santa Catarina - UDESC
890233-100 Joinville, SC, Brazil
fiorese@joinville.udesc.br

Abstract. Cross-domain network and service management is still a challenging task. Most network management solutions that exist today are only applicable to single, homogeneous domains. Due to their distribution and technology-independence, P2P overlays have very good potential to boost inter-domain network and service management. The proposed work intends to explore this potential in order to develop efficient, trusted and secure cross-domain management, with benefits in terms of performance, resilience and support for autonomic operation. This paper explains the main challenges in this respect, discusses the basic approach to be taken and identifies the contributions.

Keywords: Network and Service Management, P2P, Overlays.

1 Problem Description

Efficient and effective solutions to network and services management are, more than ever, an unfulfilled promise. As networks and domains get larger and their inter-relations get more complex, the lack of solutions for global, end-to-end network management becomes apparent. On the other hand, users are requesting (and paying) services that require monitoring and control in order to guarantee a certain quality of experience, and do not care if the service spans several domains or requires complex low-level resource management.

The main obstacles for network management are administrative domain frontiers. Relatively good and stable solutions exist for intra-domain management, but they are not feasible at inter-domain level due to two main reasons: 1) administrative barriers imposed by domain owners; 2) several low-level management operations cannot be performed and/or are not efficient at this level (e.g., simple SNMP-based [1,2] solutions, or even Management by Delegation (MbD) [3] solutions).

In this respect, some researchers have begun to explore P2P-based systems for network management [4, 5, 6]. The potential of P2P overlays in the network management area is very good, as they lead to several immediate benefits: independence in relation to underlying technologies and providers; domain transparency; very good level of abstraction; and, last but not least, easy support of a user-oriented quality of experience view of the services.

The work proposed and presented in this paper addresses and explores the use of P2P-based overlays for network and service management in cross-domain environments.

2 Challenges

The main challenges of the proposed work are summarized in the following:

- Efficient cross-domain management;
- Provision of a trusted and secure solution;
- Distributed support for configuration operations;
- Improved performance when compared to traditional network management approaches;
- Enhanced resilience;
- Support for autonomic management, comprising intelligent communication, network control and management paradigms.

Current network management frameworks were developed for intra-domain environments, in which managing entities and managed entities trust each other and interact on a cooperation basis. In this context, all management information is available for managers, with various levels of detail. This is, of course, not the case when several administrative domains exist. In general, management information concerning one domain is hidden from entities in other domains, or is highly filtered. The use of a peer-to-peer overlay formed by nodes belonging to different administrative domains offers network managers the possibility of cooperatively managing these different domains. Management services provided by the overlay nodes are not bounded by domain borders, as these borders do not exist in the overlay. All overlay nodes cooperate, providing management services to each other.

One important challenge is that of trust and security. Appropriate mechanisms must be in place for ensuring that management operations performed in the overlay are carried out with the necessary assurance in terms of authentication and authorization. In addition, these operations must be translated into proper low-level inter-domain management operations when appropriate, filtering sensitive information and guaranteeing the necessary trust level.

Another challenge relates to configuration operations. Traditional solutions for network resources configuration are based on centralized configuration data repositories and it is part of a centralized approach. Large scale management configuration updates, however, can push servers and links to the limit. Additionally, centralized approaches lead to single points of failure and pose scalability problems. In this respect, P2P approaches open a range of possibilities that will be explored in the context of this work, including increased data configuration resilience and bandwidth savings.

Adequate performance of the network management operations is also a challenge. In the case of a centralized large scale management operation, the NMS may have to trigger a high number of messages which will cross the entire network domain and will reach managed elements one by one. A mix of management by delegation and P2P can achieve better performance. For instance, once a peer belonging to the management overlay receives a command for an update operation, it can perform the operation quicker, as it lies nearer to the network element than the central NMS.

Another motivation for the use of a P2P overlay for network management is the possibility of keeping network partitions managed in the event of a partitioning on the whole network due to failure of network elements or links. In a centralized network management approach, when a network partition occurs, just the partitions where the network managers reside remain managed. However when a self-organizing P2P overlay is used, the partitions themselves can elect nodes to take up management responsibility and keep things working.

Global inter-domain scope, trust and security, distributed operation, efficient load balancing and enhanced resilience are key aspects closely related to P2P systems that can be explored in order to pursue autonomic management, comprising intelligent communication, network control and management paradigms. The P2P approach can lead to smart network and service management entities that are capable of automatically configuring and organizing themselves.

3 Approach

Our approach consists of building a P2P-based network management framework to cope with the identified challenges. It will explore the cross-domain ability of P2P overlays to manage network resources and services. The proposed approach builds on and goes beyond of related work [4,5,6]. In our approach every peer in the network management overlay will be able to execute some management functions. These management functions/services will be published to network administrators, and will be made available to management tools. To do that, the peers in the overlay will advertise their services to one or more entities which will aggregate these advertisements. The component responsible for that is called Aggregation Service (see Fig. 1), and it will be composed of super-peers in the management overlay that will collect the management services advertised by the peers in the overlay. It is expected that this component can improve the search speed for specialized management services. These advertised management services will be looked up by high level managers in order to know which peers provide the services and which management interfaces can be used. Specialized management service providers can join the overlay in an non-deterministic way, offering their services. New management services can be deployed in the overlay in a natural way, just starting up the software.

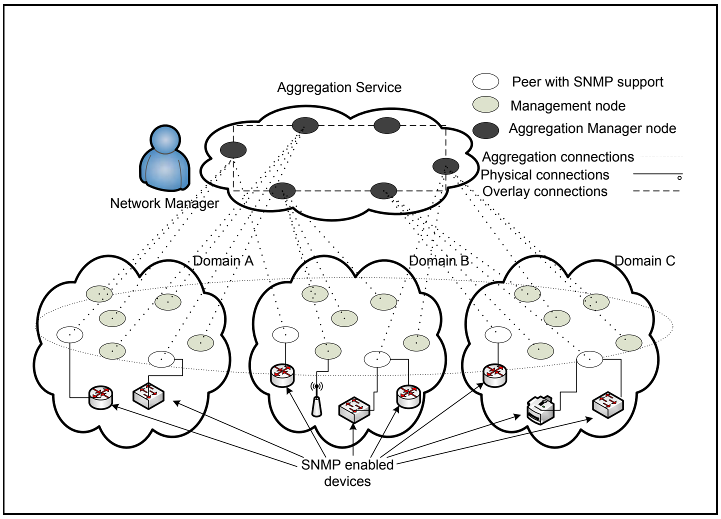


Fig. 1.

4 Expected Contributions

- The proposal, implementation and evaluation of a P2P-based network and service management system;
- A novel cross-domain management approach, adequately dealing with the trust and security issues that are the main obstacles to the deployment of management solutions spanning several administrative domains;
- An approach to distributed configuration operations, enhancing performance and resilience;
- The development of a set of mechanisms which can support autonomic management, in inter-domain environments.

References

1. Case, J.D., Fedor, M., Schoffstall, M., Davin, J.: Rfc1157 - simple network management protocol (SNMP) (1990)
2. Case, J.D., Mundy, R., Partain, D., Stewart, B.: Rfc3410 - introduction and applicability statements for internet standard management framework (2002)
3. Goldszmidt, G., Yemini, Y.: Distributed management by delegation. In: Proceedings of the 15th ICDCS, pp. 333–340 (1995)
4. Leitner, M., Leitner, P., Zach, M., Collins, S., Fahy, C.: Fault management based on peer-to-peer paradigms; a case study report from the celtic project madeira. In: 10th Integrated Network Management (IM), May 2007, pp. 697–700 (2007)
5. Granville, L.Z., da Rosa, D.M., Panisson, A., Melchiors, C., Almeida, M.J.B., Tarouco, L.M.R.: Managing computer networks using peer-to-peer technologies. IEEE Communications Magazine 43(10), 62–68 (2005)
6. State, R., Festor, O.: A management platform over a peer-to-peer service infrastructure. In: 10th International Conference on Telecommunications (ICT 2003), vol. 1, pp. 124–131 (2003)

PeerCollaboration

Thomas Bocek[1] and Burkhard Stiller[1,2]

[1] Department of Informatics IFI, University of Zurich, Switzerland
[2] Computer Engineering and Networks Laboratory TIK, ETH Zurich, Switzerland
{bocek,stiller}@ifi.uzh.ch

Abstract. Increasing traffic due to increased bandwidth or the number of users calls for scalable systems, which can be built with peer-to-peer (P2P) mechanisms. Scalability is a key issue for systems that rely on many participants, such as large-scale collaboration system. This paper introduces PeerCollaboration, a fully decentralized P2P collaboration system for documents, which is robust against malicious behavior, provides an efficient content search, and offer mechanisms for distributed control. Typical tasks in PeerCollaboration are searching, retrieving, creating, changing, and maintaining documents in a collaborative manner. Three problem areas are investigated within this system. The first problem focuses on similarity search on top of existing P2P networks and highlights a novel algorithm, which outperforms compared approaches. The second problem deals with incentive schemes, which work with indirect reciprocity. Thus, the novel and robust incentive scheme finds more reciprocities than with compared approaches. The third problem focuses on user-based voting mechanisms.

1 Introduction and Motivation

The Internet is steadily growing, because Internet users and Internet bandwidth of end-users are steadily increasing. According to [8], Internet network capacity grows 55% per year. This growth is also observed for end-users bandwidth. Nielsen's Law of Internet bandwidth states that high-end user's connection speed grows by 50% per year, which yields in a 57 times increase in 10 years [7].

This situation leads to the need for scalable systems, which scale with the number of participating users. Client-server systems have resource limits and as soon as this limit is reached, the system needs to be upgraded or replaced. Scalable P2P systems do not have such limitations, because every participating user contributes resources. Such P2P systems can reduce upgrading costs, increase stability, and improve fault-tolerance.

Large-scale centralized collaboration systems, such as Wikipedia, which depend on monetary donations for hardware infrastructure, cannot offer services, if donations decrease and infrastructure costs cannot be paid. Thus, the goal of PeerCollaboration is to demonstrate a scalable, robust, fault-tolerant, and fully decentralized P2P collaboration system for documents that provides robust incentives, an efficient content search, and offers mechanisms for distributed control. PeerCollaboration can be used for any kind of collaborative document writing, such as scientific reports, project deliverables, or online articles.

2 Decentralized P2P Collaboration System

In collaboration systems, users work together to reach a common goal. Users collaborate using communication and management tools. Examples for communication tools are Instant Messenger (IM), Email, or Voice over IP (VoIP) conferencing tools, and examples for management tools are calendar applications, project management tools, or social software. Decentralized collaboration systems work without central servers. Coordination among users is achieved in a P2P manner. Existing decentralized collaboration applications, such as Microsoft Groove, or P2P Wiki approaches PIKI [6], Wookie [10], DistriWiki [5], and [9] support users in a coordinated creation and sharing of documents. Table 1 compares distinct features of those P2P-based Wiki systems.

Table 1. P2P-based Wiki applications comparison

Application	*Distinct Features*
PIKI [6]	Concurrent editing, version control, full-text exact keyword search, semantic linking
Wookie [10]	Consistency and replication
DistriWiki [5]	No distinct features
Urdaneta et al. [9]	Fault tolerance, security issues, placement mechanisms
PeerCollaboration	Keyword similarity search, incentive schemes, user-based voting mechanism

The following use cases show the functionality, which are of particular interest for PeerCollaboration, of a general collaboration system. In use case 1, (cf. Figure 1), a user searches for the “Linux” article. The user misspells the search term and types “lniux” instead of “Linux”. PeerCollaboration searches for similar entries to “lniux”. The search can find misspellings up to the edit distance of 2. PeerCollaboration shows a preview of articles with a download link containing the keyword searched for. In use case 2, a user searches for an article about “Java”. The user types “java”. The result list including the preview does not contain this article. Thus, the user writes and submits the missing article. PeerCollaboration stores and indexes this articles after verifying that the user has contributed to the system beforehand. In use case 3 a user wants to search for an article about “Firefox”. One of the preview article shows the right article and the user downloads it. However, the user finds wrong information in this article. Thus, the user corrects and submits the article. This starts the voting process, where previous authors vote for or against the modification. Once the majority of previous authors agrees, the corrected article is published.

3 Problem Statement

Free-riders are peers that do not provide resources, *e.g.,* uploading documents. Such behavior limits scalability, because fewer peers have to provide these resources. Thus, incentive schemes are used to encourage peers to contribute. A

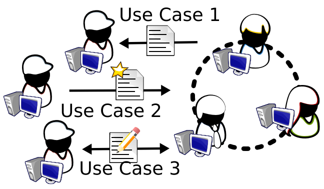


Fig. 1. Three use cases. Article search, publishing article, and modifying article

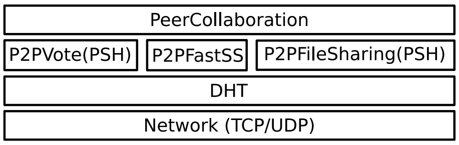


Fig. 2. Layered architecture with PeerCollaboration on top

very popular and widely used incentive scheme is Tit-for-tat (TFT). With TFT, users may only download as much as they upload. This incentive scheme keeps a per-peer history of resource transactions on every peer that is solely based on local observation. Thus, peers have a limited view of all transactions to peers with direct reciprocity. Peers with direct reciprocity can exchange resources from each other, while peers with indirect reciprocity can exchange resources only with at least one intermediate peer in between. However, with TFT indirect reciprocity is not detectable and exploitable. Thus, an incentive scheme that works with indirect reciprocity is essential in networks with many indirect connections.

Similarity search in P2P networks is important because otherwise peers that search for misspelled documents fail to find those. In flooding-based P2P systems, each peer can evaluate the search query and do a similarity lookup, however, these systems do not scale, because search queries are sent to all peers. Scalable P2P systems such as Distributed Hash Tables (DHT) allow for an exact search only. Thus, a fast similarity search algorithm on top of a DHT is essential to find documents with misspelled text.

Changing data in a P2P system is prone to attacks, because malicious peers can insert incorrect information or delete correct information, which leads to a lower quality of data. While quantitative parameters like text length and variation of words can be measured automatically, the qualitative assessment of human contributions can be performed only by humans. Thus, a voting mechanism with human interaction is essential to maintain the quality of data.

All three problem statements are addressed by PeerCollaboration, which discourages peers that download but never upload documents, enables similarity search to find documents with misspelled text or with a misspelled search query, and provides decentralized control for managing changes.

4 Design and Evaluation

The PeerCollaboration architecture can be split up into four layers as shown in Figure 2. The base layer determines the network layer, with Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) providing a socket interface to the DHT. Based on that, the DHT layer provides the interfaces `put()` and `get()` to the upper layers. On top of the DHT, P2PVote, P2PFastSS [2], and P2PFileSharing are built. The incentive scheme PSH [1] is built into P2PVote and P2PFileSharing. These three components offer the following interfaces. For

P2PFastSS the interfaces are `search` and `index`, for P2PFileSharing, `publish` and `download`, and for PeerVote, `proposeChange` and `vote`. On top of these three components, the PeerCollaboration application is built.

All three mechanisms and algorithms addressing the aforementioned problem statements have been evaluated. First, PSH [1] has been evaluated in a simulation and on EMANICSLab using input data from ThePirateBay. PSH has been compared to TFT, and two variants of PSH: PSH_r, and CompactPSH. Second, the underlying algorithm of P2PFastSS [2], FastSS and its variants have been evaluated in a simulation and on PlanetLab. FastSS has been compared to Burkhard-Keller tree, dynamic programming, NR-grep, keyword tree, neighborhood generation, and n-grams. [3]. Third, PeerVote [4] has been simulated on EMANICSLab evaluating malicious behavior, churn, and other parameters.

References

1. Bocek, T., El-khatib, Y., Hecht, F.V., Hausheer, D., Stiller, B.: Compact PSH: An Enhanced Private and Shared History-based Incentive Scheme (under review)
2. Bocek, T., Hunt, E., Hausheer, D., Stiller, B.: Fast Similarity Search in Peer-to-Peer Networks. In: 11th IEEE/IFIP Network Operations and Management Symposium (NOMS), Salvador, Brazil (April 2008)
3. Bocek, T., Hunt, E., Stiller, B.: Fast Similarity Search in Large Dictionaries. Technical Report ifi-2007.02, Department of Informatics, University of Zurich (April 2007)
4. Bocek, T., Peric, D., Hecht, F., Hausheer, D., Stiller, B.: PeerVote: A Decentralized Voting Mechanism for P2P Collaboration Systems. In: Proceedings of the 3nd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2009), Twente, The Netherlands (July 2009)
5. Morris, J.C., Lüer, C.: DistriWiki: A Distributed Peer-to-Peer Wiki. In: Proceedings of the 2007 International Symposium on Wikis (WikiSym 2007), Montreal, Quebec, Canada, October 2007, pp. 69–74 (2007)
6. Mukherjee, P., Leng, C., Schürr, A.: Piki - A Peer-to-Peer based Wiki Engine. In: Proceedings of the 2008 8th International Conference on Peer-to-Peer Computing (P2P 2008), Washington, DC, USA, September 2008, pp. 185–186 (2008)
7. Nielsen, J.: Nielsen's Law of Internet Bandwidth (2008), `http://www.useit.com/alertbox/980405.html`
8. Roberts, L.: Beyond Moore's Law: Internet Growth Trends. Computer 33(1), 117–119 (2000)
9. Urdaneta, G., Pierre, G., van Steen, M.: A Decentralized Wiki Engine for Collaborative Wikipedia Hosting. In: Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST), Barcelona, Spain (March 2007)
10. Weiss, S., Urso, P., Molli, P.: Wooki: A P2P Wiki-Based Collaborative Writing Tool. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 503–512. Springer, Heidelberg (2007)

Fast Learning Neural Network Intrusion Detection System

Robert Koch and Gabi Dreo

Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
{Robert.Koch,Gabi.Dreo}@UniBw.de

Abstract. Assuring the security of networks is an increasingly challenging task. The number of online services and migration of traditional services like stocktrading and online payments to the Internet is still rising. On the other side, criminals are attracted by the values of business data, money transfers, etc. Therefore, safeguarding the network infrastructure is essential. As *Intrusion Detection Systems (IDS)* had been in the focus of a numerous of researches for the last years, several sophisticated solutions had been found. Very capable IDS are based on neural networks. However, these systems lack of an adaptability to dynamic changing environments or require a protracted learning phase before they are operational. The approach is to overcome these restrictions by introducing a modular neural network based on pre-processed components supplemented by static policies. By that, it is possible to overcome long-lasting learning phases.

1 Introduction

The availability of services, communication lines and the network infrastructure is of essential relevance. For that, failures, attacks and other kinds of anomalies have to be detected as soon as possible. Beside firewalls, IDS are key elements for securing network infrastructures nowadays. Traditional IDS analyze the network traffic for the appearance of known patterns. Either only the headers of the packets are analyzed or also the payload, whereas analyzing the whole payload is not always possible due to performance reasons. Furthermore, pattern-based analysis is hardly able to detect new and unknown attacks.

Newer approaches examine the network infrastructure as a whole. *Network Security Situational Awareness (NSSA)* considers all information about the network, not only intrusions. The security engineer is aware of what is happening on the network [1]. In *Network Behavior Analyses (NBA)*, anomalies are detected by evaluation of the network traffic statistics. Newer routers and switches are able to generate and allocate statistics about the traffic flow through the device. Important protocols are *NetFlow* (RFC 3954) from Cisco which is now under further vendor-independable development known as *Internet Protocol Flow Information Export* (RFC 3917) on the one side, and sFlow (RFC 3176) on the other. By using flow information it is possible to detect unknown and new threats because of the divergence of the statistics to the normal network behavior in the case of an attack. Very efficient systems for the evaluation of the data are based on neural networks. Due to their ability to find behavioral patterns in the attack which could be learned [2], detection of yet unknown attacks is possible. *Modular Artificial Neural Networks (MANNs)* and hierarchical ones are better adaptable to dynamic environments. Modular networks divide the task into multiple subproblems while in a hierarchical network every node can be a neural network on its own. By combining NBA techniques with neural networks it is possible to perform online-learning and to recognize new and unknown threats.

Anyway, there are still major shortcomings: Depending on the respective realization, neural networks suffer from long lasting learning phases. Things are getting worse if the learning phase is carried out in a manipulated, non-isolated environment, because that can force the network to accept attack behavior in the final system. After finishing the learning phase, they are only able to adapt to changing environments by introducing special structures and methods. Therefore, neural networks with fast and secure learning phases are needed.

An overview of current researches will be given in the next paragraph, while section 3 describes our proposed approach.

2 Current Research

There are numerous studies in the area of IDS. Simple pattern-matching based IDSs are very efficent for detecting malicious traffic, but only known and available patterns could be found. Therefore, IDSs based on neural networks have been investigated which are able to detect new and unknown attacks. Anyway, after the network is trained, adaptability to changing environments is hard to accomplish. Possibilities to overcome these restrictions are enabled by the concepts of modular and hierarchical systems. In [3], Berg and Spaanenburg show that compositionality of MANNs is possible under further conditions, parallel and cascade (de-)composition are shown. Zhang et al. use hierarchical networks to construct a modular IDS [4]. The structure of the system consists of multiple layers and classifiers. First, a serial structure is used, later on some of the shortcomings of the serial structure are solved by using a parallel structure. Basically, an anomaly classifier detects either the packet is "normal" or not. Suspect packets are stored in a database and structured by a clustering algorithm. After exceeding a threshold value, a new classifier is trained.

For building neural networks with faster learning abilities, Moraga [5] shows that by including knowledge of the problem in the design phase, the effectiveness of the network can be increased. Doing that, he was able to construct the neural network without the need for a learning phase. Almgren et al. [6] reduce the learning data by allowing the algorithm to decide which data should be used next. By addressing the dynamics of the environment, some appendages for realizing online-learning capabilities in neural networks are done. Potter proposed a *Learning Intrusion Detection System* based on a blackboard architecture [2]. A learning layer is introduced as an additional layer in a multi-tier concept, which is able to maintain and update the training data for the network. By using multiple agents it should be more efficient in a dynamic environment.

NBA and NSSA are concepts to examine the current state of the whole network. Rehak et al. present an approach for NBA to reduce the error rate and the number of false positives. This is done by designing a framework for the integration of different anomaly detection algorithms. By this a value for the trustfullness of the current traffic is generated. *Extraction Method of Situational Factors for NSSA* [1] describes methods for the evaluation of *situational factors*, for example a special type of intrusion.

Even there exist multiple implementations, there are still major drawbacks. Neural networks have to be trained for the operational environment. Furthermore, attacks from the inside of the local network, for example by manipulating a switch or performing a replay attack, are hardly to detect and can undercut the integrity of a neural network based system in the learning phase.

3 The Idea: Fast Learning Neural Network IDS

To overcome the long learning phases and the resulting threat of learning attack behavior as valid behavior, we use a MANN. Fig. 1 gives an overview of the system. The traffic as well as the flows which are to be analyzed will be processed by the MANN. Further on, the data available by routers, switches and network scans will be used to implement policy-based rules about the network structure.

Unlike existing approaches, the initial MANN will be constructed by the usage of pre-processed modules. These modules are devided into two groups: service- and infrastructure-oriented. The former ones are designed and trained to detect anomalies in the network traffic of different services, like the hypertext transfer protocol, by learning the protocol-specific communication behavior in a secured environment. The later ones are used to detect changes in the network infrastructure like connecting new devices. In the inital network, only a few infrastructure-oriented pre-processed modules could be used, because for these modules, a generalization is not possible.

To overcome the lack of knowledge in the beginning, static data is used to monitor the network structure. For example, this could be the switch and port a host is connected to. By the use of policies, the allowed and restricted possibilities to integrate new devices and services into the network can be specified. While these are strict guidelines for the early operation of the IDS, it allows to detect the endangerment of learning forbidden network operations. If a violation of a policy is detected by the system, it informs the operator who can decide to stop or to continue the learning process of the neural network. After the learning phase of the infrastructure-oriented modules, a finer classification is available, for example by using typical load distribution, response times, etc.

Normally, the NBA is not able to detect slowly driven attacks, because of the necessity to excess a threshold to raise an alarm. The *Infrastructure and Service Policies Module* can also be used for the detection of slowly driven attacks, if they are violating the given policies. Therefore, several violations of policies are

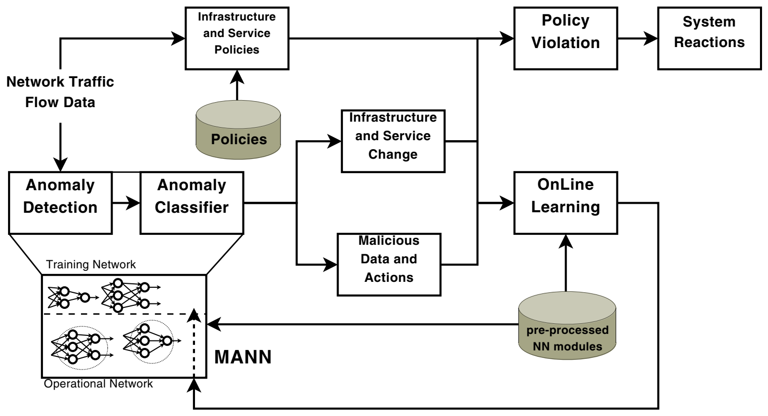


Fig. 1. Overview of the System Structure. Pre-processed modules are used to build the inital operational MANN. Additional MANN components are trained before transferred to the operational network. The policy module defines additional strict rules.

secured in a long term matrix and evaluated by using a time lapse function. Detected anomalies are forwarded to the MANN for further investigations.

Summarized, the IDS will consist of an initial neural network and an additional component for monitoring static policies. The initial system is generated by composition of the selected pre-processed modules. In the beginning these are the modules that are responsible for the services in the network, while the infrastructure is described by the policies set. Only the training network is modified during the learning phase and transferred to the operational network to discrete points in time. providing the possibility to execute a roll-back of the training network. Hereby, the initial learning effort and furthermore the endangerment of a manipulation of the learning phase is reduced to a minimum.

References

1. Wang, H., Liang, Y., Ye, H.: An Extraction Method Of Situational Factors For Network Security Situational Awareness. In: International Conference on Internet Computing in Science and Engineering (2008)
2. Dass, M., Cannady, J., Potter, W.: A Blackboard-Based Learning Intrusion Detection System: A New Approach. In: Chung, P.W.H., Hinde, C.J., Ali, M. (eds.) IEA/AIE 2003. LNCS (LNAI), vol. 2718, pp. 385–390. Springer, Heidelberg (2003)
3. Berg, A., Spaanenburg, L.: Modular and Hierarchical Specialization in Neural Networks. Journal of Electrical and Electronics Engineering 3(1) (2003)
4. Jiang, J., Zhang, C., Kamel, M.: Intrusion detection using hierarchical neural networks. Pattern Recognition Letters 26 (2005)
5. Moraga, C.: Design of Neural Networks. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS, vol. 4692, pp. 26–33. Springer, Heidelberg (2007)
6. Almgren, M., Jonsson, E.: Using Active Learning in Intrusion Detection. In: Proceedings of the 17th IEEE Computer Security Foundations Workshop, CSFW 2004 (2004)

Design of an Autonomous Framework for Efficient Large Scale Management of Next Generation Web Service Mashups

Anna Hristoskova and Filip De Turck

Department of Information Technology, Ghent University - IBBT,
Gaston Crommenlaan 8 bus 201, B-9050 Ghent, Belgium
{anna.hristoskova,filip.deturck}@intec.ugent.be

Abstract. The objective of this PhD research is the study and development of an autonomous framework focusing on the dynamic composition and deployment of service mashups. By enriching the available building blocks with semantic descriptions, automatic composition of new service mashups is achieved through the use of planning algorithms. The composed mashups will be automatically deployed on the available resources making optimal use of bandwidth, storage and computing power of the network and server elements. This system will also be extended with dynamic recovery from external errors, such as network failure or services delivering erroneous results. The last step will encompass the automatic extraction of new knowledge from request patterns, behavior and feedback of users for the optimization of future requests.

A number of relevant case studies will be defined in the domain of eHealth and Web 3.0, since the use of workflows for efficient data processing in these areas is very important.

1 Introduction

A major trend in the field of telecommunication research is the need for scalable processing of data. The main causes are the deployment of sensor networks and the growing importance of context-aware applications. Sensor networks consist of large quantities of sensors, placed in a building at regular distances on the walls, for monitoring temperature, the presence of persons or goods. Context-aware applications cover software components interacting with each other, so that the application behavior reacts on the current context, the user's preferences and the state of the network to which the user is connected.

An important aspect of the scalable processing of data is the optimal structuring of the available information. In recent years the Semantic Web [1,2] is utilized to overcome this issue offering more access not only to content but also to Web services. The use of Web resources based on keywords and the provided service content is feasible [3]. This allows for easy sharing of reusable knowledge. In addition, ontologies not only facilitate the communication between people and machines, but also between machines. Since the information is presented in a formal way, a computer can reason about it and new knowledge can be inferred. In this way the wide range of components used in sensor networks and context-aware applications can be dynamically and more efficiently combined through semantic reasoning, accomplishing new service mashups.

A service mashup is a new service that combines functionality or content from existing sources. These sources can be Web services, software components capable of being accessed via standard network protocols such as but not limited to SOAP over HTTP. The mashup construction is realized due to a workflow description: definition of how the basic building blocks interact. Today there are a number of popular standards and implementations, such as BPEL [4], BPMN, WFF, which define workflows. However they still exhibit a number of shortcomings: no automatic or dynamic deployment support, limited reliability guarantees, etc. The main goal of this PhD research is to study in depth these topics and design a framework focusing on the following aspects:

- Reasoning algorithms for automatic composition of service mashups satisfying quality of service constraints and requirements.
- Deployment strategies of the service mashups on the available resources making optimal use of bandwidth, storage and computing power of the network and server elements, extended with dynamic recovery from external errors.
- Optimization of future request through the automatic extraction of new knowledge from request patterns, behavior and feedback of users.

2 Autonomous Semantic Composition

Initially, our research will focus on the development of reasoning algorithms for the automatic composition of new service mashups [5]. For this purpose the Semantic Web will be studied that enables users and software agents to automatically discover, compose, invoke, and monitor Web resources offering services, subject to specified constraints. Semantic languages such as OWL-S, WSMO, SWSO, provide a semantic description of Web services specifying service inputs, outputs, preconditions and effects, and non-functional properties. Thanks to their semantic description, Web services can be compared and matched accomplishing a mashup composition, as shown in figure 1, realizing user defined goals. As there is no need for user intervention, this process can be fully automated.

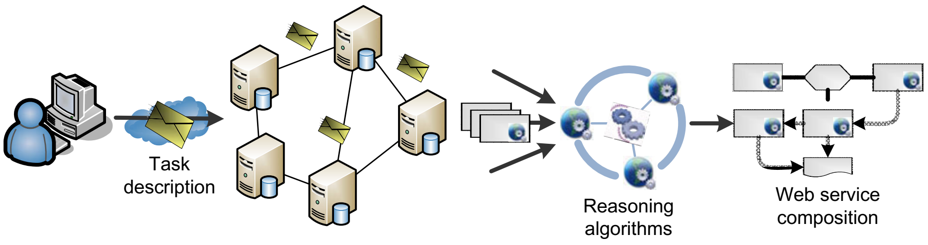


Fig. 1. Automatic composition of Web services

A number of reasoning algorithms are already developed that take into account certain quality constraints (execution time, cost) of the available building blocks used in the Intensive Care Unit of Ghent University Hospital. Semantically enriched medical services are automatically combined accomplishing a medical decision support system monitoring the patient's condition and registering critical changes in his state. This approach is still being refined with more comprehensive HTN planning techniques.

3 Resource Efficient Service Deployment

Figure 2 illustrates the autonomous deployment of abstract workflows on a set of available resources [6]. This part of the research requires the linking of abstract service definitions to concrete service instances, adaptation in the communications layer, selection of the appropriate workflow engines and possible creation of additional workflow code. Optimization algorithms will be developed to ensure translation that makes the best use of the available network infrastructure.
These algorithms will be further extended to support dynamic deployment of the workflows [7]. This is necessary to avoid down-time but it may be also crucial to modify the deployment so as to anticipate changes in the network situation [8]. Triggers can be a change in the network topology, or an alteration in the nature of the workflow distribution.

As a last step, the design of a framework will be studied for the development of large-scale context-aware applications. We will develop procedures for automatically extracting new knowledge from request patterns, behavior and feedback of users. This knowledge will be utilized for the optimization of the composition and deployment process for future requests.

We will closely look into a Web 3.0 application addressing the enrichment of business components and services (such as sensors) with semantics, reasoning, autonomous behavior (agent technology), and distributed data management. We will proceed with the selection and composition of these building blocks into an application and providing assurance about the performance of the composite services on the available resources. The most suitable composition and deployment techniques for this domain will be examined.

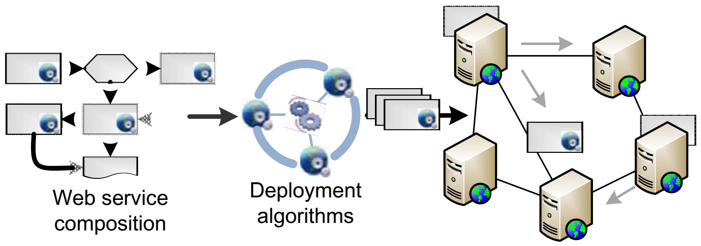


Fig. 2. Deployment of a service mashup

4 Summary

This PhD research focuses on the study of an autonomous framework for dynamic composition and deployment of the building blocks of service mashups. Based on semantic descriptions of Web services and resources, reasoning algorithms are developed for automatically composing new service mashups realizing defined goals. In addition, deployment strategies are designed for the distributed deployment and execution of the service mashups on the available resources. This system is optimized for the dynamic response to changing context such as failure or overload of network elements or Web services. Furthermore, techniques will be studied to take into account trends in user and resource behavior, in order to optimally design context-aware service mashups. Important application areas such as eHealth and Web 3.0 will be implemented to verify this framework.

Acknowledgement

A. Hristoskova would like to thank the Special Research Fund of Ghent University (BOF) for financial support through her PhD grant. F. De Turck is funded by a postdoctoral grant of the Fund for Scientific Research, Flanders (FWO-V).

References

1. Berners, L., Hendler, J., Lassila, O.: The Semantic Web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Journal of the Scientific American 284(5), 34–43 (2001)
2. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 289–303. Springer, Heidelberg (2004)
3. Keeney, J., Lewis, D., O'Sullivan, D.: Ontological semantics for distributing contextual knowledge in highly distributed autonomic systems. Journal of Network and Systems Management 15(1), 75–86 (2007)
4. Karastoyanova, D., Houspanossian, A., Cilia, M., Leymann, F., Buchmann, A.: Extending BPEL for run time adaptability. In: 2005 Ninth IEEE International EDOC Enterprise Computing Conference, September 19-23, pp. 15–26 (2005)
5. Klie, T., Gebhard, F., Fischer, S.: Towards automatic composition of network management web services. Integrated Network Management, 769–772 (2007)
6. Wang, C., Zou, Z., Ma, Q., Fang, R., Wang, H.: A comprehensive semantic-based resource allocation framework for workflow management systems. In: IEEE Network Operations and Management Symposium (NOMS 2008), pp. 831–834 (2008)
7. Lee, M., Yoon, H., Shin, H.: Supporting dynamic workflows in a ubiquitous environment. In: International Conference on Multimedia and Ubiquitous Engineering, MUE 2007, April 2007, pp. 272–277 (2007)
8. Gajewski, M., Momotko, M., Meyer, H., Schuschel, H., Weske, M.: Dynamic failure recovery of generated workflows. In: Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA 2005) (2005)

Scalable Detection and Isolation of Phishing

Giovane C.M. Moura and Aiko Pras

University of Twente
Design and Analysis of Communication Systems (DACS)
Enschede, The Netherlands
{g.c.m.moura,a.pras}@utwente.nl

Abstract. This paper presents a proposal for scalable detection and isolation of phishing. The main ideas are to move the protection from end users towards the network provider and to employ the novel bad neighbourhood concept, in order to detect and isolate both phishing e-mail senders and phishing web servers. In addition, we propose to develop a self-management architecture that enables ISPs to protect their users against phishing attacks, and explain how this architecture could be evaluated. This proposal is the result of half a year of research work at the University of Twente (UT), and it is aimed at a Ph.D. thesis in 2012.

1 Introduction

With the rapid growth of services like e-business and electronic banking, *phishing* is quickly becoming one of the main security threats for the Internet. In recent years we have witnessed a shift from incidents, towards a well organized criminal phishing "industry". Despite of the fact that the real direct losses (money stolen from users) caused by phishing is still unknown – since usually banks and companies do not disclose this information – some estimates indicate that these losses could, in the United States, be between \$ 61 million [1] and \$ 3.2 billion [2] per year. In addition, phishing also inflicts indirect costs (*e.g.*, costs associated to the staff to deal with it, network and computer resources) and compromises the trust of Internet users in e-mail and the web [1].

In order to fight phishing, several approaches have been proposed in the last few years. These approaches can be categorized into two major areas: (i) the "human factor" and (ii) techniques and algorithms to detect phishing. The former comprises social and psychological studies, the design of user interfaces, and the education of users and providers [3,4]. The latter presents different approaches to protect the user against phishing. Since phishing attacks usually consist of two phases (distribution of malicious URLs via spam, and connecting to malicious URLs by victims), there are proposals that focus specifically on detecting phishing e-mail [5] and others that focus on blocking the access to spoofed URLs. In relation to these, most of the approaches are client-centric, i.e., they require software installed at the client side (*e.g.*, a browser extension module [6,7,8]) and often rely on Content Black Lists (CBLs) for finding suspicious words. Commercial solutions also exist; the Cyveillance Anti-phishing

[9] has developed a proprietary Internet monitoring technology that claims to identify activities associated to phishing, such as suspicious domain registrations and spoofed web sites.

Despite all these efforts, a Gartner survey shows that phishing attacks numbers are, in fact, increasing [2]. We consider that this is mainly because current approaches present limitations such as: (i) Internet Service Providers (ISPs) are not directly involved (which means that anti-phishing approaches rely on end users no longer using outdated software), (ii) lack of scalability (detection of phishing messages puts a heavy load on mail servers), (iii) lack of automation (maintenance of blacklists is cumbersome), and (iv) non-persistent IP addresses (current approaches have problems dealing with the dynamics of botnets). In this paper we introduce an approach to address these limitations. The reminder of this paper is organized as follows: in Section 2 the approach is explained, in Section 3 the envisioned architecture is described – and how it can be evaluated – and conclusions are presented in Section 4.

2 Approach

To address the limitations of the current proposals, we present an approach that provides a scalable and automated framework to protect users against phishing attacks. The main ideas behind this approach are:

Move protection to the network: Instead of relying on end users, we propose to move protection towards the ISP, who should block malicious servers (by updating rules at firewalls and routers) and, in this way, offer transparent protection for *all* end users. ISPs could offer this as a service, in the same way as British Telecom's Managed Security Services are offered to enterprises [10].

Analysis of flows: To ensure scalability, we propose the use of flow analysis instead of deep packet inspection. Moreover, with flow based approaches it is possible to analyze flow *patterns* and compare the network behaviour of multiple sources.

Self-management: The high number of phishing attempts makes it impossible to manually cope with such attempts. Complete automation is desired to make detection and isolation fast and reliable. The idea is to develop an integrated architecture that is able to detect and block phishing. More details about the architecture are described in Section 3.

Bad neighbourhood concept: To cope with botnets and non-persistent IP addresses, we propose the novel bad neighbourhood concept. The idea behind bad neighbourhoods, is that the likelihood a certain IP address behaves badly, increases if neighbour IP addresses (systems within the same subnetwork) behave badly. The assumption is that we can distinguish between well managed subnetworks, in which the probability of misbehaviour is very small, and badly managed networks, in which the probability of misbehaviour is quite high. Bots will primarily be found in the later kind of networks. For example, Figure 1

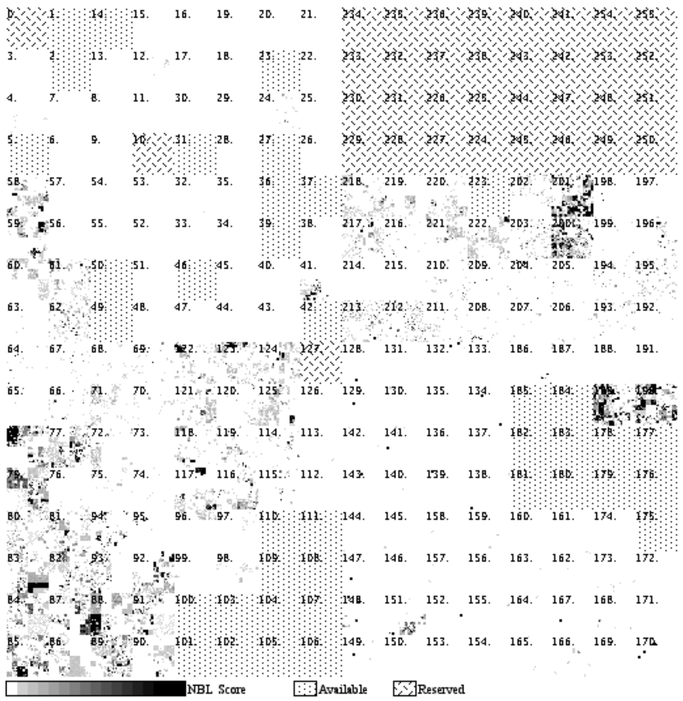


Fig. 1. Visualization of bad neighborhoods for spam senders (2008-11-01)

clearly shows that spam senders present this pattern, highlighting which subnetworks generate most spam. This picture was developed using different network blacklists. An algorithm took the blacklists data as input, and generated as output a Network Black List (NBL) score. Each point in the figure represents the NBL score for a subnet level of /24, using a Hilbert space filling curve [11]. We believe that many phishing e-mail senders and phishing web servers can also be found in bad neighbourhoods. This would provide us with statistical guidelines on the trustworthiness of neighbours (same subnetwork) IP addresses, which could be used as input to anomaly detection techniques in order to mitigate phishing attempts.

3 Architecture and Evaluation

To prove the concept and technical feasibility of our proposal, we will develop a self-management architecture for phishing detection and isolation. This architecture includes several elements, namely: e-mail servers, network routers, network firewalls, telescopes (specific kinds of honeypots, distributed all over the network, to capture and detect phishing messages), phishing e-mail senders, phishing web servers, and a phishing detector.

The architecture will be evaluated by building prototypes of key components. Data from real networks (like Géant, Surfnet and/or UT) will be used, such as network flows and telescopes logs. Next, the idea is to analyze the data using self-learning techniques and the bad neighbourhood concept. This will result into two different lists. The first one contains IP addresses that distribute phishing messages, while the second contains names and IP addresses of phishing web servers. The first list could be used as input to mail servers for detection and removal of phishing messages. The second list could be used as input for network routers and firewalls, to block users from malicious web servers. In is important that both lists are maintained and employed in an automated way.

4 Conclusions

Phishing is a important problem that, despite all efforts, still causes significant monetary losses. In this research we propose to develop and evaluate an architecture to detect and isolate machines associated to phishing activities. A novel element in this architecture is the bad neighbourhood concept, of which the merits will be further investigated as part of this Ph.D. research. This paper was supported by the EC IST-EMANICS Network of Excellence (#26854). We would like to thank Ward van Wanrooij for his ideas on the bad neighbourhood concept and for providing Figure 1.

References

1. Herley, C., Florencio, D.: A profitless endeavor: Phishing as tragedy of the commons. In: Proc. of the ACM SIGSAC New Security Paradigms Workshop, Lake Tahoe, California, USA (September 2008)
2. McCall, T.: Gartner survey shows phishing attacks escalated in 2007 (2008)
3. Dhamija, R., Tygar, J.D., Hearst, M.: Why phishing works. In: CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 581–590. ACM, New York (2006)
4. Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L.F., Hong, J.: Lessons from a real world evaluation of anti-phishing training. In: eCrime Researchers Summit, pp. 1–12 (2008)
5. Fette, I., Sadeh, N., Tomasic, A.: Learning to detect phishing emails. In: WWW 2007: Proceedings of the 16th international conference on World Wide Web, pp. 649–656. ACM, New York (2007)
6. Zhang, Y., Egelman, S., Cranor, L.F., Hong, J.: Phinding phish: Evaluating anti-phishing tools. In: Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS 2007), San Diego, CA, USA (2007)
7. Dhamija, R., Tygar, J.D.: The battle against phishing: Dynamic security skins. In: SOUPS 2005: Proceedings of the 2005 symposium on Usable privacy and security, pp. 77–88. ACM, New York (2005)
8. Phishing Protection Design Documentation (2009), `https://wiki.mozilla.org/safe_browsing:_design_documentation`
9. Cyveillance Anti-Phishing (2009), `http://www.cyveillance.com/`
10. British Telecom Managed Security Services (2009), `http://bt.counterpane.com/managed-security-services.html`
11. Irwin, B., Pilkington, N.: High level internet scale traffic visualization using hilbert curve mapping. In: VizSEC 2007: Proceedings of the Workshop on Visualization for Computer Security, pp. 147–158. Springer, Heidelberg (2007)

Multi-agent Reinforcement Learning in Network Management

Ricardo Bagnasco and Joan Serrat

Network Management Group,
Departament de Teoria del Senyal i Comunicacions,
Universitat Politècnica de Catalunya
{rbagnasco,serrat}@tsc.upc.edu

Abstract. This paper outlines research in progress intended to contribute to the autonomous management of networks, allowing policies to be dynamically adjusted and aligned to application directives according to the available resources. Many existing management approaches require static *a priori* policy deployment but our proposal goes one step further modifying initially deployed policies by learning from the system behaviour. We use a hierarchical policy model to show the connection of high level goals with network level configurations. We also intend to solve two important and mostly forgotten issues: the system has multiple goals some of them contradictory and we will show how to overcome it; and, some current works optimize one network element but being unaware of other participants; instead, our proposed scheme takes into account various social behaviours, such as cooperation and competition among different elements.

Keywords: Machine learning, policy-based management, adaptive policies, autonomic communication system.

1 Introduction

The growth of Internet and particularly the rapid advances in real time supported applications that are expected to be developed, make its management a major challenge. Among the different enabling technologies for autonomic communications, the policy based management is one of the most representatives. This paradigm allows the segregating of the rules that govern the behaviour of the managed system from the functionality provided by the system itself [1]. The most developed and contemporary implementations of the paradigm rely on pre-programmed rules based on logic. Nevertheless, to be called autonomic, a system must show a degree of flexibility to self adapt to changes in the goals, services or resources and in our opinion this is hardly achievable by means of static policies. In contrast, we propose the control of the system behaviour by means of dynamic policies; that is, policies that are allowed to change according to the evolution of the system. In short, we pretend to design a method that dynamically and continuously seeks for management policies that better fit the context of execution looking for its optimization. This is innovative because current approaches lack of learning capabilities and most of the recent work just mention the need of it [2], [3].

With this conception of the problem, the first challenge is to figure out the learning mechanism or mechanisms that will lead to new policies. The second, but no less important is how to solve the potential conflicts that can arise from such a policy generation process. Finally, it would be important to understand the whole vertical structure from devices till applications in order to combine and enforce the best configuration. All these challenges will be addressed in this work in the following manner: for the first two issues we plan to use a model free learning algorithm and because of the size of the network we will investigate multi-agent variations, especially those that acknowledge the existence of other competitive/cooperative agents in the same environment. For understanding the whole structure of policies we will define and use a policy architecture from the *Application level* to the *Device level* in a similar approach as the Policy Continuum [4] showing the relationships between policies of different levels.

The remainder of the paper is structured as follows: section 2 provides an overview of work related to our topic of interest. In section 3 we introduce our idea and show our planned topics of research. Section 4 finishes this article with the overall conclusions.

2 Related Work

Since managing a complete network involves several topics, we could find related work in a wide area of research. We will focus in communication systems but, for instance in the multi-hardware configuration [5] or sensor networks [6] we can find analogous problems too. And because we will use learning techniques we could also find related work (in an abstract way) in the machine learning field [2].

There are a few examples of approaches for a complete solution of learning in network management. A work that could be considered as a starting reference is described in [7]. It is essentially focussed in policy refinement but, from the point of view of what we pretend to do, it doesn't consider dynamic adaptation of policies in the presence of dynamic environments. Another approach to introduce an architecture oriented to an adaptable service is in [8]. Here the authors use rule-based reasoning and extended finite state machines. Their mechanism to select the rule is using a *Reasoning Procedure*. One disadvantage of their solution is the need of a careful specification of all possible events. A different approach can be found in [9], where they propose some similar goals as in our work (adaptation and flexibility) and also use a policy hierarchy. But they skip the multi-goal issue and they just sketch some framework but don't explain the details of their mechanisms such as conflict resolution because of the coexistence of other agents. In contrast, although we also pretend to follow an adaptation process based on learning, our target is a more flexible way of adaptation of policies to changes in goals and the environment with embedded policy conflict avoidance.

3 Planned Approach

We consider a continuum of policies [4] constituted by several layers; in the simplest form by only two levels. The lowest level corresponds to *device policies* managing the physical or virtual resources. At this level we have some configurations that the device can offer to the applications so the higher level can choose between those pre-configurations. The second level contains *application policies*. We understand that applications use the resources of device level. For example we could have a printer with three configurations: *Draft*, *Black_and_White*, *Full_Color*; and different applications could prefer one setting over another. Or for instance some links could have two metrics: *speed of transmission* and *error rate*, and offer five pre-configurations with different values of speed and error rate, having a different impact in the performance of the applications so it is needed to find the best option in general.

Because the algorithm we plan to use could be quite slow to converge it would be practical to initialize the policies at device level with some initial state (for instance with the output of some simulation of the situation). After this initialization the policies will be evolved online (in the real world) by means of Reinforcement Learning [10], a sub area of Machine Learning, concerned with how an agent should take actions in an environment to maximize some long-term reward. In particular we are interested in the Temporal Difference techniques and Q-Learning [11] more specifically. This technique learns an action-value function to estimate the expected utility of taking some action in a given state. Applications should give some feedback (reward) to its devices in order to inform about how good or bad its performance is to their own goals. Taking this information into account the lower level will know which action (pre-configuration over its metrics) is the best for the system at each moment. The changes in environment, devices, and applications will affect the performance making our previous preferred pre-configuration probably no longer optimal so our algorithm will notice that and adjust it to a new selection that is the best for that moment and situation. The ultimate goal will be to maximize the reward of the system, for example evaluating a weighted sum of the application's rewards, and it will be very important to pay special attention to the carefully design of that function. In addition, because it is not feasible to consider the whole system as a single agent, we plan to use multi agent systems and study several distributed algorithms in order to better solve our problem. We plan to validate our proposal using the AutoI infrastructure [12] in one of its general use cases and, eventually, using the OPNET simulator [13] extending it to use a policy-based management system. But first we should make a proof of concept showing that changes in environment cause changes in politics and then we should measure time of convergence to stable politics in several kinds of environments evaluating different algorithms.

4 Concluding Remarks

This paper has described our ongoing research work towards managing a communications system in a flexible and adaptable way and configuring the devices aligned to user and application goals. We tackle most of the open issues identified in the most recent papers and surveys [2]. Specifically the following challenges are considered: from supervised to autonomous learning; from offline to online learning; from fixed to changing environments and from centralized to distributed learning. Our immediate future work will be the definition of a scenario in which we demonstrate the feasibility of our approach and the investigation of several algorithms to choose the best suited one.

We strongly believe that, because of network complexity, learning policies is key in network management in contrast to knowledge based approaches. With this regard, our contribution is to the best of our knowledge, the first proposal to learn, adapt and align policies from the devices configuration till the application taking into account the presence of other entities competing for resources and having their own goals.

Acknowledgments. This research is supported in part by the "Programa de FPI del Ministerio de Innovación y Ciencia de España". This is also part of the AutoI project and of the EMANICS NoE.

References

1. Boutaba, R., Xiao, J.: Network Management: State of the Art". Communication Systems: The State of The Art (IFIP World Computer Congress), pp. 127–146 (2002)
2. Dietterich, T.G., Langley, P.: Cognitive Networks: Towards Self-Aware Networks. In: Machine Learning for Cognitive Networks: Technology Assessment and Research Challenges, ch. 5 (2007)
3. Badr, N., Taleb-Bendiab, A., Reilly, D.: Policy-Based Autonomic Control Service. In: Fifth IEEE International Workshop on Policies for Distributed Systems and Networks 2004 (2004)
4. Davy, S., Jennings, B., Strassner, J.: The policy continuum-Policy authoring and conflict analysis. Computer Communications 31(13), 2981–2995 (2008)
5. Wildstrom, J., Stone, P., Witchel, E., Mooney, R.J., Dahlin, M.: Towards Self-Configuring Hardware for Distributed Computer Systems. In: Proceeding of the Second International Conference on Autonomic Computing, ICAC 2005 (2005)
6. Akyildiz, I.F., Weilian, S., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine 40(8), 102–114 (2002)
7. Bandara, A.K., Lupu, E.C., Moffett, J., Russo, A.: A Goal-based Approach to Policy Refinement. In: Proceedings 5th IEEE Workshop on Policies for Distributed Systems and Networks (June 2004)
8. Supadulchai, P., Arve Aagesen, F.: Policy-based Adaptable Service Systems Architecture. In: 21st International Conference on Advanced networking and Applications. IEEE (AINA 2007)(2007)
9. Samaan, N., Karmouch, A.: An Automated Policy-Based Management Framework for Differentiated Communication Systems. IEEE Journal on Selected Areas in Communications 23(12) (December 2005)
10. Sutton, R., Barto, A.S.: Reinforcement Learning: An introduction. MIT Press, Cambridge (1998)
11. Watkins, C.J.C.H., Dayan, P.: Technical Note: Q-Learning. Machine Learning 8, 279–292 (1992)
12. Autonomic Internet Project (2009), http://www.ist-autoi.eu/autoi
13. http://www.opnet.com (2009)

Answering Queries Using Cooperative Semantic Caching

Andrei Vancea and Burkhard Stiller

Department of Informatics IFI, University of Zurich
Binzmühlestrasse 14, CH—8050 Zürich, Switzerland
{vancea,stiller}@ifi.uzh.ch

Abstract. Semantic caching is a technique used for optimizing the evaluation of database queries by caching results of previous answered queries at the client side and using the cached results when trying to answer new queries. Before sending a query to the database server, the client first checks, if there are any cached query results that semantically contain the new query or parts of the query. If such cached results are found, they can be used when answering the new query. Otherwise, the query will be answered by the database management server.

This paper proposes to extend the general semantic caching mechanism by enabling clients to share their local semantic caches in a cooperative matter. If a particular query cannot be answered using the local cache, the system will verify, if there are other clients, located across the Internet, that are able to answer the query using the data stored in their caches. Such an approach will increase the throughput of database servers, because servers will only receive queries that cannot be answered using the cooperative cache concept.

1 Introduction

Database managements systems are, most of the time, build using the classical client/server architecture [4]. A client sends a query (usually expressed in the Structured Query Language, SQL) to the database server and waits for the result of the execution. Caching data on the client side represents an important technique used both, for reducing the execution time of queries and also for increasing the throughput of the server [6].

In case of the application of the semantic caching mechanism [5] clients cache the result of the execution of old queries. In some cases, a subsequently query can be executed only using the cached data. Consider the following example: a client sends to the server a query, asking for all persons older then 15 (Q_1 : *select * from persons where age > 15*). The server returns the result set, and the client stores it in the local cache. Later, the client requires all persons older then 18 (Q_2 : *select * from persons where age > 18*). It can be clearly seen that the answer for Q_2 is totally subsumed by Q_1. Thus, Q_2 can be answered locally using the cached result set of Q_1. In the general case, new queries are, of course, not always totally contained in the cached queries. When there is just an overlapping between the new and the cached queries, the query is split into two disjoints parts: one that can be answered using the data contained in the cache (which is called the probe query) and a remainder query, that must be executed in the server [3]. Thus, a subsequent query that asks for all persons older then 10 (Q_4 : *select * from persons where age > 10)* will be split in a probe that asks for all persons older then 15 and a remainder that asks for all persons between 10 and 15. This split is done automatically, by analyzing the semantics of the queries.

Peer-to-peer networks have been applied successfully for enhancing beyond the traditional client-server communication, thus they are applicable to the distribution problem outlined. [8] presents CoopNet, a cooperative network architecture, where clients cooperate in order to improve the overall network performance. It is described how CoopNet is used for solving Web flash crow scalability problems. In this approach, clients that have already downloaded web content start serving the content to other clients, relieving the server of this task. The redirection of requests from the server to other clients is handled by a centralized component running at the server side. Thus, this approach does not integrate the distribution aspect.

Therefore, this paper develops a distributed and cooperative approach for reducing the load of database servers. Using a semantic caching approach, clients will store in their local caches the result of queries they requested. Local caches will be shared between clients in an cooperative matter. Before sending a query to the server, it will first be checked, if there are any other clients that have entries in their cache that can be used for answering the requested query. If such clients are found, their cache entries will be used when answering the query.

2 Approach

In the new approach proposed, clients are allowed to share cached query results in a cooperative matter. In order for this to be accomplished, a system named CoopSC (Cooperative Semantic Cache) is designed, which allows clients to register queries for which they have cached results and also to search for queries stored in the collaborative cache that subsume or overlap new queries for which they want the result. Two solutions for this system (Fig. 1) are foreseen: a *centralized* approach and a *fully distributed* one.

When using the centralized approach, clients register and look for queries in a centralized *cache manager*. The cache manager keeps, for each client, queries for

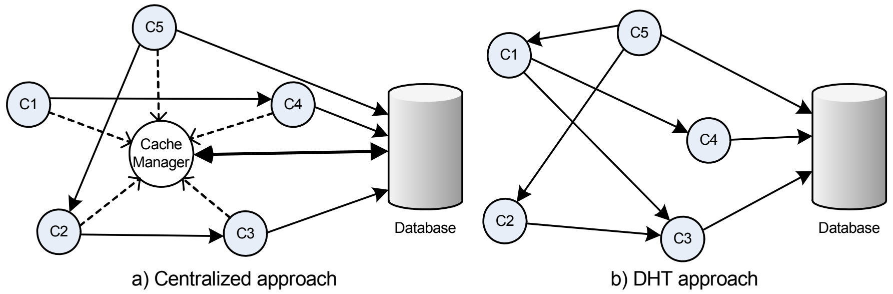


Fig. 1. CoopSC architecture

which they have cached result sets. When a client decides to cache a new query or to drop an existing query, the cache manager must be notified. Before sending a query to the database server, the client will connect to the cache manager and asks for a query cached by another client that subsumes or overlaps its original query. If the cache manager finds one, it will return the identify of the client, the probe query and the remainder. The initial client will connect in turn to the client returned by the cache manager and asks the probe query. The remainder query can be executed by the database server or by using the semantic cache of a different client. This solution is similar with the one used by the CoopNet [8] system. The web server knows what data end-hosts contain, and redirects the requests from the server to clients when needed. In the CoopNet system, the mechanism for selecting the client to which to redirect a request is fairly simple. The web server only has to know the list of clients that keep the latest version of a particular web page. However, in CoopSC, deciding which clients can be used when answering a query, is much more difficult. It must determined which clients have cached queries that subsume or overlap the new query. In order for this to be determined, the semantics of these queries — stored in the cache and of the new query — must be analyzed. Since for this approach the description of all queries cached by different clients are stored in a central point, query containment verification and query rewriting are simplified. Unfortunately, this approach could have scalability problems. Because the cache manager must be contacted before every query execution it could become a bottleneck of the system.

A different approach is possible by applying a Distributed Hash Table (DHT) for indexing queries cached by different clients. After deciding to cache a query, a client will store the description (SQL) of the query into the DHT. The DHT will also be used, when looking for a query cached by another client that contains or overlaps a given query. On one hand, the advantage of this approach is that the cache content is stored in a fully distributed way. Thus, this solution will be much more scalable. On the other hand, looking for a query from the DHT that subsumes or overlaps a given query becomes much more difficult. Special consideration must be taken about the way queries are indexed in the DHT.

CoopSC has similarities with the Wigan [2] system. The purpose of both systems is to cache old result of database queries in order to answer new queries. The main difference is the way in which cache results are chosen for answering new queries. In Wigan, a cached query Q_1 can be used for answering a query Q_2 only, if Q_2 is strictly subsumed by Q_1. In real world applications, the number of cases in which this happens is limited. CoopSC does also support cases in which there is only an overlapping between Q_1 and Q_2.

Query containment and rewriting determine a fundamental concept related to semantic caching. A query Q_2 is set to be contained is a query Q_1 if Q_2 produces a subset of the answers of Q_1. If it has been decided that Q_2 is contained in or overlaps Q_1, Q_2 must be reformulate with the respect to the structure of Q_1. There has been an intensive research in the database community related to the query containment and rewriting problems [6]. It has been proven that the query containment problem is undecidable for relational algebra and SQL but, there are efficient algorithms for queries that have particular constraints [9], [7]. It is planned to investigate how these algorithms can be adapted and used in the cooperative cache architecture proposed.

Cache consistency is another important issue that must be handled by the CoopSC approach. After the execution of a modification in the server, some cache entry can become outdated. CoopSC must contain a mechanism for invalidating cache entries stored by clients that are no longer up-to-date.

In order to evaluate all benefits of the CoopSC architecture, it must be shown that, under heavy load, a database server performs better, when using the cooperative cache. For this to be proven, it is planned to test the functionality of the CoopSC by using a test-bed consisting of a database server and a number of clients machines that execute, in parallel, queries on the database. The same set of queries will be executed under three different scenarios: (a) without using the cache; (b) using only the local semantic cache; and (c) using the cooperative semantic cache. In each scenario the average query response time will be measured. It is expected that the average response time will be lower when using the distributed cache approach.

3 Conclusions

This paper skteches a new approach for answering database queries using a cooperative semantic caching mechanism. This solution proposed and partially outlined in terms of key aspects will increase the throughput of database management servers. The respective and general architecture of the new system termed CoopSC was described. The main issues concerning the implementation of the system were discussed. Furthermore, for the upcoming fine design and implementation of CoopSC, this paper also presents a possible evaluation.

References

1. Carey, M.J., Franklin, M.J., Livny, M., Shekita, E.J.: Data caching tradeoffs in client-server DBMS architectures. ACM SIGMOD Record 20(2), 357–366 (1991)
2. Colquhoun, J., Watson, P.: A Peer-to-Peer Server based on BitTorrent, Technical Report No. 1089, School of Computing Science, Newcastle University (April 2008)
3. Dar, S., Franklin, M.J., Jonsson, B., Srivastava, D., Tan, M.: Semantic Data Caching and Replacement. In: 22th International Conference on VLDB, Bombay, India, September 1996, pp. 330–341 (1996)
4. Garcia-Molina, H., Ullman, J.D., Widom, J.D.: Database Systems: The Complete Book. Prentice Hall, Englewood Cliffs (2008)
5. Godfrey, P., Gryz, J.: Answering Queries by Semantic Caches. In: Bench-Capon, T.J.M., Soda, G., Tjoa, A.M. (eds.) DEXA 1999. LNCS, vol. 1677, pp. 485–498. Springer, Heidelberg (1999)
6. Levy, A.: Answering Queries Using Views: A Survey. The VLDB Journal 10(4), 270–294 (2001)
7. Levy, A., Rajaraman, A., Ordille, J.J.: Querying Heterogeneous Information Sources Using Source Descriptions. In: 22nd International Conference on VLDB, Bombay, India, September 1996, pp. 251–262 (1996)
8. Padmanabhan, V., Sripanidkulchai, K.: The case for cooperative networking. In: International Peer-To-Peer Workshop, Cambridge, MA, USA, March 2002, pp. 178–190 (2002)
9. Pottinger, R., Levy, A.: A Scalable Algorithm for Answering Queries Using Views. The VLDB Journal 9(1), 484–495 (2000)

Towards Cost-Aware Multipath Routing

João Taveira Araújo, Miguel Rio, and George Pavlou

Department of Electronic and Electrical Engineering
University College London
{j.araujo,m.rio,g.pavlou}@ee.ucl.ac.uk

Abstract. Traditional approaches to multipath routing ignore the economic incentives necessary in aligning both networks and users towards a common goal. While theory suggests congestion pricing can be used to maximize social welfare, even within such a framework we have no consistent method for evaluating, constructing and disseminating paths. Our research focuses on building on existing congestion pricing models to provide scalable and differentiated edge selectable routes. By redesigning traffic management to not only provide paths, but also price them accordingly, the resulting cost-aware multipath architecture could alleviate the tension between different stakeholders by evolving the Internet towards a fully functional market.

1 Introduction

The ongoing feud between network operators and bandwidth-heavy applications has shown current resource allocation mechanisms are remarkably inept at mediating between both parties. Users expect to fully utilize the network resource pool, irrespective of the harm they cause others. With no form of protecting sporadic users from this onslaught of traffic, operators are forced to penalize heavy users through the deployment of ad-hoc solutions such as the use of deep packet inspection or degrading performance artificially. Both solutions carry their own problems - they lead to an inefficient use of resources and hinder innovation at the edges - and neither is strictly necessary.

A potential workaround for this resource allocation dilemma was presented by Kelly in his seminal work on network utility maximization [1]. Using an optimization-based model, Kelly proved social welfare could be maximized if sources are charged a shadow price proportionate to the congestion they cause. Initial interest in applying these results however was tepid. On first inspection, the Internet's architecture did not lend itself to such fine grained charging models. Additionally, many feared customer hostility toward price discrimination which has historically affected some transport networks [2]. Recent research however has re-kindled interest in drawing inspiration from economic models, such as Kelly's, to solve some of the Internet's inherent flaws. A recent protocol, re-ECN [3], has not only demonstrated that congestion charging can co-exist with flat priced subscription models, but also that enforcing cost fairness can be implemented scalably at the expense of edge policing [4].

The most significant effect of these results has been to convince the congestion control community to slowly abandon TCP-friendliness and adopt a stronger paradigm [5]. A network with more active participation in congestion control, with the accurate and truthful reflection of costs to customers, opens up the possibility of exploring service models which have thus far remaining largely undeployed due to incentive issues. One such service is multipath routing.

1.1 Exploring Path Diversity

As application needs diversify, the Internet must become increasingly versatile in meeting demands which are at times at odds with each other. Traditional research in Quality of Service has favoured the view that networks should offer specialized traffic classes, each optimized for specific constraints. A radically different approach is for the network to become more agnostic and allow users to explore path diversity and load balance their traffic across multiple paths.

The benefits of multipath routing are well understood, allowing applications to achieve greater end-to-end reliability, higher throughput and optimize traffic according to requirements. Despite various proposals over the past decade (e.g. [6][7]), multipath routing has been notoriously difficult to implement in practice. The simplest path to deployment would rely on the use of source routing. Unfortunately, source routing has always been at odds with traffic engineering - users are empowered to bypass network policy or engineering decisions undermining their effectiveness. With subtle pricing schemes however the balance of power is shifted towards network operators who are able to charge user traffic according to congestion caused. This market will approach equilibrium so long as the cost a network applies is inferior to the utility a user extracts for a particular path.

In the absence of congestion charging, networks have no incentive to provide additional paths. Under such conditions, multipath routing may still be deployed through the use of multihoming, as proposed in [8], requiring no network assistance. While easily deployable, this may further aggravate network operators. Direct competition between providers would be a positive development, but limiting operators to offering a one-size-fits-all path may further accelerate commoditization - effectively a race to the bottom.

As such, we believe congestion pricing should play an integral role in evaluating and constructing multiple paths, with differing characteristics, to be offered by the same provider. A shift towards cost fairness, coupled with the widespread use of multiple paths, could force traffic management to evolve from a tool predominantly responsible for minimizing costs to a valuable asset in generating revenue.

2 Building a Cost-Aware Multipath Architecture

The confluence of congestion pricing and edge selectable routes provides an opportunity to rethink the role of traffic management in a network which would behave itself increasingly as a market. Our research focuses on evolving traffic management to offer multiple paths to users, thereby allowing the same operator to tailor paths to different uses. What distinguishes this multipath solution from related work is the use of a pricing framework to overcome incentive issues and ensure users only use what they are willing to pay for. Under these assumptions, our specific goal is to answer each of the following questions:

– *What pricing strategies must be in place to accurately reflect cost?*

The use of ECN markings as a cost indicator may suggest congestion is the only metric networks should charge users for. However, it is feasible that networks may induce markings to internalize other costs, such as providing low delay. We intend to evaluate whether a coarse accountability framework like re-ECN provides sufficient flexibility for such fine-grained pricing. In particular, we wish to understand whether ECN markings alone are sufficient to accurately convey non-congestion costs, or whether we must extend re-ECN to allow greater flexibility.

An additional concern is how traffic engineering can translate policies through the use of pricing. With the possibile advent of user selectable routes, cost would ensure operators maintain control over their network, effectively replacing the role of link weights. How traffic engineering can set costs to ensure network resources are used efficiently and provide an adequate response to market demand is an open issue which is intrinsically related to our next research question.

– *How can we construct path segments locally to satisfy different demands?*

An open question in multipath routing is assessing how much control over path selection can be passed onto the end-user. Ideally, users would have the power to freely specify paths as deemed fit. In reality, this offers little benefit and carries the same scalability issues which plagued strict source routing. Offering users a limited array of paths may therefore strike a more appropriate balance between convenience and diversity. The use of pricing may be valuable in evaluating the relative merit of different paths. What is missing from the current network architecture is a mechanism for creating and diffusing alternative routes.

Within a domain, traffic engineering will be responsible for constructing path segments tailored for different needs. This amounts to evolving traditional offline intradomain traffic engineering, which optimizes routes based on expected traffic matrices, to a tool which calculates multiple path segments based on expected market demand. Our work evaluates the implications pricing has on current traffic engineering practices, and how these must be altered to maximize revenue over multiple paths.

– *How can we scale this solution across economic entities?*

Given a domain can construct multiple paths with differing characteristics, our focus turns to how such paths can be disseminated across economic entities. Pathlet routing [9], where path fragments are propagated throughout the network and assembled by sources to provide an end-to-end route, could provide a starting point for developing a multipath routing architecture which is able to span the entire Internet. By extending pathlet routing to take into account average congestion, as provided by re-ECN, sources could concatenate path fragments and have an accurate estimate of congestion cost before attempting to balance traffic between links. This would represent a tangible improvement on re-ECN, which only provides a meaningful estimate of path congestion over a long-lived flow.

3 Conclusions

Using the principles of congestion pricing, our research aims to provide operators with means of managing traffic in an increasingly distributed environment as the Internet adopts a multipath paradigm in order to fully utilize the network resource pool. By extending traffic engineering to apply economic pressure on flows and constructing paths to meet the demands of differing application requirements, we believe the work described herein will aid in evolving traffic management to meet the demands of the Future Internet.

References

1. Kelly, F.: Charging and rate control for elastic traffic. European Transactions on Telecommunications 8, 33–37 (1997)
2. Odlyzko, A.: The evolution of price discrimination in transportation and its implications for the internet. Review of Network Economics 3, 323–346 (2004)
3. Briscoe, B., Jacquet, A., Moncaster, T., Smith, A.: Re-ECN: Adding accountability for causing congestion to TCP/IP. Internet Draft draft-briscoe-tsvwg-re-ecn-tcp-06.txt, Internet Engineering Task Force (July 2008) (Work in progress)
4. Jacquet, A., Briscoe, B., Moncaster, T.: Policing Freedom to Use the Internet Resource Pool. In: Proc. Workshop on Re-Architecting the Internet (ReArch 2008) (2008)
5. Mathis, M.: Reflections on the TCP macroscopic model. SIGCOMM Comput. Commun. Rev. 39(1), 47–49 (2009)
6. Yang, X., Clark, D., Berger, A.: NIRA: A New Inter-Domain Routing Architecture. IEEE ACM Transactions on Networking (Janurary 2007)
7. Xu, W., Rexford, J.: MIRO: multi-path interdomain routing. In: SIGCOMM 2006: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 171–182. ACM, New York (2006)
8. Wischik, D., Handley, M., Braun, M.: The resource pooling principle. SIGCOMM Comput. Commun. Rev. 38(5), 47–52 (2008)
9. Godfrey, P., Shenker, S., Stoica, I.: Pathlet routing. In: Proceedings of the Seventh Workshop on Hot Topics in Networks (HotNets-VII), ACM SIGCOMM (2008)

Author Index