Distributed Systems
ECEG-6504

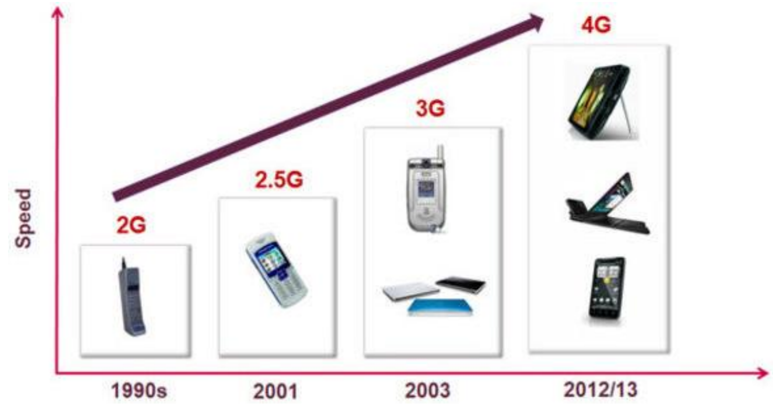
Introduction

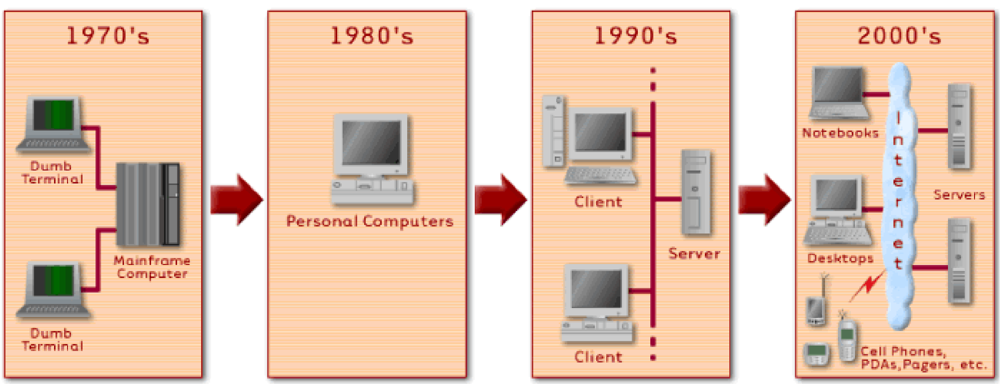
Surafel Lemma Abebe (Ph. D.)

Topics

- What is a distributed system?
- Examples of distributed systems
- Goals and challenges of distributed systems
- Software concepts
- Architectural styles
- System architectures

What is a distributed system?





- Revolution of computers and networks
  - High computing power with low cost
  - High speed networking

=> **Network of computers is everywhere**

What is a distributed system?...

- Abundant “unused” resources
  - Resources
    - Hardware
    - Software

- Need for sharing resources that can be shared over a network

  => Motivation for DS

What is a distributed system?...

"A distributed system is a collection of independent computers that appear to the users as a single coherent system." [Tanenbaum]

- Aspects
  - System Architecture
    - » Independent computers => autonomos
  - Users' perception
    - » Single coherent system => collaboration

"A system in which (hardware or software) components located at networked computers communicate and coordinate their actions only by message passing." [Coulouris]

What is a distributed system?...

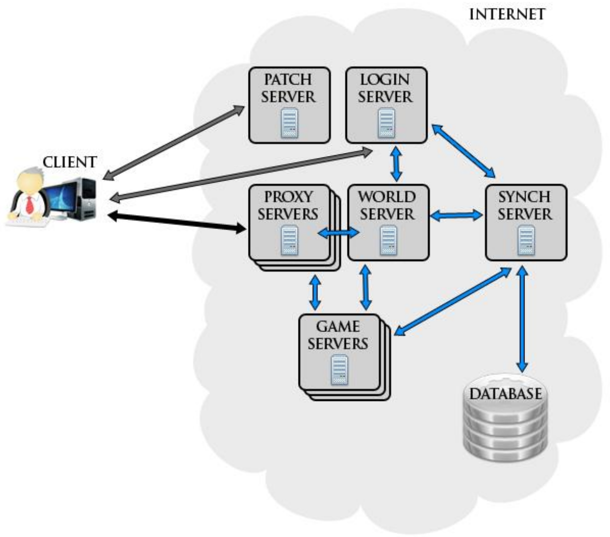
- Characteristics
  - Differences between computers is hidden
  - How computers communicate is hidden
  - Internal organization of distributed system is hidden
  - Interaction with a DS is consistent and uniform regardless of where and when interaction takes place
  - Easy to expand or scale
  - Distributed system is normally continuously available
  - Users and applications do not notice that parts are being added, replaced or fixed

Examples of distributed systems

- Network file systems
  - Files are accessed using the same interfaces and semantics as local files
- Web search
  - Web search engine indexes the entire contents of WWW
  - Complex task
    - WWW consists over 63 billion pages
    - Analyze the entire web content
    - Perform sophisticated processing
  - Example
    - Google
      - A number of data centers all around the world
      - Distributed file systems and storage systems, ….

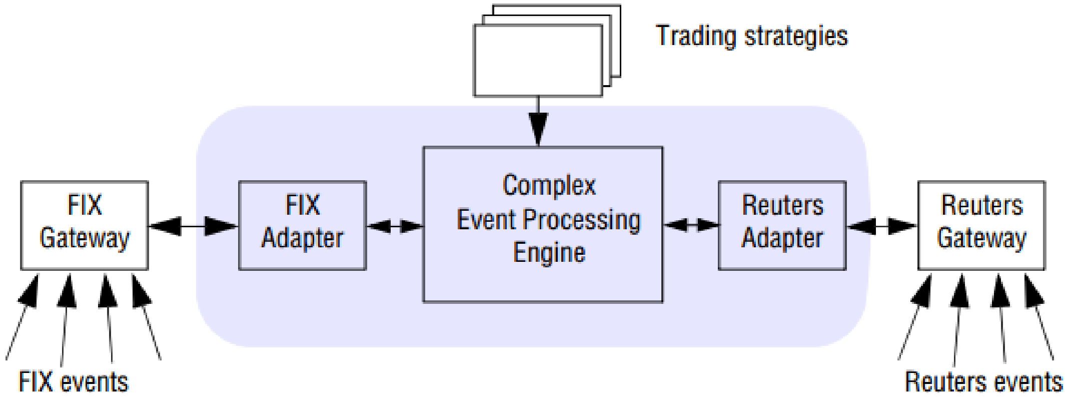
Examples of distributed systems…

- Massively multiplayer online games
  - Large number of users interact through Internet with a persistent virtual world
  - Example: Face of Mankind
    - First and third person online action role playing game
  - Complex virtual realities
  - Different factions
  - Increasing number of players
  - Challenges
    - Fast response time
    - Real time propagation of events



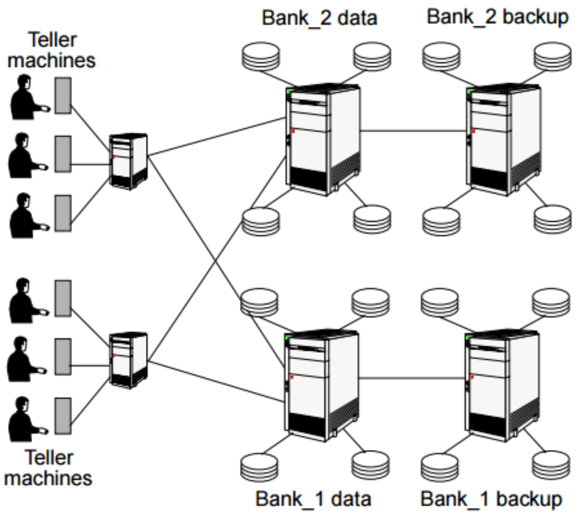
Examples of distributed systems…

- Financial trading
  - Real time access to a wide range of information sources
  - Focus on communication and processing of items of interest
    - Example: share prices and trends
    - Distributed event-based systems
  - Characteristics
    - Heterogeneous formats (e.g., Financial Information eXchange)
    - Real time processing of events that arrive at a rapid rate



Examples of distributed systems…

- Automated banking system
  - Challenges
    - Security
    - Reliability
    - Consistency of replicated data
    - Concurrent transactions
    - Fault tolerance



- SETI@HOME
  - SETI = Search for Extraterrestrial Intelligence
  - Search for extraterrestrial life (intelligence) by analyzing specific radio frequencies emanating from space
  - Internet based public volunteer computing project
  - Uses Berkeley Open Infrastructure for Network computing (BONIC)

Goals and challenges of DS

- Goals
  - Make resources easily accessible
  - Hide the fact that resources are distributed across a network
  - Open
  - Scalable

Goals and challenges of DS …

- Making resources accessible
  - **Goal:** To make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way
  - Security issues, availability, fairness, …
- Distribution transparency
  - **Goal:** Hide the fact that its processes and resources are physically distributed across multiple computers
  - Types of transparency

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is physically located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several independent competitive users
Failure	Hide the failure and recovery of a resource

Goals and challenges of DS …

- Distribution transparency…
  - Degree of transparency
    - *Attempting to completely hide all distribution aspects may be too ambitious*
      - Users may be located in different time zones
      - Completely hiding failures is impossible
        - » Slow system vs. a failed system
      - Trade off between high degree of transparency and performance of a system
        - » Updating a data replica

Goals and challenges of DS …

- Openness of DS
  - **Goal:** Allow interaction irrespective of the underlying environment
    - Services should be offered using standard rules
    - Properly define the interface of a service
      - Should be **complete** and **neutral**
  - Completeness and neutrality are important for
    - Interoperability
    - Portability
  - **Interoperability**
    - Characterizes co-existence and the capability to work together by relying on each other’s services
  - **Portability**
    - Characterizes capability of execution without modification in different environment
  - **Extensibility**
    - Should be easy to add new components or replace existing ones without affecting those components that stay in place
  - How to achieve openness?
    - Make DS independent from heterogeneity of the underlying environment

Goals and challenges of DS …

- Openness of DS …
  - Policies vs. Mechanisms
    - Examples:

Policy	Mechanisms
Define type and level of security	Offer different levels and algorithms
Define QoS	Provide adjustable QoS parameters
Web caching	Allow different settings for caching

Goals and challenges of DS …

- Scalability
  - Dimensions
    - Size
    - Geographically
    - Administrative
  - Challenges
    - Size
      - We can easily add more users and resources to the system
      - Usually centralized systems are not good

Concepts	Example scenario
Centralized services	A single (bank account) server for all users
Centralized data	A DNS server with single table
Centralized algorithm	Doing routing based on complete information

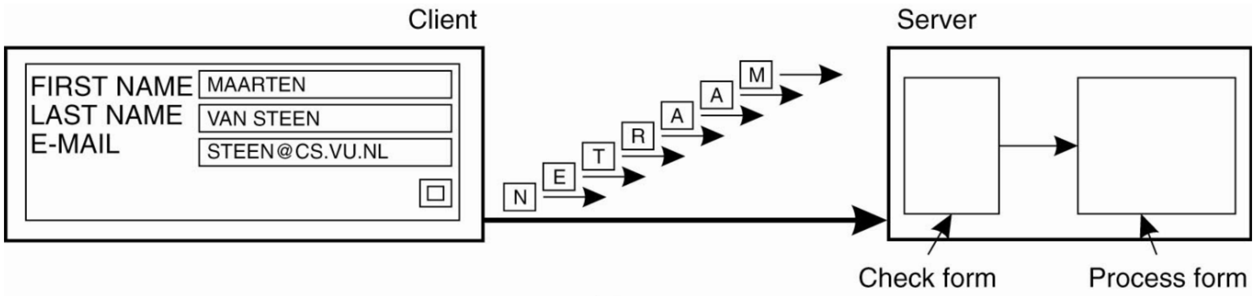
      - In some cases using a centralized system is unavoidable

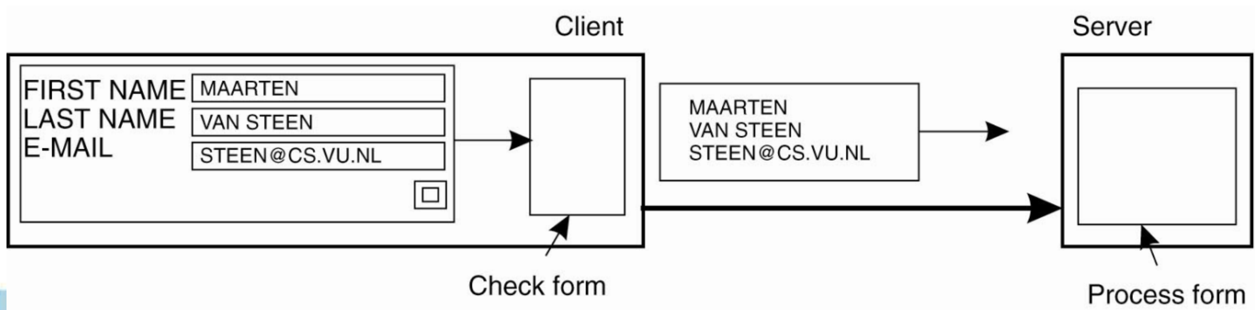
Goals and challenges of DS …

- Challenges in scalability …
  - Geographical
    - Users and resources may lie far apart
    - Latency in communication
    - Reliability
  - Administrative
    - Can still be easy to manage even if it spans many independent administrative organizations
    - Conflicting policies with respect to resource usage, management, and security

Goals and challenges of DS…

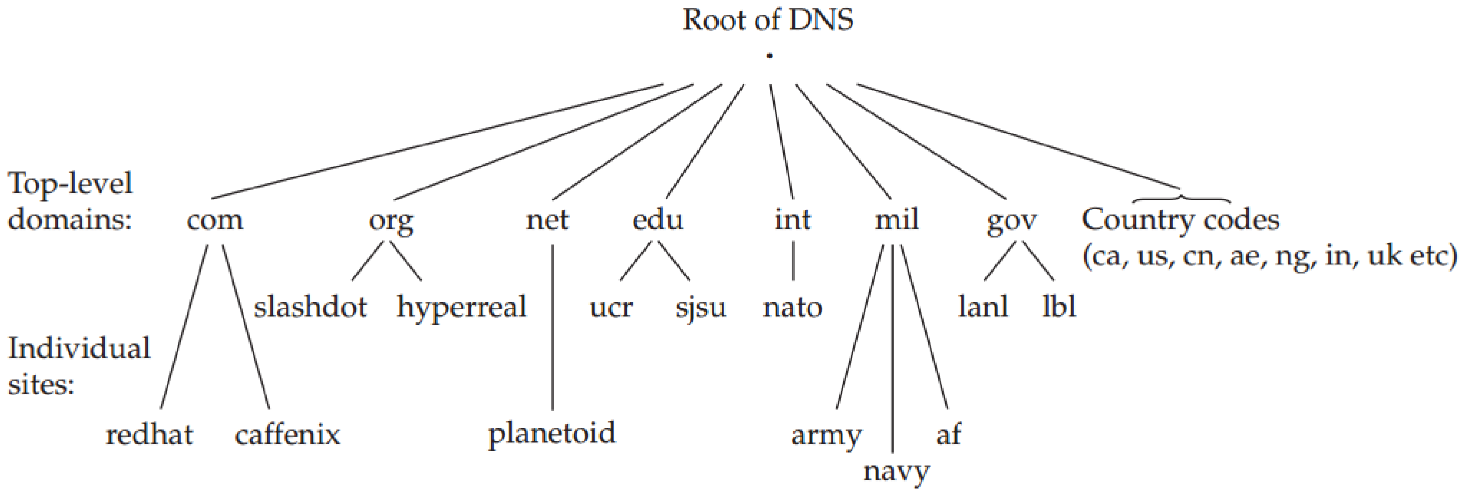
- Techniques for scaling
  - Hide communication latencies
    - Avoid waiting for responses
      - Make use of asynchronous communication
      - Have separate handler for incoming responses





Goals and challenges in DS …

- Techniques for scaling …
  - Distribution
    - Splitting a component into smaller parts, and subsequently spreading it across multiple machines
      - Move computations to clients
      - Decentralized naming services
      - Decentralized information systems



Goals and challenges of DS …

- Techniques for scaling …
  - Replication and caching
    - Increases availability
    - Balances load
    - Hides much of the communication latency
    - Challenge
      - An update of a copy could lead to inconsistency
      - Global synchronization makes large scale solutions impractical
    - Tolerating inconsistencies is application dependent

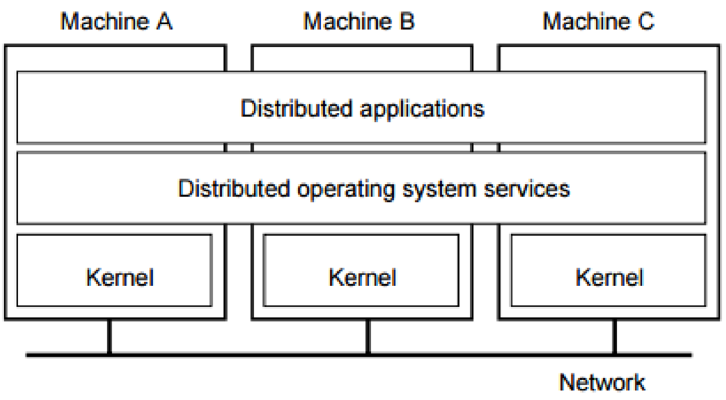
Pitfalls of developing DS

- False assumptions
  - The network is reliable
  - The network is secure
  - The network is homogeneous
  - The topology does not change
  - Latency is zero
  - Bandwidth is infinite
  - Transport cost is zero
  - There is one administrator

Software concept

- OS types for multiprocessors and multicomputers
- Distributed OS
  - Tightly coupled OS for multiprocessors and homogeneous multicomputers
  - Goal: Hide and manage hardware resources
  - Multi-processor OS
    - Shared memory
    - Handles multiple CPUs
    - Aim at supporting high performance through multiple CPUs
    - Communication => through shared memory location
    - Protection => synchronization primitives (e.g., semaphores)

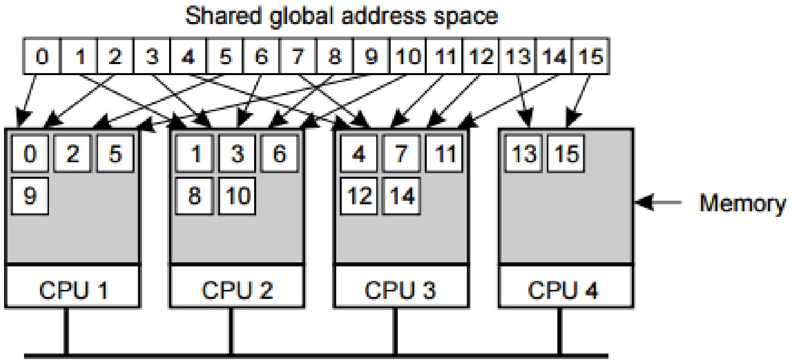
Software concepts…

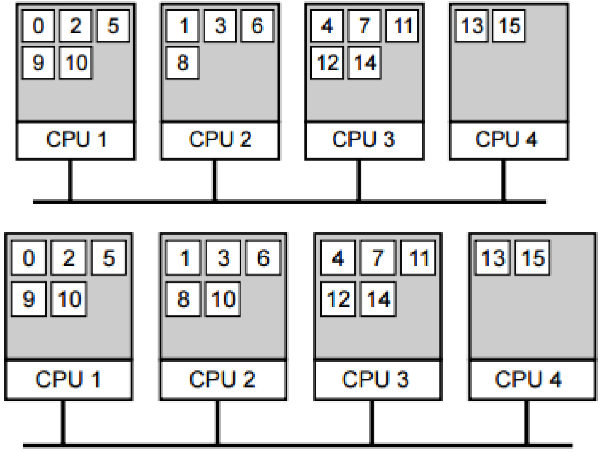


- Distributed OS…
  - Multicomputer OS
    - Usually same OS on all machines
    - Memory not shared
    - OSs on each computer knows about the other computers
    - Communication => message passing between processors
    - Synchronization => message passing
    - Each machine has its own kernel
      - Manages local resources + handles interposes communication
    - Services are usually transparently distributed across computers
    - E.g., Amoeba and MOSIX

Software concepts…

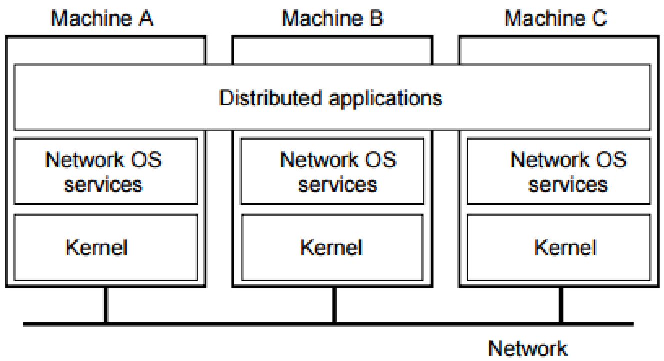
- Distributed OS…
  - Multicomputer OS…
    - Programming multi-processor is easier
      - Accessing shared data + simple synchronization methods
    - Distributed shared memory
      - Address space is divided up into pages with the pages being spread over all the processors in the system
      - When a processor references an address that is not present locally, a trap occurs, and the OS fetches the page





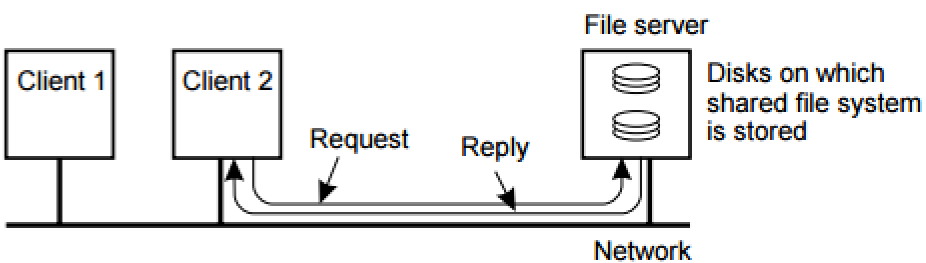
Software concepts …

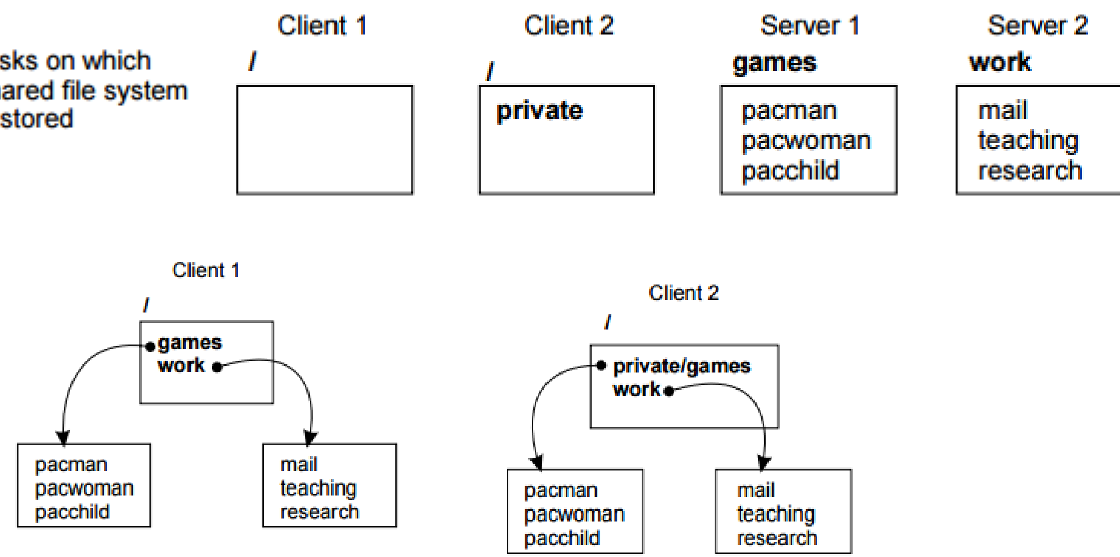
- Network OS
  - Loosely coupled OS for heterogeneous multicomputer
  - Doesn't assume that the underlying OS should be managed as if it were a single system
  - Goal: offer local services to remote clients



Software concepts…

- Network OS …
  - Shared global file system
  - Different view of file system





Software concepts…

- Network OS…
  - Characteristics
    - Each computer has its own OS with networking facility
    - Computers work independently
    - Services are to individual computers
    - Processes only share files
    - Compared to distributed OS
      - Lacks transparency
      - Scalable and open

Software concepts…

- Recap

  "A distributed system is a collection of independent computers that appear to the users as a single coherent system." [Tanenbaum]

  – Question

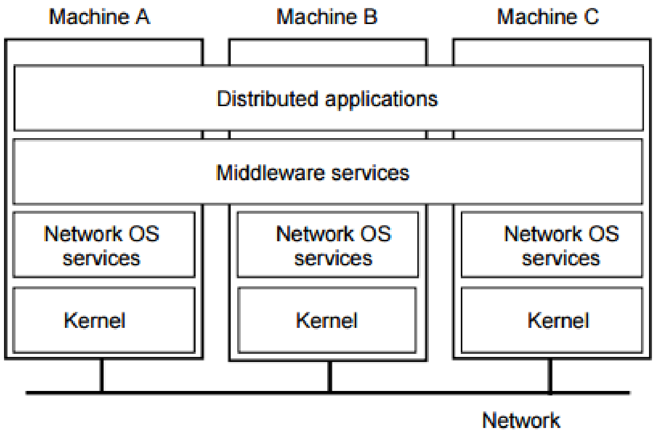
    - Are distributed OS and network OS distributed systems?
    - Is it possible to develop a distributed system which has the best of distributed OS and network OS?

      – *Additional layer of software that is used in network OS to more or less hide the heterogeneity of the collection of underlying platforms and also improve distribution transparency*

      => Middleware

Software concepts…

- Middleware



  - Doesn’t manage an individual computer (node)
    - Network OS manages local resources and communication with other computers
  - OS on each computer need not know about the other computers
  - OS on different computers need not be the same
  - Services are generally transparently distributed across computers

Software concepts…

- Middleware …
  - All resources are treated as files (adopted from UNIX)
  - Offer high-level communication facilities
    - Remote procedure calls (RPCs)
      - Allow a process to call a procedure whose implementation is located on a remote machine
    - Distributed objects
      - Transparently invoke objects residing on remote machines

Software concepts…

- Comparison

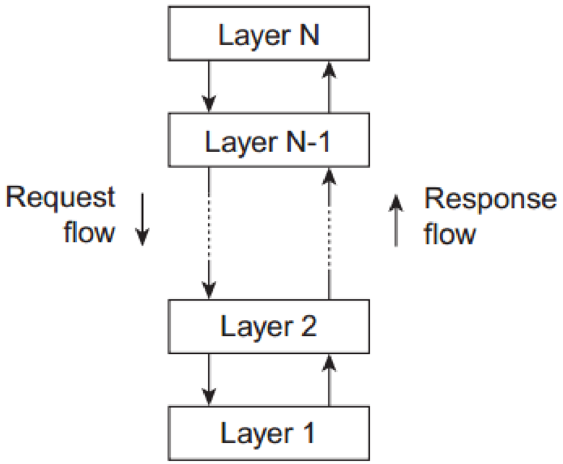
Item	Distributed OS		Network OS	Middleware-based DS
	Multiproc.	Multicomp.		
Degree of transparency	Very high	High	Low	High
Same OS on all nodes?	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

Architectural styles

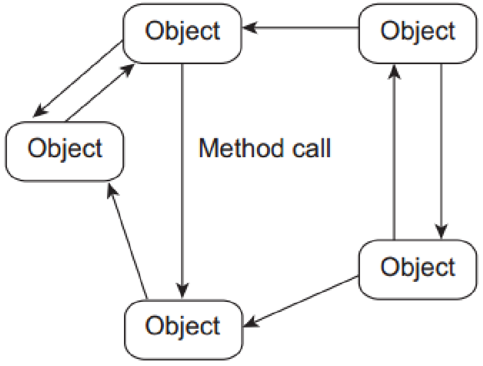
- Provide a high-level view of the distribution of functionality between system components and the interaction relationship between them
- Formulated in terms of
  - Components
  - Connector
  - Data exchanged between components
  - How components are configured
- Idea
  - Organize into logically different components, and distribute those components over the various machines
- Classification
  - Layered architectures
  - Object-based architectures
  - Data-centered architectures
  - Event-based architectures

Architectural styles…

- Layered architecture
  - Requests go down the hierarchy where as results flow upward

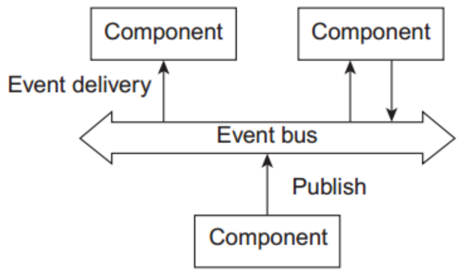


- Object based architecture
  - Components are connected through a (remote) procedure call

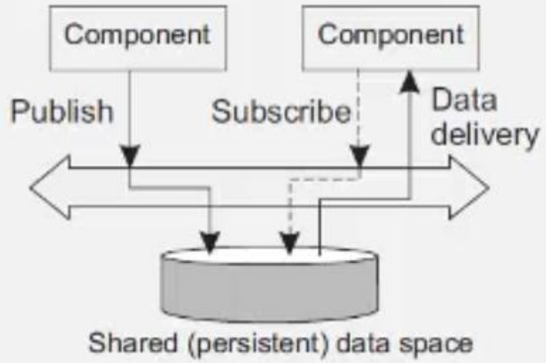


Architectural styles...

- Data-centered architectures
  - Processes communicate through a common (passive or active) repository
  - E.g., network applications that rely on a shared distributed file system
- Event-based architectures
  - Communicate through the propagation of events
  - Publish/subscribe systems (a) => decoupled in **space (referentially decoupled)**
  - Shared data spaces (b) => decoupled in **space** and **time**
    - Uses data-centered architectures



(a)

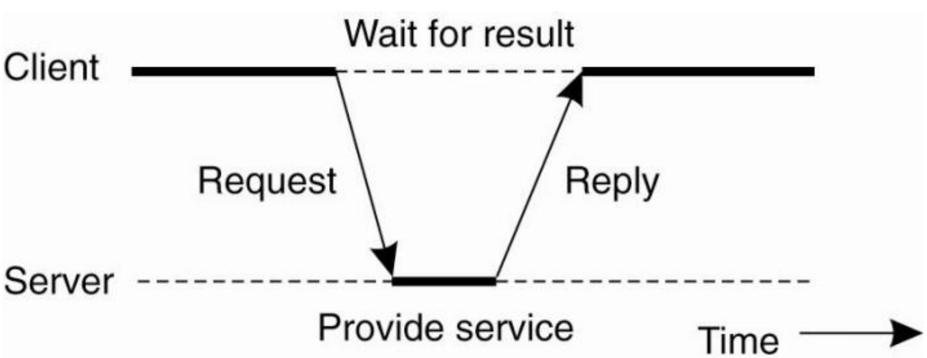


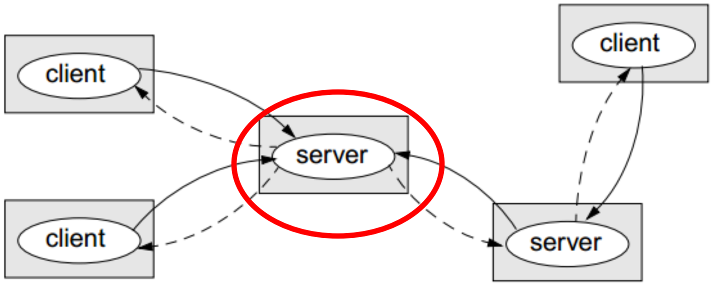
(b)

System architectures

- Organization of components
  - Where components are placed in DS?
    - Centralized , decentralized, and hybrid forms
- Centralized architecture
  - Client-server architecture

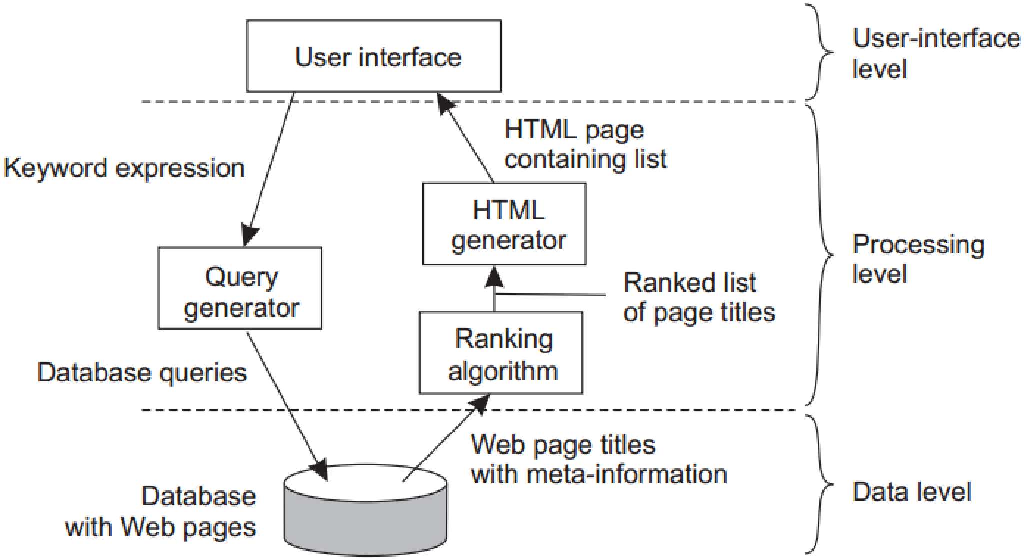
    The system is structured as a set of processes, called servers, that offer services to the users, called clients
    - Characteristics
      - Processes offering services
      - Processes that use services
      - Clients and servers could be on different machines
      - Follow request/reply protocol





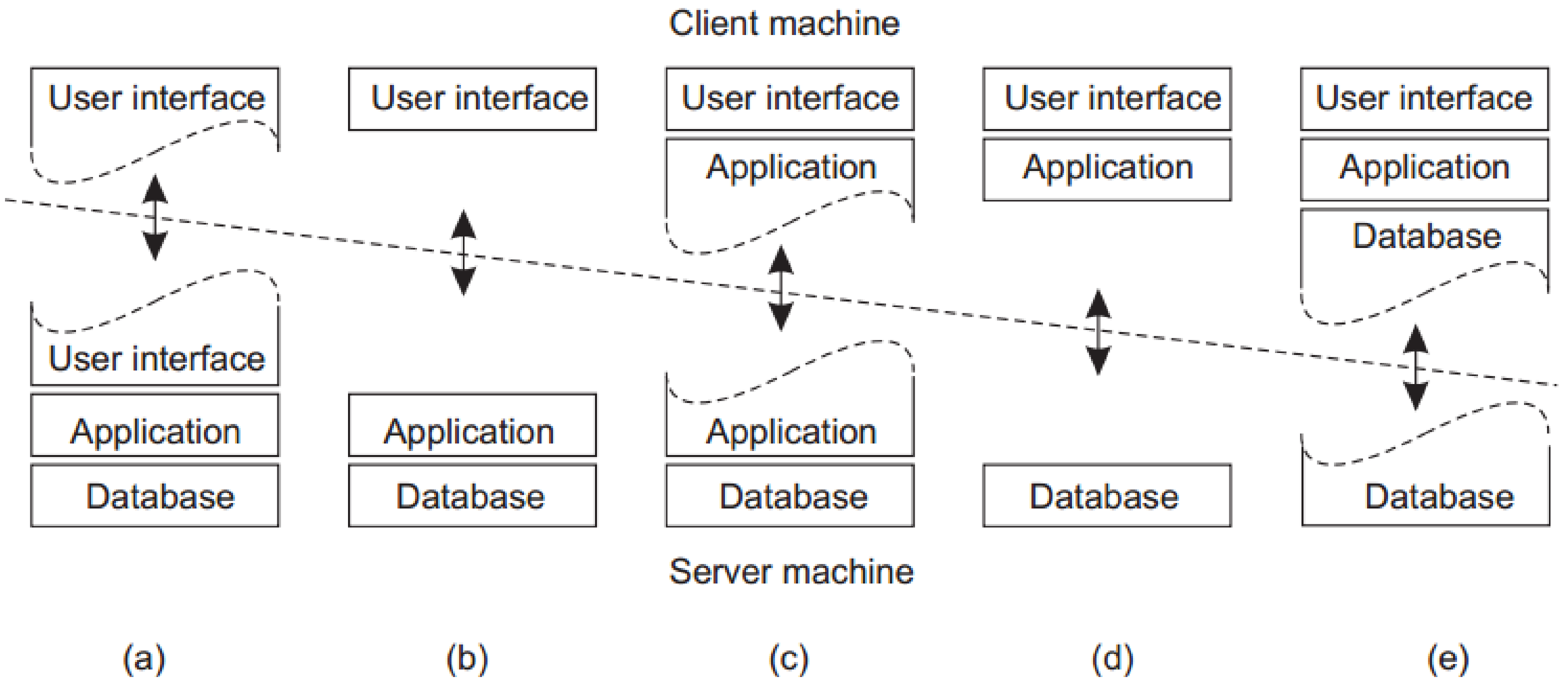
System architectures…

- Centralized architecture…
  - Client-server architecture…
    - Application layering
      - Three-layered view
      - Multi-layered view
    - Three-layered view
      - User interface layer
      - Processing layer
      - Data layer
      - Example
        - » Search engine



System architectures…

- Centralized architecture…
  - Client-server architecture…
    - Multi-tiered architecture
      - Each layer on a separate machine
      - Vertical distribution
    - Client-server configurations



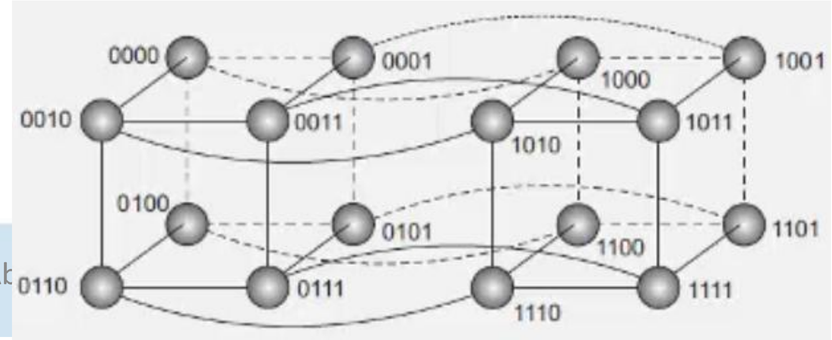
System architectures…

- Decentralized architectures
  - Horizontal distribution
    - Clients and servers are physically split up into logically equivalent parts
    - Each part is operating on its own share of the complete dataset

    => **peer-to-peer** systems
  - Peer-to-peer systems
    - All processes in peer-to-peer system are equal
    - Interaction between processes is symmetric

System architectures…

- Decentralized architectures…
  - Peer-to-peer systems…
    - Structured P2P systems
      - Nodes are organized following a specific distributed data structure
        - » E.g., logical ring, hypercube
      - The organization of nodes uses structured overlay network
      - System provides a LOOKUP(key) to route lookup requests
        - » Key could be file, movie, …

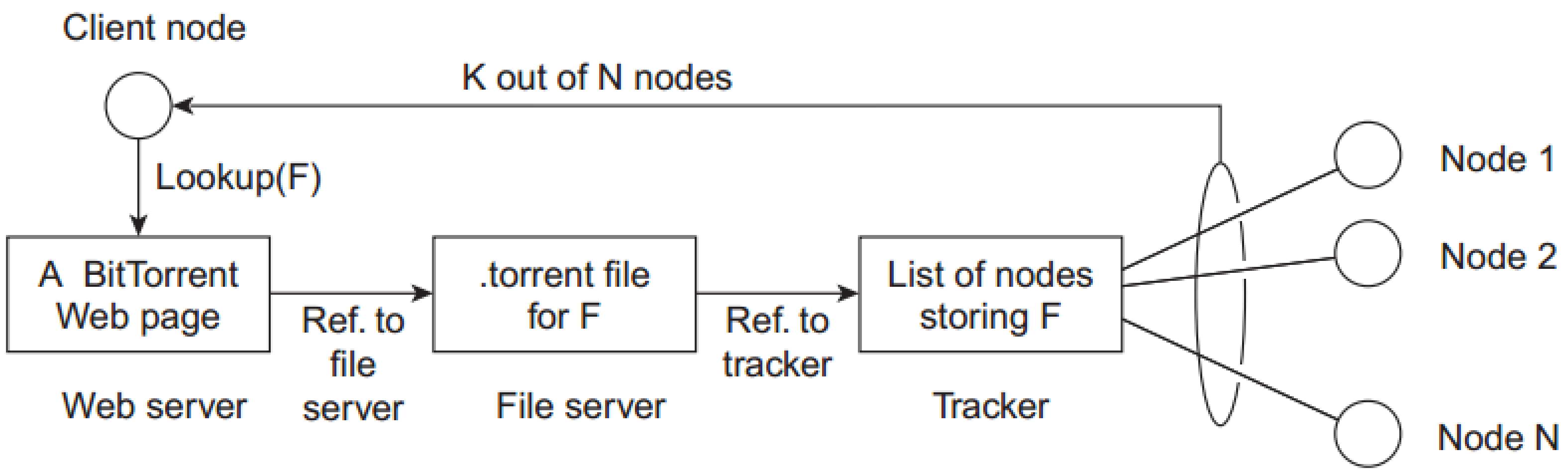


System architectures…

- Decentralized architectures…
  - Peer-to-peer systems…
    - Unstructured P2P systems
      - Rely on randomized algorithms for constructing an overlay network
      - Nodes have randomly selected neighbors, which are also referred to as partial view
      - Information ***can not*** be looked up **deterministically**, hence has to resort to searching
        - » Flooding
          - Node u sends a lookup query to all of its neighbors. A neighbor responds or forwards (floods) the request
        - » Random walk
          - Randomly select a neighbor v. If v has the answer, it replies, otherwise v randomly selects one of its neighbors

System architectures…

- Decentralized architectures…
  - Peer-to-peer systems…
    - Hybrid P2P architectures
      - Collaborative distributed systems
        - » Example: bit-torrent
          - .torrent: contains metadata about the files
          - Tracker: coordinates the file distribution



System architectures…

- Architectures vs. middleware
  - Middleware
    - Forms a layer between applications and distributed platforms
    - Purpose
      - Provide a degree of distribution transparency

    - Follows a specific architecture
    - Benefit
      - Designing applications may become simpler
    - Limitation
      - Middleware may no longer be optimal, may not be good for other applications

=> (Dynamically) adapt the behavior of the middleware

- Example, using interceptors
  - » break the usual flow of control and allow other (application specific) code to be executed