CIS 630 - Fall 2004
Distributed Systems

Lecture 2
Architecture Models

University of Oregon
Department of Computer and Information Science
Business and Logistics

☐ Course webpage
  ◆ State of flux with new updates (stable by weekend)

☐ Programming exercise
  ◆ Updates on demonstration

☐ Term paper assignment
  ◆ Begin thinking about your topic
  ◆ Reference links add

☐ Survey
  ◆ Sent in email and on webpage

☐ Guest lecture for Oct. 12
Lecture Objectives

- Provide conceptual models to motivate distributed system design problems and solutions
- Look at architectural models
  - High-level view of distribution of functionality
  - Relationships between components
  - Useful for evaluating properties of distributed systems
- Look at functional models
  - Vertical views or slices representing some key aspect
  - Sets of issues that must be addressed in design
- Emphasis on middleware to support development
- Synchronisation, reliability, end-to-end argument
Distributed System Models

- Architecture model of distributed system concerned with placement of parts and their relationships
  - Defines the way components of the system
  - Defines how they map to underlying network/computers

- Fundamental models are concerned with more formal descriptions of properties (correctness, reliability, ...)

- Distributed system characteristics addressed by:
  - Interaction model
  - Failure model
  - Security model
Difficulties For / Threats to Distributed Systems

- Widely varying models of use
  - Components are subject to wide variations in workload
  - Some parts of system are poorly connected
  - Applications have different requirements
    - bandwidth and latency

- Wide range of system environments
  - Heterogeneous hardware, OS, and networks
  - Networks vary in performance
  - Systems have widely differing scales

- Internal problems - clocks, data, component failure

- External problems - attacks, data integrity, secrecy
Lamport’s Definition of a Distributed System

- Lamport once defined a distributed system as:

  “One on which I cannot get any work done because some system I never heard of has crashed.”

- Applications need to adapt gracefully in the face of partial failure
Architecture Models

- Goal is to ensure structure will meet requirements
- First simplify and abstract functions of individual components of distributed system, then consider
  - Placement across network of computers, seeking to define useful patterns for data distribution / workload
  - Inter-relationships between components, their functional roles and communication patterns
- Simplification achieved by classifying processes
  - Server, client, peer
  - Identifies: responsibilities, behavior, workload, failure
  - Analysis used to specify placement to meet objectives
System Architectures

- Concerned with division of responsibilities
  - Between system components (apps, servers, processes)
  - Placement on computers in the network
- Implications for performance, reliability, and security
- Types
  - Client-server model
  - Services provided by multiple servers
  - Proxy servers and caches
  - Peer processes
  - Mobile code / agents / spontaneous networking
  - Network computers / thin clients
Client-Server Model / Multiple Servers

- Most common distributed system architecture
- Defines interaction relationship
  - Service: task machine can perform
  - Server: machine that performs the task
  - Client: machine that is requesting the service
- Model allows chaining and hierarchy
  - Servers may be clients of other servers
    - example: web server being a client of local file server
- Service types
  - Directory service, print service, file service, …
Client-Server and Distributed Systems

- Services may be implemented by distributed processes
  - May require distributed resources (e.g., WWW)
  - May choose to partition and distribute for reliability
  - Replication used to increase performance and availability and to improve fault tolerance
Clients Invoke Individual Servers

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
A Service Provided by Multiple Servers

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
More on Client-Server Model

- **Clients**
  - Usually blocks until server responds
  - Usually invoked by end users when they require service
  - Interacts with users through a user interface
  - Interacts with client middleware using middleware API

- **Server**
  - Implements services
  - Usually waits for incoming requests
  - Usually a program with special privileges
  - Invoked by server middleware
  - Provide error recovery and failure handling services
Software Layers

- Software architecture refers to structuring of software
  - Layers and services (service layers)
- Platform
  - Lowest-level hardware and software layers
- Middleware
  - Layer of software to mask heterogeneity and provide convenient programming model for applications
  - Provides useful building blocks to develop software
  - Raises level of communication activities through support for communication abstractions / mechanisms
  - Make distributed nature transparent
Software and Hardware Services Layers

- Applications, services
- Middleware
- Operating system
- Computer and network hardware

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Middleware Packages

- Remote procedure call (RPC)
- Group communication (Isis)
- Object-oriented
  - Common Object Request Broker Architecture (CORBA)
  - Java Remote Method Invocation (RMI)
  - Microsoft Distributed Common Object Model (DCOM)
- Packages provide higher-level applications services
  - Naming, security, transactions
  - Persistent storage, event notification
Middleware Limitations

- End-to-end argument (Saltzer, Reed, Clarke, 1984)
  - Some communication-related functions can be completely and reliably implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that function as a feature of the communication system itself is not always sensible.
- This runs counter to the view that all communication activities can be abstracted away by middleware layers
- Correct behavior in distributed programs depends upon error measures and security at all levels
- Example: fault-tolerant, reliable, end-to-end transfer
Client-Server Middleware

Client processes

Client middleware

Local services

Network services

OS and H/W

Server processes

Server middleware

Local services

Network services

OS and H/W

exchange protocol

network protocol
Functional View of Middleware

- Information exchange services

- Application-specific services
  - Specialized services
    - Example: transactional / replication services for distributed database
    - Example: groupware services for collaborative applications

- Management and support services
  - Needed for locating distributed resources
  - Administering resources across the network
Commercial Middleware

- Middleware components that provide only one service
  - Example: HTTP for retrieving remote documents
  - Example: SUN RPC for remote procedure call

- Middleware environments that combine many services
  - Integrates RPC, security, directory, time, file services
  - Example: DCE, CORBA, DCOM, .NET, Java

- Compound middleware environments that combine many middleware environments into a single framework
  - Example: transaction management + RPC/RMI
Proxy Servers and Caches

- Proxy servers used to increase availability and performance of services by reducing the load on the network and servers
  - Separation of service functionality
  - Sharing of proxy server among clients

- A cache stores recently used data objects closer to users of the objects than the actual objects themselves
  - Cache is check first when data is required
  - If present in cache (hit), data is taken from there
  - Otherwise, fetched from actual
  - Cache aid in performance improvement
Web Proxy Server

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Peer Processes

- Cooperative interaction to perform distributed activity or computation without any distinction between clients and servers
- Each machine on network has equivalent capabilities
- Code in peer processes maintains consistency of application-level resources and synchronizes application-level actions when necessary
- No machines are dedicated to service others
- Example: collection of workstations
  - File access, direct emailing, Gnutella-style content sharing, SETI@home computation
Distributed Application Based on Peer Processes

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Process Peer Model

- What about idle or lightly loaded workstations?
- Sharing of computing resources
  - Either let idle (lightly loaded) machines sit idle
  - Run jobs on unused computing resources
- Alternatively
  - Treat machines as a collection of CPUs and memory
  - Assign processes to run on resource on demand
  - Users won’t need heavy duty workstations locally
    - run GUI on local machine
    - use remote machine for heavy duty processing
- Computation model of Plan 9
Grid Computing

- Provide users with seamless access to:
  - Storage capacity
  - Computing capacity
  - Network bandwidth between storage and computing
- Heterogeneous and geographically distributed
- On demand resource allocation
- Adaptive to variations in load and reliability
- Ubiquitous
Variations on Client-Server Model

- Mobile code moves between computer systems
- Applets are well-known and widely used example
  - Example: web applets
- Help to achieve better performance
  - Does not suffer from delays or bandwidth variability associated with network communication
- Accessing services means running code that can invoke their operations
  - Pull model: client initiates
  - Push model: server initiates
Web Applets

a) client request results in the downloading of applet code

b) client interacts with the applet

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Spontaneous Networking in a Hotel

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Thin Clients and Thick Clients

☐ Thin clients
   ☐ Bulk of processing done on the server
   ☐ Little administration on client
   ☐ Reduced hardware (e.g., sometime diskless)
   ☐ Example: X terminal (X-11 windowing system)
   ☐ Examples: network computer, information appliance

☐ Thick clients
   ☐ Clients perform bulk of processing operations
   ☐ Server provides other resources and services
      ➢ files, web server, print services
   ☐ Example: conventional workstations
Thin Clients and Compute Servers

Network computer or PC

Thin Client

network

Application Process

Compute server

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Multi-Tier Client-Server Architectures

- Two-tier architecture
  - Common from mid 1980’s to early 1990’s
  - UI on user’s desktop
  - Application services on server
  - Performance deteriorates with large user communities
    - server may spend a significant amount of time managing connections
    - legacy services may have to run on environments poorly adapted for networking
    - databases are performance hogs
Three-Tier Architecture

- **Client**
  - User interface
  - Some data validation and formatting

- **Middle tier**
  - Queueing and scheduling of user requests
  - Transaction processor
  - Connection management
  - Format conversion

- **Backend server**
  - Database
  - Legacy application processing
Design Requirements

- Performance issues
  - Throughput, responsiveness, workload balance

- Quality of service
  - Non-functional properties
    - reliability, security, performance
  - Adaptability to meet changing resource availability
  - Time-critical vs. batch requirements

- Caching and replication
  - Access enhancement and validation

- Dependability
  - Fault tolerance and security
Fundamental models

Interaction model

- Performance of communication channels
- Computer clocks and timing events
- Two variants:
  - synchronous distributed systems
    - time to execute each step has known lower/upper bounds
    - each message transmitted is received in bounded time
    - each process has local clock with known relation to real
  - Asynchronous distributed systems
    - no bounds on process execution speeds
    - no bounds on message transmission delays
    - no bounds on clock drift rates
Real-Time Ordering of Events

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Processes and Channels

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
# Omission and Arbitrary Failures

<table>
<thead>
<tr>
<th>Class of failure</th>
<th>Affects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail-stop</td>
<td>Process</td>
<td>Process halts and remains halted. Other processes may detect state.</td>
</tr>
<tr>
<td>Crash</td>
<td>Process</td>
<td>Process halts and remains halted. Other processes may not detect.</td>
</tr>
<tr>
<td>Omission</td>
<td>Channel</td>
<td>A message inserted in an outgoing buffer never arrives.</td>
</tr>
<tr>
<td>Send omission</td>
<td>Process</td>
<td>A process completes a send, but message not put in buffer.</td>
</tr>
<tr>
<td>Receive omission</td>
<td>Process</td>
<td>A message is put in incoming buffer, but never received</td>
</tr>
<tr>
<td>Arbitrary (Byzantine)</td>
<td>Process or Channel</td>
<td>Process/channel exhibits arbitrary behavior.</td>
</tr>
</tbody>
</table>
## Timing Failures

<table>
<thead>
<tr>
<th>Class of failure</th>
<th>Affects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock</td>
<td>Process</td>
<td>Process’s local clock exceeds the bounds on its rate of drift from real time.</td>
</tr>
<tr>
<td>Performance</td>
<td>Process</td>
<td>Process exceeds the bounds on the interval between two steps.</td>
</tr>
<tr>
<td>Performance</td>
<td>Channel</td>
<td>A message’s transmission takes longer than the stated bound.</td>
</tr>
</tbody>
</table>
Objects and Principals

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
The Enemy

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg
Secure Channels

* Graphics from Distributed Systems: Concepts and Design, Coulouris, Dollimore, and Kindberg