System Models
Dr. Xiaobo Zhou

Adapted from Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design
Edition 4, © Addison-Wesley 2005

Architectural Models

- The architecture of a system is its structure in terms of separately specified components
- Two main issues about an architectural model
  - The placement of the components across a network of computers
    - Patterns for the distribution of data and workload
  - The inter-relationships between the components
    - Functional roles and the patterns of communications
Software and Hardware Service Layers

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

Middleware: mask heterogeneity and provide programming model; Sun RPC, CORBA, Java RMI, MS DCOM, ODP, etc.

Client-Server System Model

- Server: a process accepts requests from other processes
- Client: a process requesting services from a server
  - The terms are applied only to the roles played in a single request
  - Is a search engine server or client, or both?
A distributed application based on peer processes

Multiple-servers System Model

- Object placement
  - Partitioning:
    - DNS
  - Replication:
    - SUN NIS
Proxy Server and Caches

- **Proxy server:**
  - To increase availability and performance of the service by reducing the load on the WAN and Web servers
  - To access remote Web servers through a firewall
  - But any additional considerations / concerns? P.46

Web Client-Server Architecture
Email Client-Server Architecture

Variations on the Client-server Model: Mobile Code

a) client request results in the downloading of applet code

b) client interacts with the applet locally

c) what are main advantages and disadvantages?
Variations (2): Mobile Agents

a) client launches a mobile agent (a running program including both code and data) that travels from one computer to another in a network

b) Mobile agent interacts with the server *locally*

c) what are main advantages and disadvantages?

Variations (3): Network computes

Network computer:
- Downloads its OS and application software needed from a remote file server
- Applications are run *locally* but the files are managed by the remote file server; low software management and maintenance cost
- If disk is present, mostly used as a cache storage
Variations (4): Thin Clients and Compute Servers

Thin client:
- A software layer that supports a window-based user interface on a computer that is local to the user while executing application programs on a computer server remotely; X11 Window System
- Main drawback: poor performance in highly interactive application such as CAD and image processing

Performance Issues

Responsiveness
- Delay, response time, slowdown, stretch factor
- Determined by load and performance of the server and the network, and by delays in all software components involved

Throughput
- The rate at which computational work of the server or data transfer of the network is done

Load balancing / load sharing
- Enable applications and service processes to proceed concurrently and exploit the available resources
Quality of Service (QoS)

- Main non-functional properties of systems/networks
  - Reliability, security, and performance
  - Adaptability to system configuration and resource availability is raised recently

- Absolute QoS assurance / differentiation
  - Ability to meet timeliness deadlines
  - Hard real-time applications

- Relative QoS assurance / differentiation
  - One receives relatively, say proportionally, better services than the other

Interaction Model

- A distributed system are composed of interacting processes
  - Multiple server processes cooperate to provide a service
  - Peer processes cooperate to achieve a common goal

- Two factors affect interacting processes
  - Communication performance is often a limiting characteristic
    - Latency, bandwidth, and jitter
  - It is impossible to maintain a single global time
    - Different clock drift rate
Clocking: Synchronous and Asynchronous

**Synchronous Bus:**
- Example: Processor-memory bus
- Includes a clock in the control lines
- A fixed protocol for communication that is relative to the clock
- Advantage: involves very little logic (cheap) and can run very fast
- Disadvantages: to avoid clock skew, they cannot be long if they are fast

**Asynchronous Bus:**
- It is not clocked; I/O bus
- It can accommodate a wide range of devices
- It can be lengthened without worrying about clock skew
- It requires a handshaking protocol to coordinate/synchronize the transmission of data between sender and receiver; a series of steps, sender/receiver proceed to the next step when both agree.

---

Synchronous Distributed Systems

**Process execution time**
- The time to execute each step of a process has known lower and upper bounds

**Message delivery time**
- Each message transmitted over a channel is received within a known bounded time period

**Clock drift rate**
- Each process has a local clock whose drift rate from real time has a known bound

**Observations:**
- It is possible to suggest likely bounds, but hard to guarantee
- Time-outs can be used for failure detection
Asynchronous Distributed Systems

- There are no bounds on
  - Process execution time: each step of a process may take an arbitrarily long time
  - Message delivery time: a message may be received after an arbitrarily long time
  - Clock drift rate: a clock’s clock drift rate is arbitrary

**Observations:**
- Actual distributed systems are very often asynchronous because of the needs to share the processors and for communication channels to share the network
- Any solutions valid for synchronous also for asynchronous
- Time-outs may NOT be used for failure detection, but can be used for failure masking (failure suspect)

Real-time Ordering of Events

![Diagram of Real-time Ordering of Events](image-url)
Logical-time Event Ordering

Event: X sends m1 before Y receives m1
Event: Y sends m2 before X receives m2
Event ordering 1, 2, 3, 4

Failure Models: Process Omission Failures

In an asynchronous distributed system
- A timeout means that a process is NOT responding; may have crashed or may be slow, or the message may not have arrived

In a synchronous distributed system
- A timeout means that a process is crashed, so called fail-stop
Failure Models: Communication Omission Failures

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 not detect this state.</td>
</tr>
<tr>
<td>Crash</td>
<td>Process</td>
<td>Process halts and remains halted. Other processes may not be able to detect this state.</td>
</tr>
<tr>
<td>Omission</td>
<td>Channel</td>
<td>A message inserted in an outgoing message buffer never arrives at the other end’s incoming message buffer.</td>
</tr>
<tr>
<td>Send-omission</td>
<td>Process</td>
<td>A process completes a send, but the message is not put in its outgoing message buffer.</td>
</tr>
<tr>
<td>Receive-omission</td>
<td>Process</td>
<td>A message is put in a process’s incoming message buffer, but that process does not receive it.</td>
</tr>
<tr>
<td>Arbitrary (Byzantine)</td>
<td>Process (channel)</td>
<td>Process/channel exhibits arbitrary behaviour; it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.</td>
</tr>
</tbody>
</table>
### Timing Failures in Synchronous Systems

<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>

- Timing failures are only applicable in synchronous distributed systems
  - Time limits are set on process execution time, message delivery time, and clock drift rate

### Failure Masking

- To construct reliable services from component that exhibit failures
  - **Hiding a failure:**
    - Multiple servers hold replicas of data can continue to provide a service when one of them crashes
  - **Converting it to a more acceptable type of failure:**
    - Use checksums to mask corrupted messages, converting from arbitrary failure into an omission failure
    - An omission failure can further be hidden by using a protocol that retransmit messages that do not arrive at their destination
Reliability of One-to-One Communication

*reliable 1-1 communication* is defined in terms of *validity* and *integrity*:

**validity:**
- any message in the outgoing message buffer is eventually delivered to the incoming message buffer;

**integrity:**
- the message received is identical to one sent, and no messages are delivered twice.

How do we achieve validity?

*validity* - by use of acknowledgements and retries

How do we achieve integrity?

*integrity* -
- by use of checksums, reject duplicates (e.g. due to retries; sequence numbers).
- *If* allowing for malicious users, use security techniques

Security Model: Objects and Principals

*Access rights:* specify who is allowed to perform the operations of an object

*Principals:* the authorities which issue the invocation and result
The Enemy/Adversary

- Threats to processes:
  - Servers: determine the identity of the principal behind an invocation
  - Clients: tell the resource of the returned result

- Threats to communication:
  - An enemy can copy, alter, or inject messages in the networks

Secure Channels

- Cryptography, shared secrets, and encryption
- Authentication proves the identities supplied by the senders
- Secure channels: a server layer on top of existing communication services