Slides for Chapter 2: Architectural Models

From Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design
Edition 4, © Pearson Education 2005

Software Layers

- Platform: hardware, OS
- Middleware
  - masking heterogeneity of underlying platform, hence providing a common "platform"
  - Examples:
    - Sun RPC, Java RMI
    - CORBA, Microsoft .NET, Java J2EE
  - reliable services in the middle layer, still need application-specific reliability
- Applications, services

Client-Server

- Servers provide services
- Clients access services

Peer to Peer

- Peers play similar roles
- No distinction of responsibilities

A service provided by multiple servers

- Objects are partitioned/replicated
  - Web: each server manages its objects
  - NIS: replicated login/password info
  - Cluster: closely coupled, scalable (search engines)

Web proxy servers and caches

- Cache: local copies of remote objects for faster access
- Browser cache
- Proxy server
  - Additional roles: filtering, firewall
Mobile code and web applets

- Running code locally vs. remotely
  - Network bandwidth
  - What examples have you seen?
  - Security issues?
    a) Client request results in the downloading of applet code
    b) Client interacts with the applet

![Diagram showing client, web server, applet code, and web server]

Mobile agents

- Running program that travels from one computer to another
  - Perform tasks on users’ behalf
  - Security issues
  - How does it compare to one client interacting with multiple servers?
    - What if the program needs to access a lot of remote data?
  - How does it compare with applets?
    - Agents can provide functionality a remote web site does not provide, but allows access to data.

Thin clients and compute servers (network computers)

- Thin client
  - Basically a display with keyboard (“dumb terminal”)
  - Remote computation, storage
- How does it compare with the
  - PC model or
  - distributed workstation model?

![Diagram showing network computer or PC, network, compute server, and application process]

Mobile devices and spontaneous interoperation

- Laptops, PDA’s, cell phones, wearable computers...
- Metropolitan (GSM, CDPD): hundreds of meters, 100s Kb/s
- Local Area (BlueTooth, infra-red): meters, 10 Mb/s
- Infrastructure vs ad-hoc networking
- Accessing services, variable bandwidth/connectivity, power supply, security,
- Spontaneous interoperation—easy connection and location of services
  - Service discovery
  - Context-aware

Interfaces and objects

- At the programming level
  - Need standard interfaces
    - Access/provide services/objects
    - RPC/RMI

Design requirements for distributed architectures

- Performance
  - Responsiveness, throughput, load balancing
- Quality of service (QoS)
  - Time-critical applications (real-time apps)
  - Guarantee certain level of quality delivered by the deadline
  - Allocation of computation and communication resources
- Caching and replication
  - Web caching: server provides expiration time
- Dependability
  - Fault tolerance: redundancy, recovery
  - Security
**Fundamental Models**

- Fundamental properties in processes and communication, shared among different architectures discussed previously

**Interaction model (1)**

- Sequential vs. distributed algorithms timing, distributed state
- Performance of communication channels
  - Latency: transmission, access, os
  - Bandwidth
  - Jitter: variation among messages
- Clocks and timing events
  - Clock drift
  - Synchronization

**Interaction Model (2) [Synchronous vs Asynchronous]**

- Synchronous distributed systems
  - Lower and upper bounds for execution of a step
  - Message transmission in bounded time
  - Clock drift rate is bounded
  - Failures can be detected when bounds are exceeded
  - Accomplished by allocating sufficient resources
- Asynchronous distributed systems
  - No bounds on process speed, message delay, clock drift rate
  - Failures are harder to detect
  - Performance can’t be guaranteed

**Interaction Model (3)**

- Event ordering
  - Relative ordering might be more important than exact time
  - Logical clocks—ordering events without physical clocks

**Failure Model (1)**

- Failure of processes or communication channels

**Failure Model (2): 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 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 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 the process does not receive it.</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>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>
### Failure Model (3): 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>

### Failure Model (4)

- **masking failures**
  - hiding—use another server to respond
  - converting it into more acceptable—drop the packet if it is corrupted

- **reliable one-to-one communication**
  - validity: eventually delivered
  - integrity: content not corrupted or duplicated

### Security Model (1)

- **Protecting objects:**
  - authorization (access rights to principals)
  - authentication (identity of parties/principals)

### Security Model (2)

- **Threat to processes & communication channels**

### Security Model (3)

- **Threat to processes & communication channels**
- **Denial of service**
- **Mobile code**
- **Cryptography:** science of keeping messages secret
  - encryption: process of scrambling a message to hide its content
  - secret keys—large numbers that are difficult to guess
  - authentication—encrypt the identity, check the decrypted identity
  - secured channels—authentication, privacy/integrity, time stamp to prevent replaying and reordering