Distributed Systems

a.y. 2023/2024
Models for Distributed Systems
Models ...

- Shared properties and common design problems can be represented in the form of descriptive models.

- Each model is intended to provide an abstract, simplified but consistent description of a relevant aspect of the design.
...two types of models...

- Architectural models
- Fundamentals models:
  - interaction
  - failure
  - security

Remember ... No global time message-based communication
Models for Distributed Systems: Architectures
Architecture

Concerned with:
- the placement of parts
- the relationships between them
  (software architecture)
- The mapping onto the underlying network of computers
  (system architecture)
- Processes and objects
Architectural Styles

- A component is a modular unit with well-defined required and provided interface that is replaceable within its environment.
- A connector is a mechanism that mediates communication, coordination, cooperation among components.
Architectural Styles

- Layered architectures
- Object-based architectures
- Data-centered architectures
- Event-based architectures
…layered…
...object-based...
...data centered...

- A common repository
- Communication via a shared space
- Blackboard
- Shared file system/Shared data center
...event-based...
...shared data-space
Process taxonomy

- Server process ...
- Client process ...
- Peer process ...
- Moving process ...
System architectures

- Software layers

- Architectural models
  - client-server, peer processes,…
  - mobile code, agents,…
Software Layers

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

- **applications**
- **Open (distributed) services**
- **Middleware**
- **Operating system**
- **Platform**
- **Computer and network hardware**
Software layers

- Service layers
- Higher-level access services at lower layers
- Services can be located on different computers
Important layers

• Platform
  – lowest-level hardware+software
  – common programming interface, yet
  – different implementations of operating system facilities for co-ordination & communication

• Middleware
  – programming support for distributed computing
...middleware provides...

support for distributed processes/objects:
  – suitable for applications programming
  – communication via
    remote method invocation (Java RMI), or
    remote procedure call (Sun RPC)

services infrastructure for application programs
  – naming, security, transactions, event notification,
Support a higher level of abstraction for:

- Communication between group of processes
- Notification of events
- Replication of shared data
- Multimedia data transmission in real time
- Object Management Group’s Common Object Request Broker Architecture (CORBA)

- Microsoft Distributed Component Object Model (DCOM)

- Java RMI
The layered view...

- though appropriate for simple types of resource data sharing:
  - e.g. databases of names/addresses/exam grades

- too restrictive for more complex functions?
  - reliability, security, fault-tolerance, etc, need access to application’s data

- see end-to-end argument [Saltzer, Reed & Clarke]
“Some communication-related functions can be completely and reliably implemented only with the knowledge and help of the application standing at the end points of the communication systems”

Checks, error-correction mechanism and security are at many levels ...

Checking the correctness within the communication systems could not be enough
Architectural models

- Define
  - software components (processes, objects)
  - ways in which components interact
  - mapping of components onto the underlying network

- Why needed?
  - to handle varying environments and usage
  - to guarantee performance
Main types of models...

- **Client-server**
  - first and most commonly used

- **Multiple servers**
  - to improve performance and reliability

- **Proxy servers**
  - to reduce load on network, provide access through firewall

- **Peer processes**
  - faster interactive response
Server1 acts as **client** for Server2
Main types of models:

- **Client-server**
  - first and most commonly used

- **Multiple servers**
  - to improve performance and reliability

- **Proxy servers**
  - to reduce load on network, provide access through firewall

- **Peer processes**
  - faster interactive response
Centralized Architectures

- Request-Reply Behaviour
Client-Server Systems

One Tier Architecture

Network computer or PCs with terminal emulation

Presentation (to clients)
+ processing (transactions, applications)
+ data (management & access)

Two Tier Architecture

workstations

Presentation + processing ↔ Data (remote data access)
or presentation ↔ Data processing (remote procedure call)

Client-server “fat client” or “fat server”
Three Tier Architecture

Two tier is satisfactory for simple client-server applications, but for more demanding transaction processing applications*....

clients

shared application services

remote data access, procedure call

remote data access or transaction processing

presentation

processing

data

shared data services
…Application Layering…

- The user-interface level
- The processing level
- The data level
...Application Layering...

User interface

Keyword expression

Query generator

HTML page containing list

HTML generator

Ranked list of page titles

Ranking algorithm

Web page titles with meta-information

Database with Web pages

User-interface level

Processing level

Data level
The simplest organization is to have only two types of machines:

- A client machine containing only the programs implementing (part of) the user-interface level
- A server machine containing the rest, i.e., the programs implementing the processing and data level
...Multitiered Architectures...
Figure 2. An example of a server acting as client.

User interface (presentation)

Application server

Database server

Wait for result

Request operation

Wait for data

Request data

Return result

Return data

Time
Multiple servers

Servers may interact
Proxy servers

![Diagram of proxy servers](image)

- Client
- Web server

intranet firewall outside world
Peer processes

Application
Co-ordination code

Application
Co-ordination code

Application
Co-ordination code

‘White-board’
(event notification)
A distributed application based on peer processes
- Vertical distribution
- Horizontal distribution
- Peer-to-peer systems
- Overlay networks
- DHT…Distributed Hash Table
- Membership management
Structured Peer-to-Peer Architectures

![Diagram of a structured peer-to-peer network with nodes and associated data keys.](image)

- Actual node
- Associated data keys
  - Node 12: {8,9,10,11,12}
  - Node 1: {0,1}
  - Node 2: {2,3,4}
  - Node 3
  - Node 4
  - Node 5
  - Node 6
  - Node 7
  - Node 8
  - Node 9
  - Node 10
  - Node 11
  - Node 13
  - Node 14: {13,14,15}
  - Node 15

The diagram shows a circular network with nodes connected in a specific order, illustrating the structure and connectivity of the peer-to-peer system.
Structured Peer-to-Peer Architectures

![Diagram of structured peer-to-peer architectures]

- Keys associated with node at (0.6, 0.7)
- Actual node
- Points at (0.2, 0.8), (0.6, 0.7), (0.2, 0.3), (0.9, 0.9), (0.9, 0.6), (0.7, 0.2)
Structured Peer-to-Peer Architectures
Unstructured Peer-to-Peer Architectures

- Random graph
- Flooding
Topology Management of Overlay Networks

Structured overlay

Protocol for specific overlay

Random peer

Random overlay

Protocol for randomized view

Links to topology-specific other nodes

Links to randomly chosen other nodes
Actions by active thread (periodically repeated):

select a peer P from the current partial view;
if PUSH_MODE {
    mybuffer = [(MyAddress, 0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
} else {
    send trigger to P;
}
if PULL_MODE {
    receive P’s buffer;
}
construct a new partial view from the current one and P’s buffer;
increment the age of every entry in the new partial view;

(a)
Actions by passive thread:

receive buffer from any process Q;
if PULL_MODE {
    mybuffer = [(MyAddress, 0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
}
construct a new partial view from the current one and P’s buffer;
increment the age of every entry in the new partial view;

(b)
Topology Management of Overlay Networks
Superpeers
Edge-Server Systems

Figure 2-13: Viewing the Internet as consisting of a collection of edge servers.
Client server and mobility

- Mobile code
  - downloaded from server, runs on locally
  - e.g. web applets
Web applets

Client requests results, applet code is downloaded:

Client interacts with the applet:
...mobility...

- Mobile agent (code + data + state)
  - travels from computer to another
  - collects information, returning to origin.

- Mobile devices forming “spontaneous networks”
Spontaneous networks ...

- Easy connection to a LAN
- Easy integration with local services
- Limited connectivity
- Security and privacy
- Discovery service=registration+lookup
Distributed Systems:

Models for Distributed Systems: Fundamentals Models
Fundamental models

- Interaction model ... it reflects communication delays and limited accuracy due to local timing

- Failure model ... at processing and communication level

- Security model ...
Interaction model

It deals with communication and coordination

Distributed Algorithms ...

No matter how communication channels are realized ...

- Latency
- Bandwidth
- Jitter

Clock drift
Synchronous distributed systems
- Clock drift
- Execution step
- Message transmission time

Asynchronous distributed systems

Agreement problem
…Event ordering…

- Occurred before
- Occurred after
- Concurrent

- A message is received after it is sent and replies are sent after receiving messages
...real-time ordering of events...
To provide an understanding of the effects of failures .... DSs expected to continue if failure has occurred:

- Omission failures…fail to perform actions
  - Process omission failures ... A stop, A crash ... A clean crash ...
- Fail-stop
- Crash
- Arbitrary
...processes and channels...

process $p$

$send \ m$

Communication channel

Outgoing message buffer

Incoming message buffer

process $q$

receive
Communication omission failures

- Send-omission
- Receive-omission
- Channel-omission: the communication channel does not transport a message from the out buffer to the in buffer
Failure issues: …types of failures…

- benign failures (omission, stopping, timing/performance)
- arbitrary (called Byzantine)
  - corrupt message, wrong method called, wrong result
## 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 a <code>send</code>, 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</td>
<td>Process or 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

<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>
Security models

- Protecting objects ... Access rights
- Securing processes and their interaction
  - Enemy (adversary) modeling
Objects and principals

Client
Principal (user)

invocation

result

Network

Server
Principal (server)

Object
Access rights
…the enemy…
- Cryptography and shared secrets
- Authentication
- Secure channels
...secure channels...
...Distributed Systems...

end of lectures

References: