Chapter 2  System Models

The architecture of a system is its structure in terms of separately specified components. The overall goal is to ensure that the structure will meet present and likely future demands on it.

An architectural model of a distributed system first simplifies and abstracts the functions of the individual components of a distributed system and then it considers:
- the placement of the components across a network of computers — seeking to define useful patterns for the distribution of data and workload
- the interrelationships between the components — that is, their functional roles and the patterns of communication between them.

- Processes
  - server processes
  - client processes
  - peer processes

- Software architecture
  Originally: to the structuring of software as layers or modules in a single computer
  Recently: in terms of services offered and requested between processes located in the same or
different computers. This process- and service-oriented view can be expressed in terms of service layers.

- Software and hardware service layers in distributed systems

```
+-----------------+       +-----------------+
| Applications, Services | --- | Operating system |
+-----------------+       +-----------------+
          +-----------------+   +-----------------+
          | Middleware      | --- | Platform        |
          +-----------------+   +-----------------+
          | Computer and network hardware |
```

* Platform: the lowest-level hardware and software layers. These low-level layers provide services to the layers above them, which are implemented independently in each, e.g., Sun SPARC/SunOS, Intel x86/Windows.

* Middleware: a layer of software whose purpose is to mask heterogeneity and to provide a convenient programming model to application programmers.
early stage: Sun RPC (Remote Procedure Calls)
current stage:
1. Object Management Group's Common Object Request Broker Architecture (CORBA)
2. Java Remote Object Invocation (RMI).
3. Microsoft's Distributed Common Object Model (DCOM)

Middleware can also provide services for use by application programs. For example, CORBA offers a variety of services that provide applications with facilities which include:
- naming, security, transactions, persistent storage and event notification.

Limitations of middleware:
Much has been achieved in simplifying the programming of distributed systems through the development of middleware support, but some aspects of dependability of systems require support at the application level.

E.g. transfer of large email messages from the mail host of the sender to that of the recipient. TCP provides some error detection and correction, but it cannot recover from
Major network interruptions.

The mail transfer service adds another level of fault tolerance, maintaining a record of progress and resuming transmission using a new TCP connection if the original one breaks.

* System architectures
  1. Client - Server model

* The simple structure in which client processes interact with individual server processes in separate host computers in order to access the shared resources that they manage.
  * Servers may in turn be clients of other servers, e.g. a web server is often a client of a local file server that manages the files in which the web pages are stored.
Services provided by multiple servers

Services are implemented as several server processes in separate host computers interacting as necessary to provide a service to client processes. e.g. the Web provides a common example of partitioned data in which each web server manages its own set of resources. A user can employ a browser to access a resource at any one of the servers.
3. Proxy servers and caches.

Cache: a store of recently used data objects that is closer.

Web proxy servers provide a shared cache of web resources for the client machines at a site or across several sites. The purpose of proxy servers is to increase availability and performance of the service by reducing the load on the wide-area network and web servers.
all of the processes play similar roles, interacting cooperatively as peers to perform a distributed activity or computation without any distinction between clients and servers.

elimination of server processes
- reduces inter-process communication delays for access to local objects.
Variations on the client-server model:
- the use of mobile code and mobile agents;
- users' need for low-cost computers with limited hardware resources that are simple to manage;
- the requirement to add and remove mobile devices in a convenient manner.

Mobile code:
- example: Applets

Web applets
a) client request results in the downloading of applet code.

b) client interacts with the applet

Good response time
Mobile agents:
- a mobile agent is a running program (including both code and data) that travels from one computer to another in a network, carrying out a task on someone's behalf, such as collecting information, eventually returning with the results.

- a potential security threat to the resources in computers that they visit.

NetworkComputers:
It downloads its operating system and any application software needed by the user from a remote file server. Since all the application data and code is stored by a file server, the users may migrate from one network computer to another. The processor and memory capacity of a network computer can be constrained in order to reduced its cost.
Thin clients

- 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 remote computer.

Thin clients and Compute Server

Network Computer

or

PC

Compute Server

Mobile devices and spontaneous networking:

The form of distribution that integrates mobile devices and other devices into a given network is perhaps best described by the term *spontaneous networking*.

The key features of spontaneous networking are:

- Easy connection to a local network.
- Wireless links avoid the need for pre-installed cabling and avoid the inconvenience and reliability issues surrounding plugs and sockets.
Easy integration with local services:

Devises that find themselves inserted into (and moving between) existing networks of devices discover automatically what services are provided there, with no special configuration actions by the user.

Problems:

- Limited connectivity:
  Users are not always connected as they move around.
  - on a train through tunnels.

Security and privacy:

Discovery services:

- The purpose of discovery service is to accept and store details of services that are available on the network and to respond to queries from clients about them.

Discovery services offer two interfaces:

- A registration service accepts registration requests from servers and records the details that they contain in the discovery service's database of currently available services.
A lookup service accepts queries concerning available services and searches its database for registered services that match the queries. The result returned includes sufficient details to enable clients to select between several similar services based on their attributes and to make a connection to one or more of them.
Fundamental models

A model contains only the essential ingredients that we need to consider in order to understand and reason about some aspects of a system's behavior. A system model has to address the questions:

• What are the main entities in the system?
• How do they interact?
• What are the characteristics that affect their individual and collective behavior?

The purpose of a model is:

• To make explicit all the relevant assumptions about the systems we are modeling.
• To make generalizations concerning what is possible or impossible, given these assumptions. The generalizations may take the form of general-purpose algorithms or desirable properties that are guaranteed. The guarantees are dependent on logical analysis and, where appropriate, mathematical proof.
The aspects of distributed system that we wish to capture in our fundamental models are intended to help us to discuss and reason about:

1. Interaction
2. Failure
3. Security

Interaction model:
- two significant factors affecting interacting processes in a distributed system
  - Communication performance is often a limiting characteristic;
  - it is impossible to maintain a single global notion of time.

Two variants of the interaction model:

0. synchronous distributed systems

Three bounds:
1. the time to execute each step of a process
   has known lower and upper bounds;
2. Each message transmitted over a channel
   is received within a known bounded time;
3. Each process has a local clock whose
   drift rate from real time has a known bound.
Asynchronous distributed systems:
e.g. Internet

- no bounds on:
  - process execution speeds
  - message transmission delays
  - clock drift rates

Event ordering

Example: Consider the following set of exchanges between a group of email users X, Y, Z and A on a mailing list:

1. User X sends a message with the subject Meeting
2. User Y and Z reply by sending a message with the subject Re: Meeting

4. X sends message
3. Y reads it and replies
4. Z reads both X’s message and Y’s reply and then sends another reply.
Due to the delay in message delivery, and some users may view these two messages in the wrong order. For example, use A might see.

<table>
<thead>
<tr>
<th>Item</th>
<th>From</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Z</td>
<td>Re: Meeting</td>
</tr>
<tr>
<td>24</td>
<td>X</td>
<td>Meeting</td>
</tr>
<tr>
<td>25</td>
<td>Y</td>
<td>Re: Meeting</td>
</tr>
</tbody>
</table>

Real-time ordering of events:

- Send from X at $t_1$
- Send from Y at $t_2$
- Send from Z at $t_3$
- Receive at $t_4$

Timeline with physical time.
Solution: If the clocks on X's, Y's and Z's computers could be synchronized, then each message could carry the time on the local computer's clock when it was sent.

Messages m₁, m₂ and m₃ could carry times t₁, t₂, and t₃, where t₁ < t₂ < t₃.

Problem: Clocks cannot be synchronized perfectly across a distributed system.

2) Logical time:
   e.g. State a logical ordering for pairs of events
   
   X sends m₁ before Y receives m₁,
   Y sends m₂ before X receives m₂,
   Y receives m₁ before sending m₂.

Logical time takes this idea further by assigning a number to each event corresponding to its logical ordering, so that later events have higher numbers than earlier ones.
- Failure model.
  1. Omission failures: a process or communication channel fails to perform actions that it is supposed to do.
  2. Can be hidden.
  3. Arbitrary failures: describe the worst possible failure semantics, in which any type of errors may occur.
     e.g. a process may set wrong values in its data items, or it may return a wrong value in response to an invocation.

- Security model.
  1. Protecting objects
  2. Securing processes and their interactions
  3. The enemy:
     an enemy is capable of sending any message to any process and reading or copying any message between a pair of processes.
- The threats from a potential enemy
  1. threats to processes
  2. threats to Communication Channels
  3. denial of service.

- The uses of security models
  problem: the use of Security techniques such as encryption and access control incurs substantial processing and management costs.
  The security model outlined above provides the basis for the analysis and design of secure systems in which these costs are kept to a minimum.
Communication and Computer Networks