Chapter 1  Characterization of Distributed Systems

A distributed system is defined as one in which components at networked computers communicate and coordinate their actions only by passing messages. This definition allows for concurrent execution of programs, but prevents the possibility of a global clock and means that components can fail independently of one another.

The motivation for constructing and using distributed systems stems from a desire to share resources. The term 'resource' is a rather abstract one, but it best characterizes the range of things that can be usefully be shared in a networked computer system:

- hardware components
  - disks and printers
- software-defined entities
  - files, database and data objects of all kinds
- stream of video frames
- audio connection that a mobile phone can represent.
- Significant consequences for distributed systems.
  - Concurrency:
    concurrent program execution.
    The coordination of concurrently executing programs that share resources is also an important and recurring topic.

- No global clock:
  there is no single global notion of the correct time. This is direct consequence of the fact that the only communication is by sending messages through a network.

- Independent failures:
  Each component of the system can fail independently, leaving the others still running.
- Examples of distributed systems
  1. The Internet
  2. Intranets
  3. Mobile and Ubiquitous computing

- Internet
  - The Internet is a very large distributed system that allows users throughout the world to make use of its services.
  - The Internet is a vast interconnected collection of computer networks of many different types.
The figure shows a collection of intranets—subnetworks operated by companies and other organizations. Internet service providers (ISPs) are companies that provide modem links and other types of connection to individual users and small organizations, enabling them to access services anywhere in the Internet as well as providing local services such as e-mail and web hosting.

The intranets are linked together by backbones, backbone: a network link with a high transmission capacity, employing satellite connections, fibre optic cables and other high-bandwidth circuits.

- Can provide multimedia services
  - access audio and video data
    - music, radio and TV channels and to hold phone and video conferences
  - problem: it does not provide the necessary facilities to reserve network capacity for individual streams of data.
Intranets

An intranet is a part of the Internet that is separately administered and uses a firewall to enforce its own local security policies. Users in an intranet share data by means of file services.

A typical intranet

- It is composed of several networks (LANs) linked by backbone connections.
- Configuration is the responsibility of the organization.
  - a LAN on a single site
  - a connected set of LANs belonging to branches of a company
- An intranet is connected to the Internet via a router.
- A firewall is to protect an Internet by preventing unauthorized messages from leaving or entering.
- A firewall is implemented by filtering incoming and outgoing messages.
- Some organizations do not wish to connect their internal networks to the Internet.

- Mobile and ubiquitous computing.
  Distributed systems techniques are equally applicable to mobile computing (involving laptops, PDAs (personal digital assistants), and wearable computing devices) as well as ubiquitous computing (involving small computers embedded in appliances — washing machines, cars, and refrigerators).

- Mobile computing (also called nomadic computing) users who are away from their "home" Internet are still provided with access to resources via the devices they carry with them. There is increasing provision for users to utilize resources such as printers that are conveniently nearby as they move around — location-aware computing.
- Ubiquitous Computing
  many small, cheap computational devices that are present in users' physical environments, including the home, office and elsewhere.

- Portable and handheld devices in a distributed system.

The users have access to three forms of wireless connection:
* Laptop connects to the host's wireless LAN (a few hundreds of meters) - one floor.
  It connects to the rest of the host intranet via a gateway.
* Mobile telephone connects to the Internet using the Wireless Application Protocol (WAP) via a gateway.
• Digital camera can communicate over a link when pointed at a corresponding device such as a printer.

• Shared resources are managed by server processes, which provide client processes with access to these resources via a well-defined set of operations. In a distributed system written in an object-oriented language, resources may be encapsulated as objects whose methods are invoked by client objects.

  - Web browser is an example of a client. The Web browser communicates with a Web server, to request Web pages from it.
• Challenges
  • Heterogeneity
    Heterogeneity applies to all of the following
    • networks
    • computer hardware
    • operating systems
    • programming languages
    • implementations by different developers

Middleware: a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying network hardware, operating systems, and programming languages.

CORBA, RMI, Java RMI

• Openness
  • The openness of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways.
  • The openness of a distributed system is determined primarily by the degree to which new resource-sharing services can be added and be
made available for use by a variety of client programs.

- Open systems are characterized by the fact that their key interfaces are published.
- Open distributed systems are based on the provision of a uniform communication mechanism and published interfaces for access to shared resources.
- Open distributed systems can be constructed from heterogeneous hardware and software, possibly from different vendors. Each component of the published standard must be carefully tested and verified if the system is to work correctly.

- Security
  Security for information resources has three components:
  1. Confidentiality: protection against disclosure to unauthorized individuals
  2. Integrity: protection against alteration or corruption
  3. Availability: protection against interference with the means to access the resources.
* Encryption techniques can be used for this purpose.

Two security challenges have not yet been fully met:

1. Denial of service attacks:
   Send a large number of pointless requests that the serious users are unable to use it.

2. Security of mobile code:
   Mobile code needs to handled with care, e.g. an executable program as an electronic mail attachment.

* Scalability:

A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users.

Challenges:

1. Controlling the cost of physical resources:
   As the demand for a resource grows, it should be possible to extend the system, at reasonable cost, to meet it.
e.g. if a single file server can support 20 users, then two such servers should be able to support 40 users. — difficult

2) Controlling the performance loss:
   e.g. the table with the correspondence between the domain names of computers and their Internet addresses held by the Domain Name System.
   - hierarchic structures scale better than those that use linear structures.
   - an increase in size will result in some loss in performance even using the hierarchic structure.
   - need to minimize the loss in performance.

3) Preventing software resources running out:
   e.g. numbers used as Internet address
   Now: 32 bits IPv4
   New: 128 bits IPv6
4. Avoiding performance bottlenecks:
   Algorithms should be decentralized to avoid having performance bottlenecks.

   Possible techniques that have been successful to handle the scalability:
   1. Replicated data
   2. Caching
   3. Multiple servers to handle commonly performed tasks.
   4. Enabling several

• Failure handling:
  1. Detecting failures
e.g. Checksum

  The challenge is to manage in the presence of failures that cannot be detected but may be suspected.
e.g. Remote crashed server in the Internet

  2. Masking failures:
  Some failures that have been detected can be hidden or made less severe.
examples:
1. Messages can be retransmitted when they fail to arrive.
2. File data can be written to a pair of disks so that if one is corrupted, the other may still be correct.

3. Tolerating failures:
   e.g. When a Web browser cannot contact a Web server, it informs the user about the problem, leaving them free to try again later.

4. Recovery from failures:
   Design of software so that the state of permanent data can be recovered or 'rolled back' after a server has crashed.

5. Redundancy:
   Services can be made to tolerate failures by the use of redundant components.
• Concurrency
  This can be achieved by standard techniques such as semaphores.

• Transparency
  1. Access transparency: enables local and remote resources to be accessed using identical operations.
  2. Location transparency: enables resources to be accessed without knowledge of their location.
  3. Concurrency transparency: enables several processes to operate concurrently using shared resources without interface between them.
  4. Replication transparency: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.

• Failure transparency: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
6. Mobility transparency: allows the movement of resources and clients within a system without affecting the operation of users or programs.

7. Performance transparency: allows the system to be reconfigured to improve performance as loads vary.

8. Scaling transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms.