Distributed Systems

Unit – I

<u>Introduction</u>: Now a days networks of computers are used in everywhare. The Internet is one, it provides the information of world as web or village. It means, know the information within fracation of seconds which we require. Different networks are composed and they used the internet. For example, Mobile phone networks, corporate networks, factory networks, campus networks, home networks, in-car networks etc. All of these networks using '*Distributed Systems*' technology to communicate and coordinate their actions only by passing messages.

DS = N/W +Distributed Architecture +Distributed OS +Distributed Application.

<u>Definition</u>:*A distributed system is one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.*

(or)

"A distributed system is a collection of independent computers that appear to the users of the system as a single computer."

This definition covers the entire range of systems in which networked computers can usefully be deployed.

Characterization of Distributed Systems:

Computers that are connected by a network may be spatially separated by any distance. They may be on separate continents in the same building or in the same room. Our definition of distributed Systems have the following Characteristics of Distributed Systems or Significant Consequences:

1.<u>Concurrency</u>: In a network of computers, concurrent program execution is common. It means, " *I can do my work on my computer while you can do your work on your computer*", sharing resources of both work (such as web pages on files) when necessary. The capacity of sharing can be increased by adding more resources (for example computers) to the network.

2. <u>No global clock</u>: Programs coordinate actions by exchanging messages in distributed systems. Close coordination often depends on a shared idea of the time at which the programs' actions occur. But there are limits to the accuracy with which the computers in a network can synchronize their clocks. In this case, "*there is no single global notion of the correct time*". This is a direct consequence of the fact that the only communication is by sending messages through network.

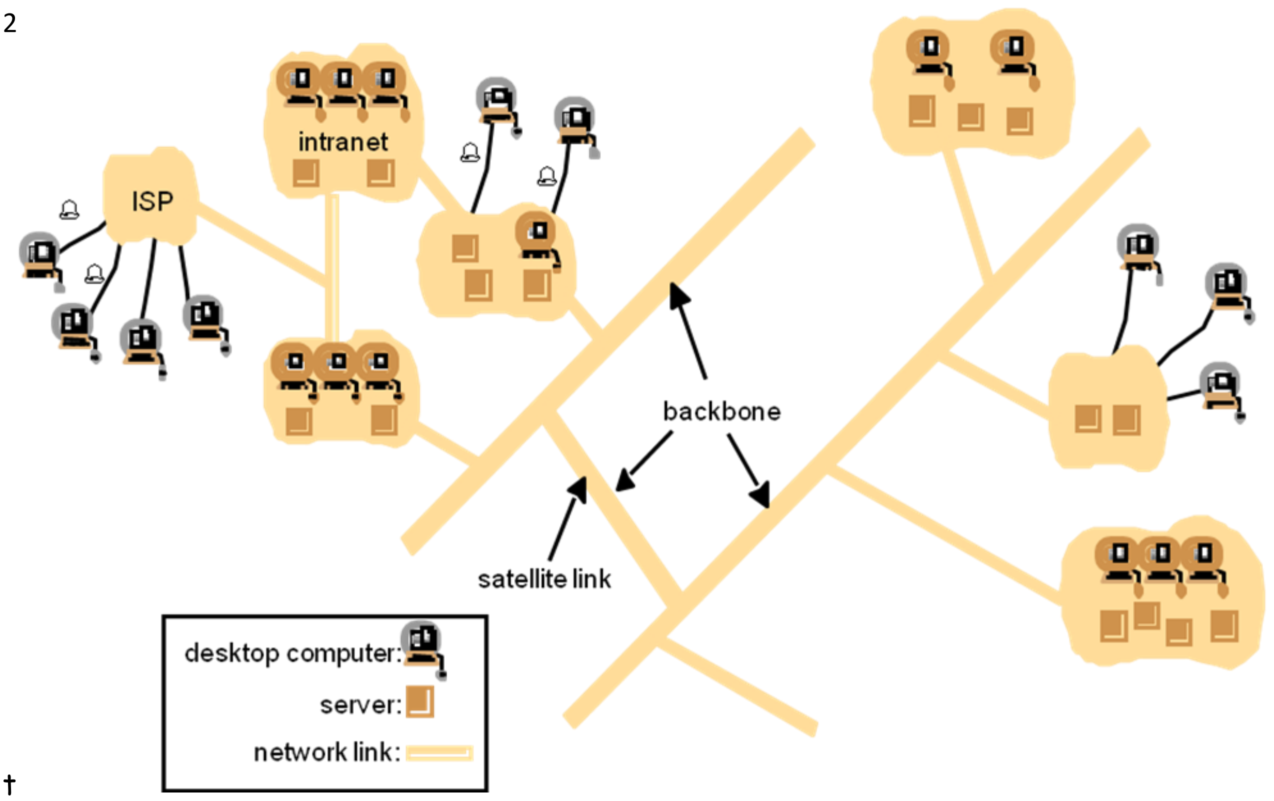
3. <u>Independent failures:</u> Every system can fial based on the responsibility of system designers. But Distributed systems can fail in new ways. Faults in the network result in the isolation of the computers that are connected it, but they doesn't mean stop running. In fact, the programs on them may not be able to detect whether the network has failed or has become unusually slow. Similarly, the failure of a computer, or unexpected termination of a program by a crash is not effect to other computer in distributed networks.

when some systems fail, others may not know and other system will provide the service goal of the distributed system is sharing the resource.

<u>Examples of Distributed Systems</u>: Examples of distributed systems which are familiar and widely used networks. The Internet, Intranet, Mobile Computing and Mobile and ubiquiting computing.

1. **<u>Internet</u>**: It is a very large distributed system that allows users throughout the world to make use of its services. For example, World Wide Web, web search, online gaming, email, social networks, ecommerce, etc.
    - WWW, email, FTP, VOD, etc

- Internet protocols is a major technical achievement.
  - TCP/IP: connect different type computer networks.



t

The figure illustrates a typical portion of the Internet. Programms running on the computers connected to it interact by passing messages in communication. The design and construction of the Internet communication mechanism (the Internet protocols) is a major technical achievement, enabling a program running anywhere to address messages to programs anywhere else.

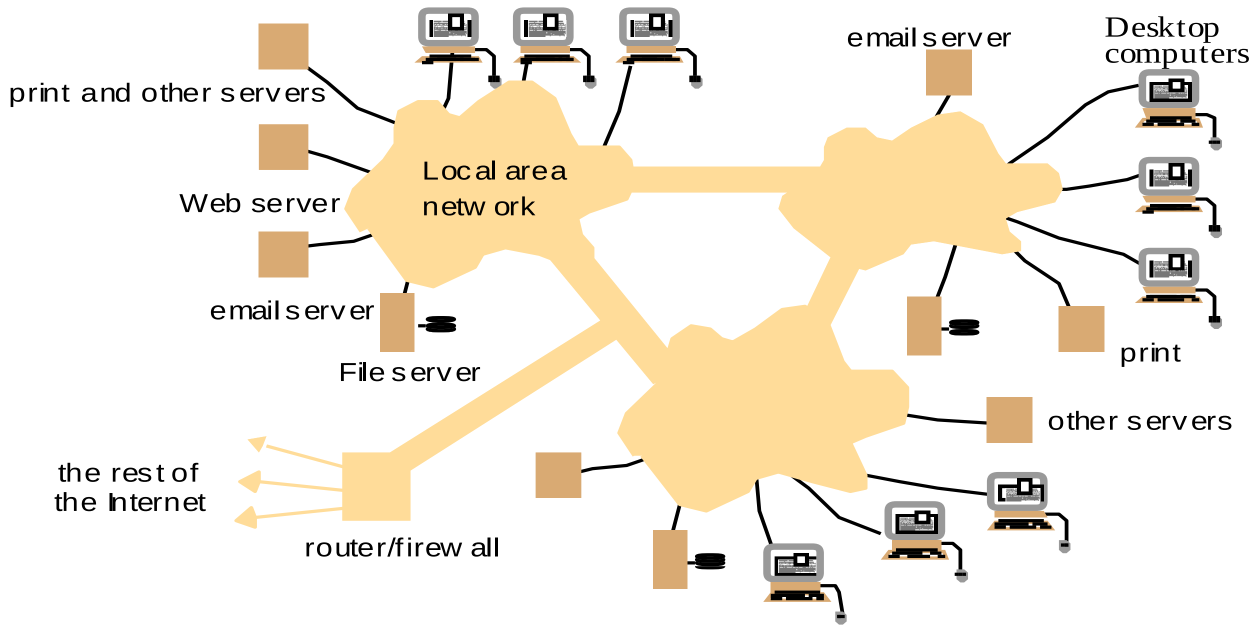
Internet is also a very large distributed system. It enables users, wherever they are, to make use of services such as the World Wide Web, email and file transfer.

The figure shows a collection of intranets that subnetworks operated by compaines 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 Intranets are linked together by backbones.

A backbone is a network link with a high transmission capacity, employing satellite connections, fiber optic cables and other high-bandwidth circuits.

2. **Intranets:** An intranet is a portion of the internet that is separately administered and has a boundary that can be configured to enforce local security policies.
The figure shows a typical intranet. It is composed of several local area networks (LANs) linked by backbone connections. The network configuration of a particular intranet is the responsibility of the organization that administers it and may vary widely.



An intranet is connected to the Internet via a router, which allows the users inside the intranet to make use of services elswhere such as the Web or email. It also allows the users in other intranets to access the services it provides. Many organizations need to protect their own serives from unauthoriezed use by possibly malicious users elsewhere.

For example, a company will not want secure information to be accessible to users in competing organizations, and a hospital will not want sensitive patient data to be revealed. Companies also want to protect themselves from harmful programs such as viruses entering and attacking the computers in the intranet and possibly destroying valuable data.

The role of a *firewall* is to protect an intranet by preventing unauthorized messages leaving or entering. A firewall is implemented by filtering incoming and outgoing messages, for example according to their source or destination. A firewall might for example allow only those messages releated to email and web access to pass into or out of the intranet that it protects.

The main issues arising in the design of components for use in intranets are:

- File services are needed to enable users to share data.
- Firewalls tend to impede legitimate access to services
- The cost of software installation and support is an important issue.

3. **Mobile and Ubiquitous Computing:** Due to the technological advances in device efficiency and wireless networking have been increased to the integration of small and portable computing into distributed systems.

These devices include:

- Laptop computers.
- Handheld devices, including mobile phones, smart phones, GPS-enabled devices, pages, personal digital assistants (PDAs), video cameras and digital cameras.
- Wearable devices, such as smart watches with functionality similar to a PDA.
- Devices embedded in appliances such as washing machines, hi-fi systems, cars and refrigerators.

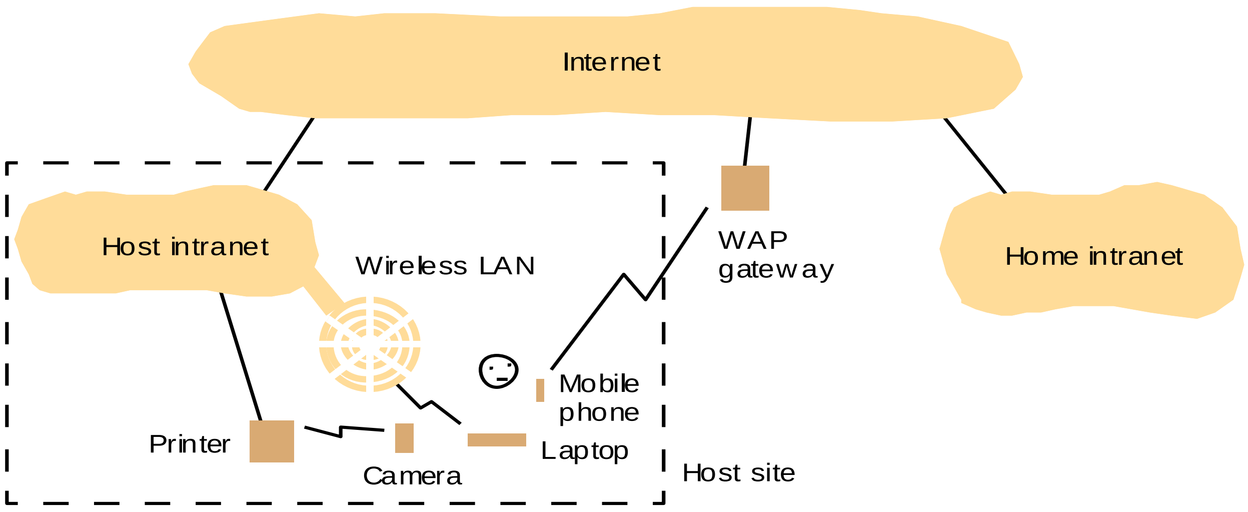
Mobile computing: Mobile computing is the performance of computing tasks while the user is on the move, or visiting places other than their usual environment. In mobile computing, users who are away from their 'home' intranet are still provided with access to resources via the devices they carry with them. The mobile computing increasing the service for users to utilize resources such as printers or even sales points that are conveniently nearby as they move around. This mobility service is also known as *location-aware or context-aware computing.*

Thus, Mobility introduces a number of challenges for distributed systems including to maintain operation in the face of device mobility.

Ubiquitous computing is small, cheap computational device that present in users' physical environments, including the home, office and even natural settings. The term 'ubiquitous' is intended to suggest that small computing devices will eventually become so pervasive (persistent) in everyday objects that they are hardly noticed. It means, its computational behaviour will be transparently and tied up with their physical function.

The presence of computers everywhere only becomes useful when they can communicate with one another. For example, it may be convenient for users to control their washing machine or their entertainment system from their phone or a 'universal remove control' device in the home. Equally, the washing machine could notify the user via a smart badge or phone when the washing is done.
The following figure shows a user who is visiting a host organization. The figure shows the user's home intranet and host intranet at the site that the user is visiting. Both intranets are connected to the rest of the Internet.

The user has access to three forms of wireless connection. Their laptop has a means of connecting to the host's wireless LAN. This network provides coverage of a few hundred metres(a floor of a building). It connects to the rest of the host intranet via a gateway or access point. The user also has a mobile telphone, which is connected to the internet. The phone gives access to the web and other internet services. Finally, the user carries a digital camera, which can communicate over a personal area wireless network with a device such as a printer.



Focus on resource sharing and web

We routinely share hardware resources such as printers, data resources such as files, and resources with more specific functionality such as search engines.

- Resources in a distributed system are physically encapsulated within computers and can only be accessed by communication.
- For effective sharing, each resource must be managed by a program that offers a communication interface enabling the resource to be accessed and updated reliably and consistently.

Service: A *service* is a distinct part of a computer system that manages a collection of related resources and presents their functionality to users and applications.

For example,
→ We access shared files through a file service;
→ We send documents to printers through a printing service;
→ We buy goods through an electronic payment service.

The only access we have to the service is via the set of operations that it exports.
For example, a file service provides *read*, *write* and *delete* operations on files.

Server and Client: The term *server* is refers to a running program (a *process*) on a networked computer that accepts requests from programs running on other computers to perform a service and responds appropriately.

The requesting processes are referred to as *clients,* and the overall approach is known as *client-server computing.*

In this approach, requests are sent in messages from clients to a server and replies are sent in messages from the server to the clients. When the client sends a request for an operation to be carried out, we say that the client *invokes an operation* upon the server.

A complete interaction between a client and a server, from the point when the client sends its request to when it receives the server's response, is called a *remote invocation.*

The same process may be both a client and a server, since servers sometimes invoke operations on other servers.

The terms 'client' and 'server' apply only to the roles played in a single request. In this case, Clients are active (making requests) and servers are passive (only waking up when they receive requests);

Servers run continuously, whereas clients last only as long as the applications of which they form a part.

World Wide Web The World Wide Web [www.w3.org I, Berners-Lee 1991] is an evolving system for publishing and accessing resources and services across the Internet. Through web browsers, users retrieve and view documents of many types, listen to audio streams and view video streams, and interact with an unlimited set of services.

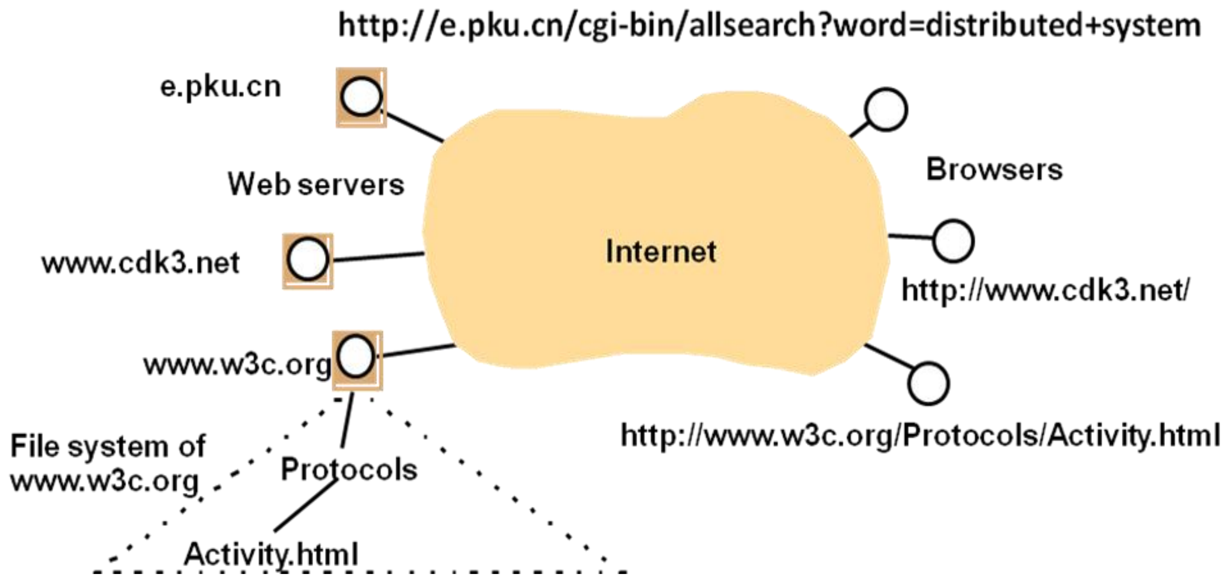
The Web began life at the European centre for nuclear research (CERN), Switzerland, in 1989 as a vehicle for exchanging documents between a community of physicists connected by the Internet [Berners-Lee 1999]. The feature of the Web is a *hypertext* structure among the documents that it stores, reflecting the users' requirement to organize their knowledge. This means that documents contain *links* (or *hyperlinks*) – references to other documents and resources that are also stored in the Web.

- **Definition**: Web is an open system that it can be extended and implemented in new ways without disturbing its existing functionality.
  - Its operation is based on communication standards and document standards
  - Respect to the types of 'resource' that can be published and shared on it.

The Web has moved beyond these simple data resources to encompass services, such as electronic purchasing of goods. It has evolved without changing its basic architecture. The Web is based on three main standard technological components:

- The ***HyperText Markup Language*** (HTML), a language for specifying the contents and layout of pages as they are displayed by web browsers;
- ***Uniform Resource Locators*** (URLs), also known as Uniform Resource Identifiers (URIs), which identify documents and other resources stored as part of the Web;
- *A* ***client-server system architecture***, with standard rules for interaction (the HyperText Transfer Protocol – HTTP) by which browsers and other clients fetch documents and other resources from web servers. Figure shows some web servers, and browsers making requests to them. It is an important feature that users may locate and manage their own web servers anywhere on the Internet

Web servers and web browsers:


http://e.pku.cn/cgi-bin/allsearch?word=distributed+system
e.pku.cn
Web servers
www.cdk3.net
www.w3c.org
Internet
Browsers
http://www.cdk3.net/
http://www.w3c.org/Protocols/Activity.html
File system of www.w3c.org
Protocols
Activity.html

HTML : The HyperText Markup Language [www.w3.org II] is used to specify the text and images that make up the contents of a web page, and to specify how they are laid out and formatted for presentation to the user. A web page contains such structured items as headings, paragraphs, tables and images. HTML is also used to specify links and which resources are associated with them.
Users may produce HTML by hand, using a standard text editor, but they more commonly use an HTML-aware 'wysiwyg' editor that generates HTML from a layout that they create graphically. A typical piece of HTML text follows:

<IMG SRC = "http://www.cdk5.net/WebExample/Images/earth.jpg"> 1
<P> 2
Welcome to Earth! Visitors may also be interested in taking a look at the 3
<A HREF = "http://www.cdk5.net/WebExample/moon.html">Moon</A>. 4
</P> 5

- This HTML text is stored in a file that a web server can access.
- A browser retrieves the contents of this file from a web server.
  -The browser interprets the HTML text
  -The server can infer the content type from the filename extension.

URLs : The purpose of a Uniform Resource Locator [www.w3.org III] is to identify a resource. It means, the term used in web architecture documents is Uniform Resource *Identifier* (URI),

Scheme: scheme-specific-location
e.g:
mailto:joe@anISP.net
ftp://ftp.downloadIt.com/software/aProg.exe
http://net.pku.cn/
....

The 'scheme', declares which type of URL this is. URLs are required to identify a variety of resources. For example, *mailto:joe@anISP.net* identifies a user's email address; *ftp://ftp.downloadIt.com/software/aProg.exe* identifies a file that is to be retrieved using the File Transfer Protocol (FTP) rather than the more commonly used protocol HTTP.

Other examples of schemes are 'tel' (used to specify a telephone number to dial, which is particularly useful when browsing on a mobile phone) and 'tag' (used to identify an arbitrary entity).

HTTP URLs are the most widely used, for accessing resources using the standard HTTP protocol. An HTTP URL has two main jobs: to identify which web server maintains the resource, and to identify which of the resources at that server is required.

The above figure (web servers and web browser) shows three browsers issuing requests for resources managed by three web servers.

-The topmost browser is issuing a query to a search engine.
-The middle browser requires the default page of another web site.
-The bottommost browser requires a web page that is specified in full, including a path name relative to the server.

In general, HTTP URLs are of the following form:

http:// servername [:port] [/pathName] [?query] [#fragment]

where items in square brackets are optional. A full HTTP URL always begins with the string 'http://' followed by a server name, expressed as a Domain Name System (DNS) name
Consider the URLs:

http://www.cdk5.net
http://www.w3.org/standards/faq.html#conformance
http://www.google.com/search?q=obama

Server DNS name	Pathname on server	Arguments
www.cdk3.net	(default)	(none)
www.w3c.org	Protocols/Activity.html	(none)
e.pku.cn	cgi-bin/allsearch	word=distributed+system

Note that the HTML directives, known as *tags*, are enclosed by angle brackets, such as *<P>*. Line 1 of the example identifies a file containing an image for presentation. Its URL is *http://www.cdk5.net/WebExample/Images/earth.jpg*. Lines 2 and 5 are directives to begin and end a paragraph, respectively. Lines 3 and 4 contain text to be displayed on the web page in the standard paragraph format.

HTTP:The HyperText Transfer Protocol [www.w3.org IV] defines the ways in which browsers and other types of client interact with web servers.

- Main features
  - ***Request-replay interaction*** : HTTP is a 'request-reply' protocol. The client sends a request message to the server containing the URL of the required resource. The server looks up the path name and, if it exists, sends back the resource's content in a reply message to the client. Otherwise, it sends back an error response such as the familiar '404 Not Found'.
  - ***Content types*** : The strings that denote the type of content are called MIME (RFC2045,2046)
  - ***One resource per request***: Clients specify one resource per HTTP request. If a web page contains nine images, say, then the browser will issue a total of ten separate requests to obtain the entire contents of the page.
  - ***Simple access control***: Any user with network connectivity to a web server can access any of its published resources. for example, by typing in a password.
- Dynamic content
  - Common Gateway Interface: a program that web servers run to generate content for their clients
- Downloaded code : Sometimes the designers of web services require some service-related code to run inside the browser, at the user's computer. In this particular way, need run one of the following code.
  - JavaScript
  - Applet

Challenges

The main challenges are

1. **Heterogeneity:** The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks. Heterogeneity applies to all of the following:
   - Networks;
   - Computer hardware;
   - Operating systems;
   - Programming languages;
   - Implementations by different developers.

Networks: The Internet consists of many different networks and their differences are masked by the internet protocols to communicate with one another. For example, a computer attached to an Ethernet and the entire Internet protocols are implementation is over the Ethernet.

Computer hardware: Data types such as integers may be represented in different ways on different hardware. For example, there are two alternatives for the byte ordering of integers. These differences in representation must be done when messages are to be exchanged between programs running on different hardware.

Operating Systems: The Operating Systems of all computers on the Internet need to include an implementation of the Internet protocols and with different Application Programs on each. For example, the calls for exchanging messages in UNIX are different from the calls in Windows.

Programming Languages: Different programming languages use different representations for characters and data structures such as arrays and records. These differences must be addressed if programs written in different languages are to be able to communicate with one another.

Implementations by different developers: Programs written by different developers cannot communicate with one another unless they use common standards, for example, for network communication and the representation of primitive data items and data structures in messages. For this, standards need to be agreed and adopted.

Middleware: The term *middleware* applies to a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages. The Common Object Request Broker (CORBA) is an example. Some middleware such as Java Remote Method Invocation (RMI) supports only a single programming language.

Middleware provides a uniform computational model for use by the programmers of servers and distributed applications.

Heterogeneity and mobile code: The term *mobile code* is used to refer to program code that can be transferred from one computer to another and run at the destination. Java applets are an example.

The *virtual machine* approach provides a way of making code executable on a variety of host computers. The compiler for a particular language generates code for a virtual machine instead of a particular hardware order code.

For example, the java compiler produces code for a java virtual machine, which executes it by interpretation. The java virtual machine needs to be implemented once for each type of computer to enable java programs to run.

Today, the most commonly used form of mobile code is JavaScript programs in some web pages loaded into client browsers.

2. **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 distributed systems 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.

In a word, the key interfaces are *published*. This process is akin to the standardization of interfaces, but it often bypasses official standardization procedures, which are usually cumbersome and slow-moving.
However, the publication of interfaces is only the starting point for adding and extending services in a distributed system. The challenge to designers is to tackle the complexity of distributed systems consisting of many components engineered by different people.
A benefit that is often cited for open systems is their independence from individual vendors.
To summarize:

- 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. But the conformance of each component to the published standard must be carefully tested and verified if the system is to work correctly.

3. **Security:** Many of the information resources are maintained in distributed systems have a high intrinsic value to their users. Their security is therefore of considerable importance. Security for information resources has three components:

- Confidentiality (protection against disclosure to unauthorized individuals) Eg: ACL in UNIX File system,
- Integrity (protection against alteration or corruption) Eg: checksum, and
- Availability (protection against interference with the means to access the resources) Eg: Denial of service.

In a distributed system, clients send requests to access data managed by servers, which involves sending information in messages over a network.
For example:
1. A doctor might request access to hospital patient data or send additions to that data.
2. In electronic commerce and banking, users send their credit card numbers across the Internet.

In both examples, the challenge is to send sensitive information in a message over a network in a secure manner. But security is not just a matter of concealing the contents of messages – it also involves knowing for sure the identity of the user or other agent on whose behalf a message was sent.
In the first example, the server needs to know that the user is really a doctor, and in the second example, the user needs to be sure of the identity of the shop or bank with which they are dealing.
The second challenge here is to identify a remote user or other agent correctly. Both of these challenges can be met by the use of encryption techniques developed for this purpose.

4. **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. The number of computers and servers in the Internet has increased dramatically.

Figure shows the increasing number of computers and web servers during the 12-year history of the Web up to 2005 [zakon.org]. It is interesting to note the significant growth in both computers and web servers in this period, but also that the relative percentage is flattening out.

Figure Growth of the Internet (computers and web servers)

Date	Computers	Web servers	Percentage
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, July	~200,000,000	42,298,371	21
2005, July	353,284,187	67,571,581	19

The design of scalable distributed systems presents the following 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. For example, the frequency with which files are accessed in an intranet is likely to grow as the number of users and computers increases.

   For example, if a single file server can support 20 users, then two such servers should be able to support 40 users.
2. *Controlling the performance loss*: Consider the management of a set of data whose size is proportional to the number of users or resources in the system.

   For example, the table with the correspondence between the domain names of computers and their Internet addresses held by the Domain Name System (DNS), which is used mainly to look up DNS names such as www.amazon.com.
3. *Preventing software resources running out*: An example of lack of scalability is shown by the numbers used as Internet (IP) addresses (computer addresses in the Internet). In the late 1970s, it was decided to use 32 bits for this purpose, the supply of available Internet addresses is running out. For this reason, a new version of the protocol with 128-bit Internet addresses is being adopted, and this will require modifications to many software components.
4. *Avoiding performance bottlenecks*: In general, algorithms should be decentralized to avoid having performance bottlenecks. The Domain Name System removed this bottleneck by partitioning the name table between servers located throughout the Internet and administered locally.

5. Failure Handling Computer systems sometimes fail. When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the proposed computation.

Failures in a distributed system are partial – that is, some components fail while others continue to function. Therefore the handling of failures is particularly difficult.
The following techniques for dealing with failures:

Detecting failures: Some failures can be detected. For example, checksums can be used to detect corrupted data in a message or a file. That is, difficult or even impossible to detect some other failures, such as a remote crashed server in the Internet.

The challenge is to manage in the presence of failures that cannot be detected but may be suspected.

Masking failures: Some failures that have been detected can be hidden or made less severe. Two examples of hiding failures:
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.

Tolerating failures: Most of the services in the Internet do exhibit failures.

For example, when a web browser cannot contact a web server, it does not make the user wait forever while it keeps on trying.

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

Redundancy: Services can be made to tolerate failures by the use of redundant components. Consider the following examples:
1. There should always be at least two different routes between any two routers in the Internet.
2. In the Domain Name System, every name table is replicated in at least two different servers.
3. A database may be replicated in several servers to ensure that the data remains accessible after the failure of any single server;

- the servers can be designed to detect faults in their peers;
- when a fault is detected in one server, clients are redirected to the remaining servers.

6. Concurrency: Both services and applications provide resources that can be shared by clients in a distributed system. Therefore several clients will attempt to access a shared resource at the same time. For example, a data structure that records bids for an auction may be accessed very frequently when it gets close to the deadline time.

- Correctness
  - ensure the operations on shared resource correct in a concurrent environment

  e.g. records bids for an auction.
- Performance
  - Ensure the high performance of concurrent operations.

7. Transparency: Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components. The implications of transparency are a major influence on the design of the system software.
Access transparency enables local and remote resources to be accessed using identical operations.
Location transparency enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
Concurrency transparency enables several processes to operate concurrently using shared resources without interference between them.
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.
Mobility transparency allows the movement of resources and clients within a system without affecting the operation of users or programs.
Performance transparency allows the system to be reconfigured to improve performance as loads vary.
Scaling transparency allows the system and applications to expand in scale without change to the system structure or the application algorithms.

SYSTEM MODELS

Introduction: Systems which are used in real-world environments should be designed to function correctly in the widest possible.

1. Architecture model
2. fundamental model

Deficulties for and threads to distributed System:

1. **Widely varying modes of use:** The components (or) parts of systems are wide variations in workload – for example, some web pages are accessed several million times a day. In this case, some parts of a system may be disconnected, or poorly connected some of the time.

For example, multimedia applications have special requirements for high communication bandwidth and low latency.

2. **Wide range of system environments:** A distributed system must accommodate heterogeneous hardware, operating systems and networks.
   → The networks may differ widely in performance i.e., wireless networks operate at a fraction of the speed of local networks.
   → In distributed systems, differing scales of system must be supported i.e., ranging from tens of computers to millions of computers.
3. **Internal problems:** Non-synchronized clocks, conflicting data updates and many modes of hardware and software failure involving the individual system components.
4. **External threats:** Attacks on data integrity and secrecy, denial of service attacks.

To overcome the above problems in distributed system, the properties and design issues can change with the use of **descriptive system models**. Each type of model is planned to provide an abstract, simplified but consistent description of a relevant aspect of distributed system design:

Architectural models describe a system in terms of the computational and communication tasks performed by its computational elements. It means, the computational elements being individual computers or aggregates of them supported by appropriate network interconnections.

Fundamental models are concerned with properties that are common in all of the architectural models. It is addressed by three fundamental models that examine the important aspects of distributed systems:
→ ***interaction models,*** which consider the structure and sequencing of the communication between the elements of the system;
→ ***Failure models,*** which consider the ways in which a system may fail to operate correctly.
→ ***Security models***, which consider how the system is protected against attempts to interfere with its correct operation or to steal its data.

Architectural models: The architectural of a system has a collection of individually defined components. The complete objective is to make sure that the structure will satisfy the current and future demands on it. The vital responsibilities are to make the system cost effective, manageable, reliable and adaptable. The architectural design of a building has same features like determining architectural style gives a constant frame of reference for the design.

The central architectural models are constructed almost on the notion of process and object. The functioning of separate components of a distributed system is first simplified and then abstracted in an architectural model. This model later evaluates the following things.

a) The arrangement of components throughout a network of computers i.e., trying to specify beneficial patterns for the distribution of data and workload.
b) The interrelationships b/w the components i.e., the components functional roles and the pattern of communication.

By altering client server model, one can construct few dynamic systems.

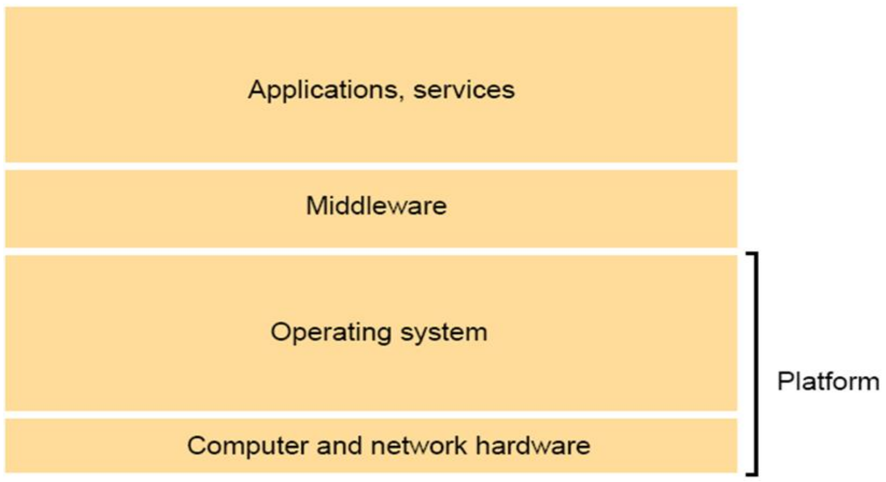
a) A process can assign a task to another, since it is possible to the code from one process to another.
   Example: The code from servers is downloaded by the client processes and executed locally. To minimize access delays and communication traffic, the code and objects that accesses the process can be transferred.
b) Few of the distributed systems are designed to allow the computers and other mobile devices to be added or removed smoothly, enabling to find obtainable services and also to provide their services to others.

There are many patterns that are used fully for the allocation of work in a distributed system. These patterns have a major effect on the performance of the resulting system.
An architectural model of a distributed system is concerned with the placement of its parts and the relationships b/w them.
Example: Client-server, Peer-to-peer

1. **Software Layers:** The software architecture refers to services offered and requested b/w processes located in the same or different computers.

The structuring of software as modules or layers in a single computer and the services extended and appealed b/w processes situated in the same or different computers is actually referred to as software architecture. The disclosed form of this process and service is a layer of services in distributed systems. This layer of service included both software and hardware services. The process which accepts request from other processes is a server. One or more servers can supply a distributed service. To support a constant system-wide view of the service resources, these servers can communicate with one another and also with client processes.
The above figure introduced two important terms platform and middleware.

1. **Platform:** The bottom layers (i.e., operating system, computer and network hardware layers) provide a platform for distributed applications. The services provided by these bottom layers are utilized by higher layers. In every computer, operating system layer and computer network layer are implemented separately. This implementation facilitates communication and coordination b/w processes by transferring the system's programming interface to that level.
   Examples: Intel x86/Windows, Intel x86/Solaris, PowerPC/MAC OS, Intel x86/Linux



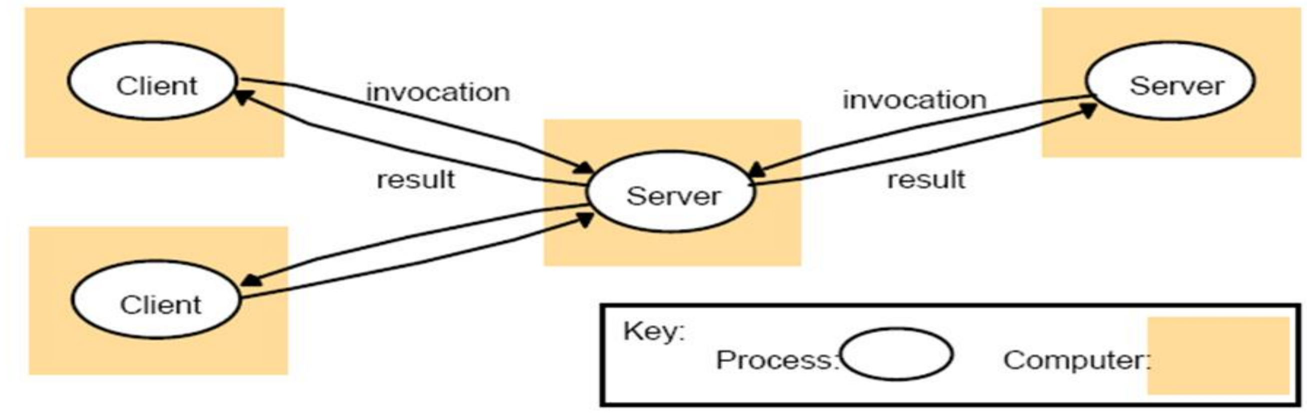
2. **Middleware:** A piece of software placed b/w the application and operating system is called "middleware". Its intension is to enclose heterogeneity and to supply an accessible programming model to application programmers. The concept of middleware can be easily explained in the context of objects or processes in a set of computers that interact with each other to carryout communication and resource-sharing support for distributed application. Middleware supplies helpful building blocks in the construction of software components. These components can interact in distributed applications.

   It provides useful building blocks. They are Remote method invocation, communication b/w a group of processes, notification of events, partitioning, placement and retrieval of data or objects, replication, transmission of multimedia data in real time.

   The earliest examples of middleware is remote procedure calling packages (eg: sun RPC) and group communication system [eg: Isis]. Object-oriented middleware products and standards are
   a) CORBA b) java RMI c)Microsoft Distributed Component objects model (DCOM)
   d) Web services e) The ISO/ITU-T's Reference model for open distributed processing (RM-ODP).

2. **System Architecture:** The architecture model has two types of environments in distributed systems. They are
1. Client-server model and **2. Multiple server model.**

Client-server model: It is very frequently referred in distributed systems. This architecture is broadly utilized and significantly very important. The figure below shows the structure of client's server model. The figure shows the structure of client-server model.

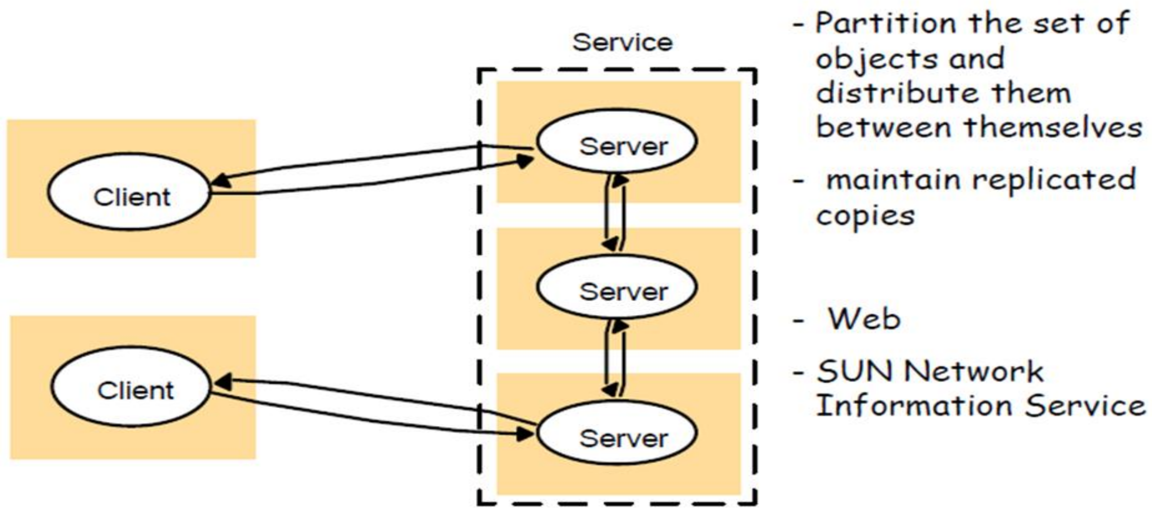


In order to access the shared resources managed by server processes in different host computers, the client processes interact with those server processes. The figure above specifies that the clients of some servers may also be consisting the web pages has a common client called web server. The DNS service has many clients such as web servers and many other internet services. The service provided by DNS is that it interprets Internet domain names to network address. Search engines are another example related to web. It allows users to search the sites all over the Internet for synopsis of information accessible on web pages. These synopses are created by programs called web crawlers.

At search engine site, these web crawlers are executed in the background, using HTTP requests to access web servers all over the Internet. A search engine can play the role of both a server and a client. It performs the following two tasks,

a) It replies to queries from browser clients.
b) Executing web crawlers that act as clients of other web servers.

Multiple server's model: In multiple server model, several servers implement the services as its processes in separate host computers. After implementation these computers interact with each other if required in order to provide the service either by partitioning the services into set of objects and distributing them among themselves or by maintaining replicate copies of these services on several hosts. The server may also provide the service either by partitioning the services into set of objects and distributing them among themselves by maintain replicate copies of these services on several hosts. The multiple server model is shown in fig.


Service
Client
Server
Server
Client
Server
- Partition the set of objects and distribute them between themselves
- maintain replicated copies
- Web
- SUN Network Information Service

The concept of this model is illustrated in the following examples.

1. On the web each server maintains their own resources, called partitioned data. Therefore, a user can access a resource of any one server.
2. Replication is an important concept of increasing availability and performance and for improving fault tolerance. The processes that run on different computers can have multiple

consistent copies of data due to replication. For an instance, data available at mittal elec.com might also be provided by some other web services. It is due to mapping of service onto several servers mainlining database replicated in memory.

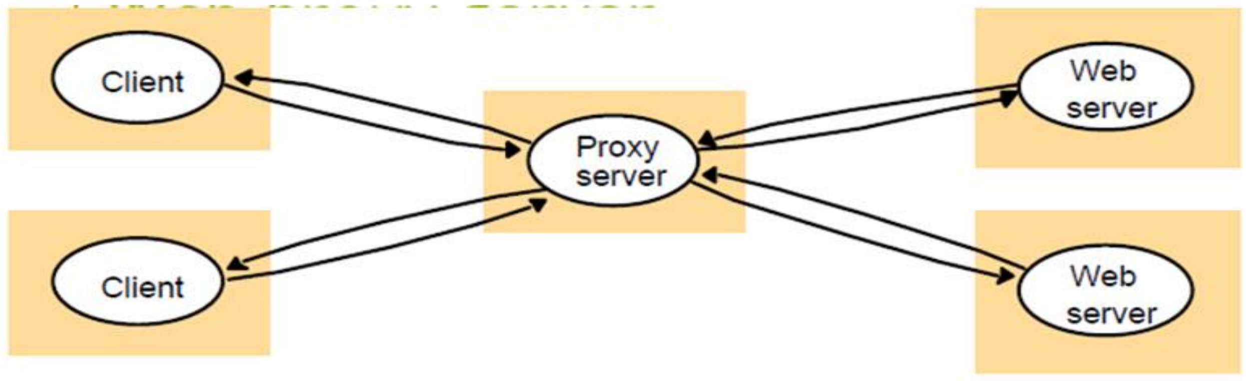
3. Another example of replication based service is the Sun NIS (Network Information Service). Each server of NIS maintains duplicate copies of password file that holds a list of user's login names along with encrypted passwords.

3.Proxy servers and caches: A cache is repository. A cache has storage facility of recently used objects that is closer to the objects themselves. A web proxy server provides a shared cache of web resources for the client machines at a site or across several sites.

→ Increase availability

→ Increase performance

→ Reduce the load on the wide-area network and web server.

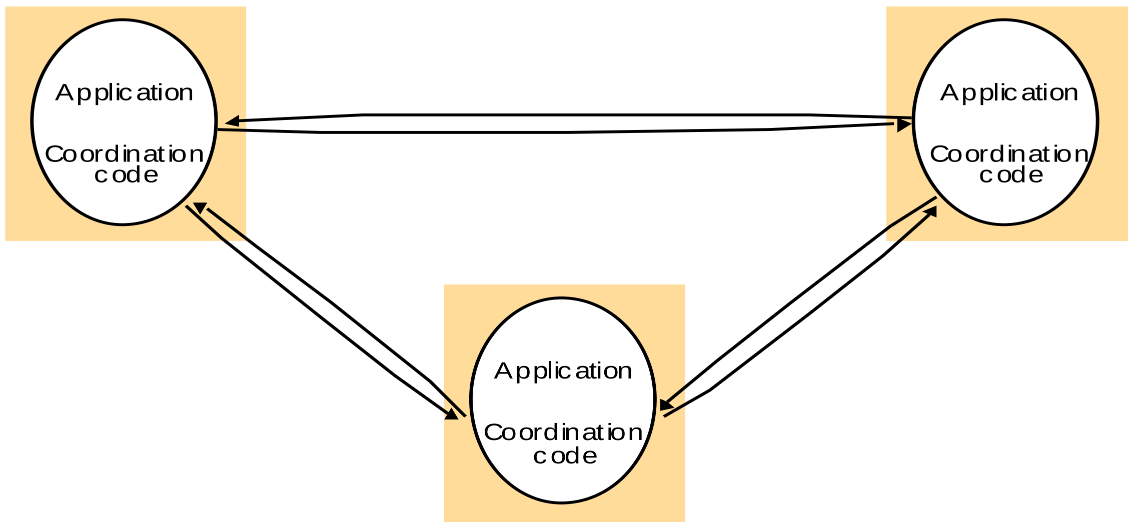


When a client process requests an object, the caching service starts examining the cache for updated copy. If the updated copy is available, then caching service provides the object from the repository. Otherwise, it retrieves an updates copy. In general caches are situated in a proxy server that is shared by multiple clients or available with each client.

The objective of a proxy server is to extend the performance and availability of the service by decreasing the load on wide-area network and web servers. Apart from this, they are used to get remote web servers through a firewall. For a client machine at a site or over multiple sites, a shared cache is supplied by web proxy server.

4.A Distributed application on Peer process: In peer processes architecture, all processes play similar roles and interact cooperatively with each other as peers, performing distributed activity or computation without making any differentiation among clients and servers.

The code in peer processes is responsible of maintaining consistency for application level resources and synchronizing application level actions based on the requirements. The peer processes architecture is shown in figure.

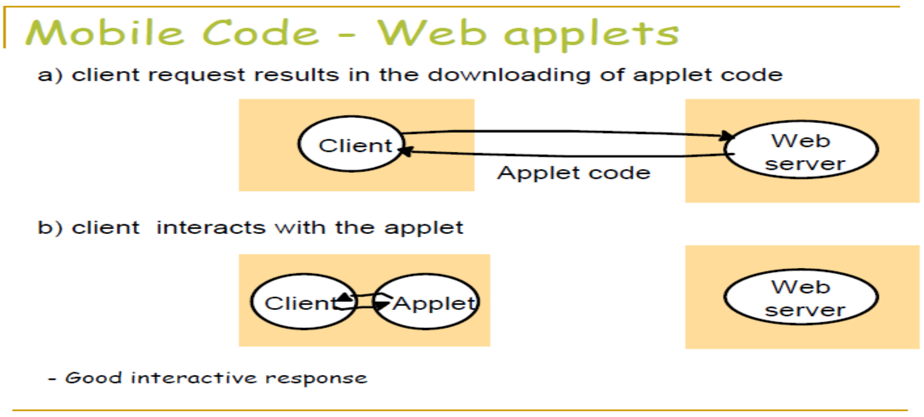


For example, assume an application of a distributed 'white board' that allows users to view and modify a picture shared b/w several computers. This can be made possible by implementing as an application process at each site relying on middleware layers. So that event notification and group communication can be performed in order to notify the changes to the picture for all application processes.

5. Client-server Model Variations for Mobile Code:

Several variations on the client server model can be derived from the consideration of the following factors:

- ➢ The use of mobile code and mobile agents
- ➢ Users need for low cost computer with limited H/W resources that are simple to manage.
- ➢ The requirement to add and remove mobile device in convenient manner.
- Example: Java applets
  - The user running a browser selects a link to an applets whose code is stored on a web server.
  - The code is downloaded to the browser and runs there.
- Advantage:
  - Good interactive response since.
  - Does not suffer from the delays or variability of bandwidth associated with network communication.
- Disadvantage:
  - Security threat to the local resources in the destination computer.


Mobile Code - Web applets
a) client request results in the downloading of applet code
Client
Web server
Applet code
b) client interacts with the applet
Client
Applet
Web server
- Good interactive response

Mobile Agents: A mobile agent is a running program that travels from one system to another in a network. A network performs a task such as gathering information, providing results on behalf of someone. A mobile agent may make many invocations to local resources at each site it visits. Eg: access to individual database entries.

Mobile agents are possibly used in cases such as

a) Install software on the computers of an organization.
b) To compare the prices of products different vendors by visiting the site of every vendor and carryout a sequence of database operations.

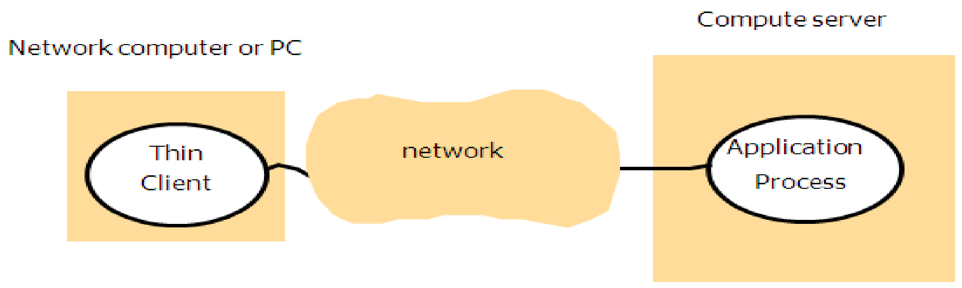
A recent example is worm program developed at Xerox PARC. It is designed in such a way to transfer detailed computations by using of idle computers.

Thin Clients: A window-based user interface on a computer local to the user while running applications on a remote computer is supported by a software layer referred to as thin client.

A computer server has the capacity to run applications simultaneously. It is a powerful computer that runs a multiprocessor version of an operating system (UNIX or windows NT).

The highly interactive graphical activities such as CAD and image processing are the limitation of the tin client architecture. Since, the transfer of image and vector information b/w the thin client and application process include both network and operating system potentials, the delay experience by users is increased.

Implementation: The implementation of thin client systems is uncomplicated. For examole, X-11 window system, a variant of UNIX. Other implementation includes The citrix Win Frame Product and the Tele-porting and Virtual Network Computer (VNC) systems.



Network Computers: The desktop computer local to a user is used to run the applications. For these computers, the operating system and application software generally needs data to be placed on a local disk and more active code. A more technical effort is needed to manage application files and to maintain local software but most of the users are not eligible to provide this effort.

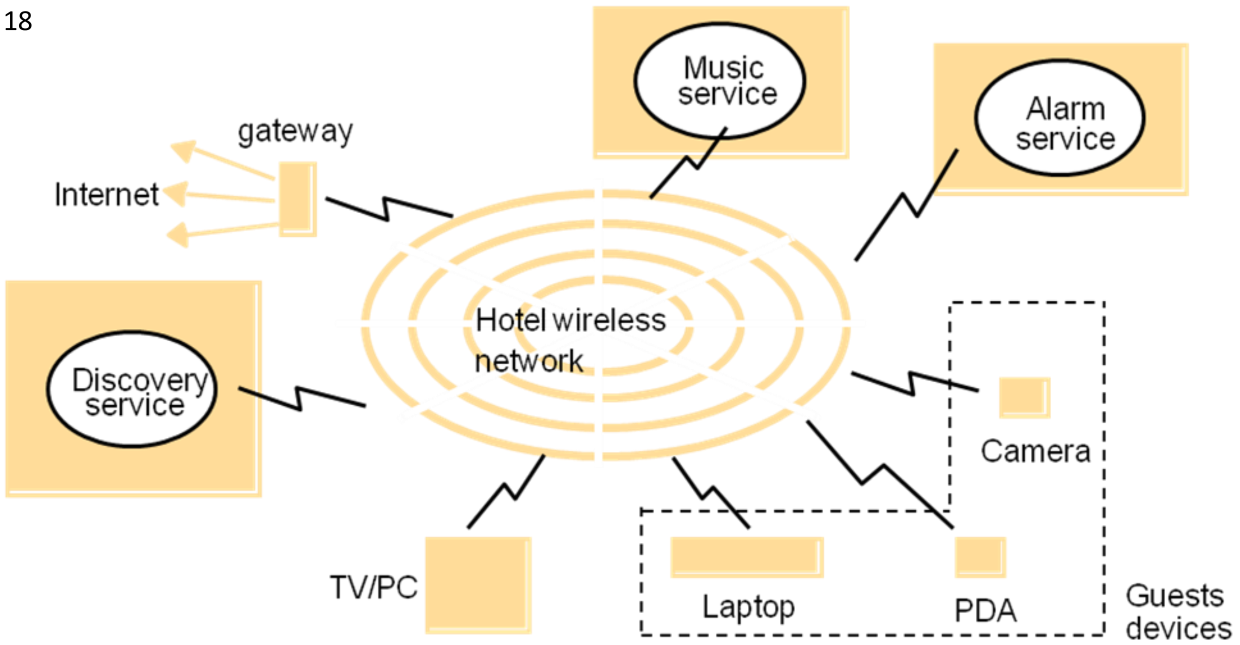
With the help of network computer, the user downloads application software and its operating system from remote file server. It takes response to the above problem. Although the files are administered by a remote file server, applications such as web browser also run locally. It is possible for a user to move from one network computer to other, because a file server stores all the data and code of an application.

Mobile devices: Mobile devices are the hardware computing components that carry software and are freely movable b/w physical locations and networks. Some examples of mobile devices include the following: laptops, handheld devices, mobile phones, PDAs (Personal Digital Assistants), digital cameras, and some wearable computers like smart watches.

Mostly mobile devices are supported by wireless networking. They can be oerated on greater ranges of networks such as GSM and 3G telecommunication networks. Some of them can operate over network upto hundreds of meters like Wifi (IEEE802.11) or upto ten meters like Bluetooth. On mobile devices there is a possibility of existing for both, client and server. The existences of clients are common, but in some cases it might be possible for a mobile device to have server as well.

6.Mobile devices and Spontaneous networking : Key Features of Spontaneous Networking are easy connection to a local network and Easy integration with local services

- Security problems from connectivity of mobile users:
  - Limited connectivity
  - Security and privacy
- The purpose of a **discovery service** is to accept and store details of services that are available on the network and to respond to queries from clients.
- A discovery service offers two interfaces:
  - A registration service accepts registration requests from servers and records the details in the discovery service's database.
  - A lookup service accepts queries concerning available services and searches its database for registered services that match the queries.



Interface and Objects:

- Interface definitions specify the set of functions available for invocation in a server (or peer) processes.
- In OO paradigm, distributed processes can be constructed as objects whose methods can be accessed by remote invocation (COBRA approach).
- Many objects can be encapsulated in server or peer processes.
- Number, types, and locations (in some implementation) of objects may change dynamically as system activates require.
- Therefore, new services can be instantiated and immediately be made available for invocation.

<u>Design Requirements for Distributed Architectures</u>:

1. **Performance Issues:**
   - Considered under the following factors:
     - Responsiveness:
       - Fast and consistent response time is important for the users of interactive applications.
       - Response speed is determined by the load and performance of the server and the network and the delay in all the involved software components.
       - If the process is in the same computer then also data transfer rate b/w processes and associated control switching is low. This can be overcome if the system must be composed of relatively few software layers and small quantities of transferred data to achieve good response times.
       - In case of remote text pages, good response time can be achieved due to their small size.
       - In case of graphical images that contain large volume of data, interactive response time is very slow.
     - Throughput:
       - It can be defined as the rate of completing the computational task. It is the traditional performance measure for computer systems.
       - It determines the ability of performing the work for all users in a distributed system.
       - Data present on remote server if needed to be passed from a server process to a client process, then it has to pass through all software layers present in both computers.

- Load balancing:
  - Enable applications and service processes to proceed concurrently without competing for the same resources.
  - Exploit available processing resources.

2. **Quality of Service provided in distributed systems:**
   - The main system properties that affect the service quality are:
     - **Reliability:** related to failure fundamental model (discussed later).
     - **Performance**: ability to meet timeliness guarantees.
     - **Security**: related to security fundamental model (discussed later).
     - **Adaptability**: ability to meet changing resource availability and system configurations.
3. **Dependability issues**:
   - The dependability issues can be defined as correctness, security and fault tolerance.
   - Dependability is the crucial requirement in most application domains.
   - The issues achieved by:
     - **Fault tolerance**: The ability to continue its function in the presence of failures that may arise in software, hardware and networks. Reliability of an application can be obtained by means of redundancy.
     - **Security:** locate sensitive data only in secure computers. For example, consider the database of a bank containing customer records with sensitive and widely used items. The sensitive issues must be visible to only some specific bank authorities.
     - Correctness of distributed concurrent programs: research topic.

Fundamental Models:
A model contains only the essential ingredients needed to understand and reason about some aspects of a system's behavior.

System Model: System model is a model used to create accurate assumptions regarding the systems that are to be modeled. It is used to create generalizations about the systems that are to be modeled based on the assumptions made. Here, generalizations are considered as efficient general purpose algorithm or the properties which are required and guaranteed.

The factors that are to be considered for the construction of fundamentals models:

a) **Security:** The distributed systems have large attacks because of their openness and modular nature. The security in fundamental model defines attack front forms with the threat analysis and system design that are resistant to threats.

b) **Interaction:** Interaction is the communication and coordination b/w the processes with the exchange of messages. The interaction model in distributed systems must reflect the following facts:
i) Communication involves delay of specific duration.
ii) The delays limit the accuracy of process coordination.
iii) Difficult to maintain same time conventions across the computers.

c) **Failure:** The operation of a distributed system is large to failure with the occurrence of fault in the network or in the computer which is connected to it. Fundamental model describes the faults and analyses their effects. It also designs fault tolerable systems capable of running even in the presence of faults.

1.Interaction Model: Interacting processes are responsible for the execution of activities in the distributed systems. Each process has a state that cannot be accessed or updated by another process. A state consists of variables and a set of data that can be updated and accessed by its state.

Interacting processes in a distributed system are affected by two significant factors:

a) Performance of communication channels: is characterized by:

i) **Latency**: It is the delay that occurs when the message is transmitted from one process to another. The delay includes the time in between sending and receipt of a message including

1. Network access time.
1. Time for first bit transmitted through a network to reach its destination.
2. Processing time within the sending and receiving processes.

ii. Throughput/ latency : number of units (e.g., packets) delivered per time unit.

- It can be defined as the rate of completing the computational task. It is the traditional performance measure for computer systems.
- It determines the ability of performing the work for all users in a distributed system.
- Data present on remote server if needed to be passed from a server process to a client process, then it has to pass through all software layers present in both computers.

iii. Bandwidth: It is the amount of data that can be transmitted over a network in a given time period. If same network is being used by many communication channels, then they have to share the bandwidth.

iv. Jitter: It is the difference in time during the transmission of a message. This difference in time can be seen when a series of audio data samples with variant time intervals are played resulting into distortion in time.

b) <u>Computer clocks:</u>

- Each computer in a distributed system has its own internal clock to supply the value of the current time to local processes.
- Therefore, two processes running on different computers read their clocks at the same time may take different time values.
- *Clock drift rate* refers to the relative amount a computer clock differs from a perfect reference clock.
- Several approaches to correcting the times on computer clocks are proposed.
- Clock corrections can be made by sending messages; from a computer has an accurate time to other computers, which will still be affected by network delays.

Types of Interaction model: Setting time limits for process execution, as message delivery, in a distributed system is hard.
Two opposing extreme positions provide a pair of simple interaction models:

***Synchronous* distributed systems:**

- A system in which the following bounds are defined:
  1. 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 perfect time has a known bound.
- Easier to handle, but determining realistic bounds can be hard or impossible.

***Asynchronous* distributed systems:**

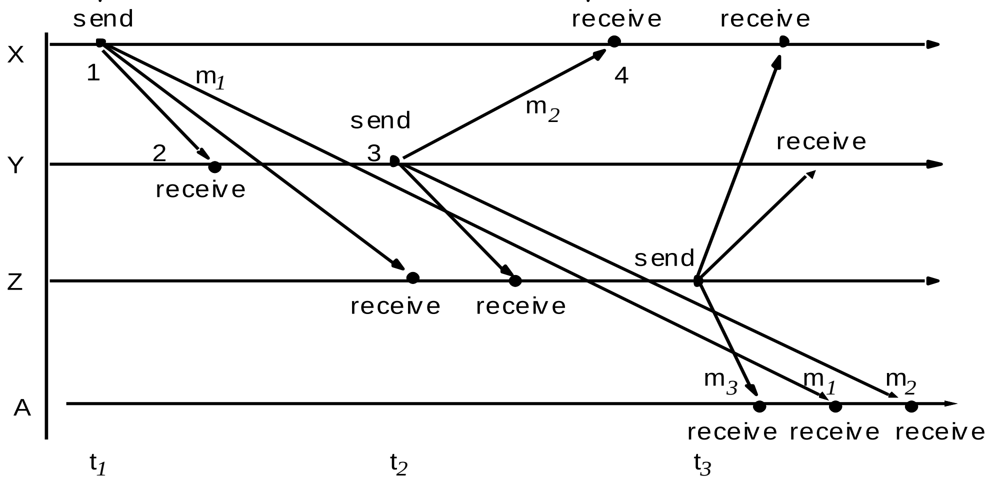
- A system in which there are no bounds on:
  1. Process execution times.
  2. Message delivery times.
  3. Clock drift rate.
- Allows no assumptions about the time intervals involved in any execution.
- Exactly models the Internet.
  - Browsers are designed to allow users to do other things while they are waiting.

➢ More abstract and general:

- A distributed algorithm executing on one system is likely to also work on another one.

Event ordering: if an event at one process (sending or receiving a message) occurred before, after, or concurrently with another event at another process, then DS algorithms keep them in order.

- Why because it is impossible for any process in a distributed system to have a view on the current global state of the system.
- The execution of a system can be described in terms of events and their ordering even though the lack of accurate clocks.
- Logical clocks define some event order based on causality.
- Logical time can be used to provide ordering among events in different computers in a distributed system (since real clocks cannot be synchronized).



2. Failure Model: Defines the ways in which failure may occur in order to provide an understanding of its effects.

- The classification of failures and differences between the failures of processes and communication channels is provided as:
  - *Omission* failures
    - Process or channel failed to do something.
    - Communication omission failures.
  - *Arbitrary* failures
    - Any type of error can occur in processes or channels (worst).
  - *Timing* failures
    - Applicable only to synchronous distributed systems where time limits may not be met.

I. *Omission* failures:

- Process or channel failed to do something.
- Communication omission failures.

➔ Omission failures are classified into

1. Process omission failure 2. Communication omission failures

1. Process Omission Failures: The primary reason in processes for omission failure is its crash. That means a process has been terminated completely. The design services that can resist the faults can be simplified on the assumption that the services crash clearly. In a clean crash, a process terminates or executes correctly. Due to a crash, the process does not respond to invocation messages repeatedly. This favors other processes in identifying the crash using timeout implies that a process is not responding. The reasons for this may be its crash or slow performance, or undelivered of messages.

A crash that is identified by other processes is called fail-stop. This can be introduced in a synchronous system as:

→ If messages delivery is correct.

→ If processes employ timeouts identify failure of their processes.
Consider the example, if the reply to a process is not delivered within the specified time limit, then the process that is waiting for the reply concludes that sender process has been failed.

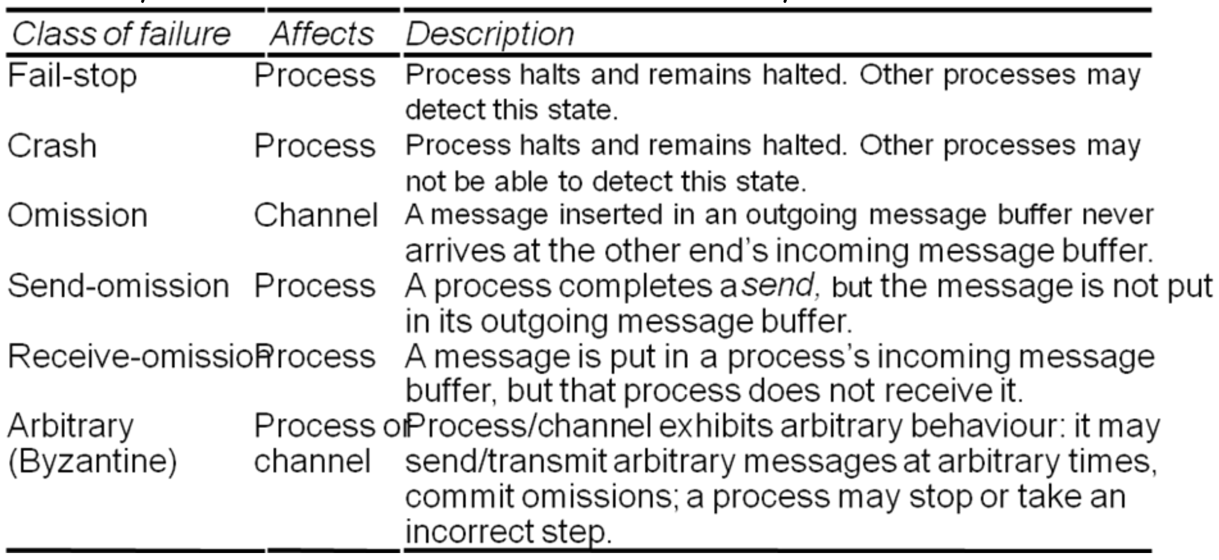
2. Communication Omission Failures: Send and receive are two communication primitives. Consider the example where send operation is performed by process A. It inserts a message in its outgoing message buffer. Communication channel transmits this message to process B's incoming message buffer. Receive operation is performed by process B that takes the message from its incoming message buffer and delivers it.

The communication channel raises an omission failure if the message from A's outgoing message buffer is not transmitted to B's incoming message buffer. This is called a dropped message which is due to insufficient buffer space at the receiver site or at the intervening gateway, or by error at network transmission. These consequences are identified by the checksum that is attached with the message content.

Following are the failures resulting due to loss of messages:

a) **Send-omission failure:** Occur in b/w sending process and outgoing message buffer.
b) **Receive-Omission failures:** Occur in b/w incoming message buffer and receiving process.
c) **Channel-Omission failures:** Occur in b/w outgoing message buffer and incoming message buffer.

II. Arbitrary Failures: Arbitrary failure describes the semantics of a worst failure in which there is a possibility of occurrence of any type of error. In a process an arbitrary failure is caused when it skips any of its execution phases or adopts a new phase for execution. For example, a process may set incorrect values in its data items or it may return an incorrect value.

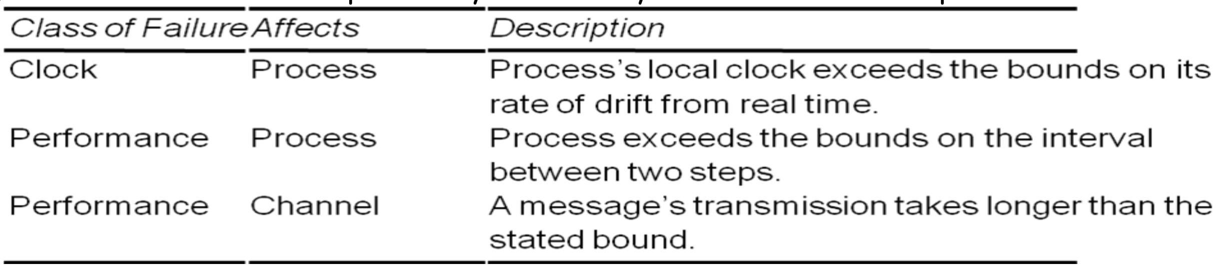


Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	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.

III. Timing Failures: Timing failure is applicable in synchronous distributed system whereas in asynchronous distributed system, it is not guaranteed.

a) **Synchronous Distributed System:** In synchronous distributed system, time limits are fixed on process execution time, message delivery time and clock drift rate. A process is affected if it exceeds its time bounds. A channel is affected if the time taken for the transmission exceeds the specified bounds. A clock failure affects its process if its local time exceeds its drift rate bounds from real time. These three failures fail to respond within the specified bounds.

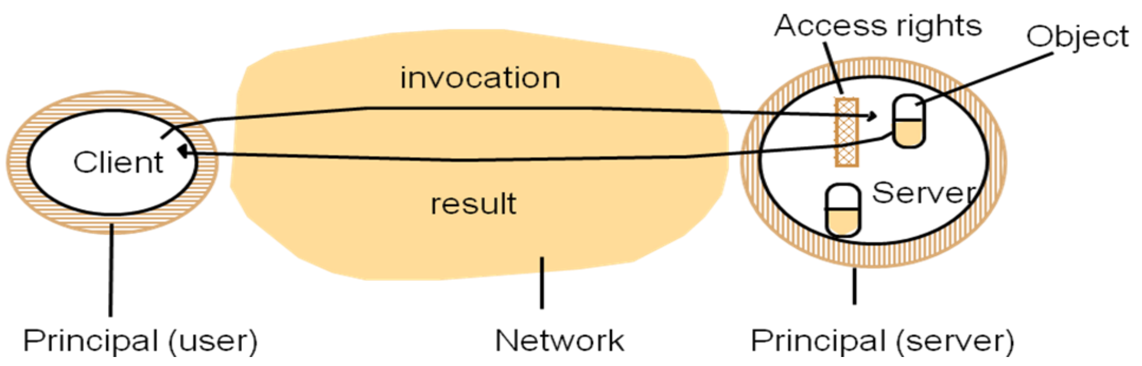
b) **Asynchronous Distributed system:** In asynchronous distributed system, timing failure cannot be guaranteed and there is a possibility of a heavily loaded server to respond late.



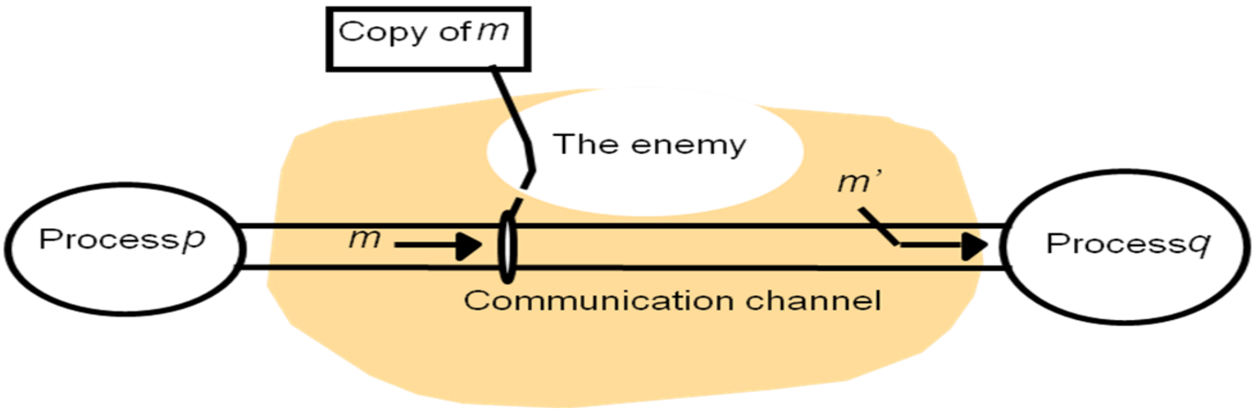
Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

2.4.4. Security Model: Secure processes and channels and protect objects encapsulated against unauthorized access.

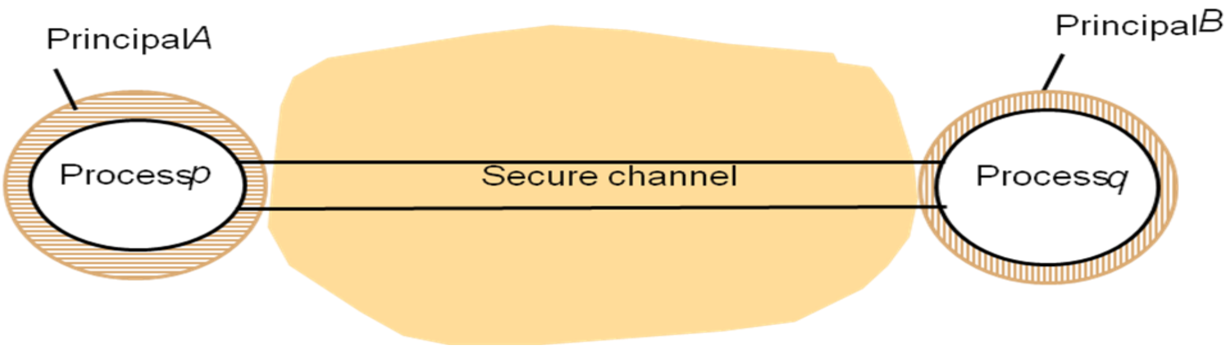
- **Protecting access to objects**
  - Access rights
  - In client server systems: involves authentication of clients.
- **Protecting processes and interactions**
  - Threats to processes: problem of unauthenticated requests / replies.
    - e.g., "man in the middle"
  - Threats to communication channels: enemy may copy, alter or inject messages as they travel across network.
    - Use of "secure" channels, based on cryptographic methods.



Objects and principals



The enemy



Making failures – Reliability of one-to-one communication:

- A service masks a failure, either by hiding it all together or by converting it into a more acceptable type of failure.
  - Checksums are used to mask corrupting messages -> a corrupted message is handled as a missing message
  - Message omission failures can be hidden by re-transmitting messages.
- The term reliable communication is defined in terms of validity and integrity as follows:
  - Validity: any message in the outgoing buffer is eventually delivered to the incoming message buffer
  - Integrity: the message received is identical to one sent, and no messages are delivered twice.