Distributed Systems

LECTURE NOTES

B.E III YEAR – V SEM (2025-26)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Sri Raghavendra Educational Institutions Society(R)
Sri Krishna Institute of Technology
(Accredited by NAAC Approved by A.I.C.T.E. New Delhi,
Recognized by Govt. of Karnataka Affiliated to V.T U., Belagavi)
#57, Chimney Hills, Hesaraghatta Main Road, Chikkabanavara Post,
Bangalore- 560090

MODULE I

Characterization of Distributed Systems: Introduction, Examples of Distributed systems, Resource Sharing and Web, Challenges.

System Models: Introduction, Architectural models, Fundamental models.

Characterization of Distributed Systems: Introduction

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages.

A distributed system as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages. This simple definition covers the entire range of systems in which networked computers can usefully be deployed.

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 has the following significant consequences:

***Concurrency*:** In a network of computers, concurrent program execution is the norm. I can do my work on my computer while you do your work on yours, sharing resources such as web pages or files when necessary. The capacity of the system to handle shared resources can be increased by adding more resources (for example. computers) to the network.

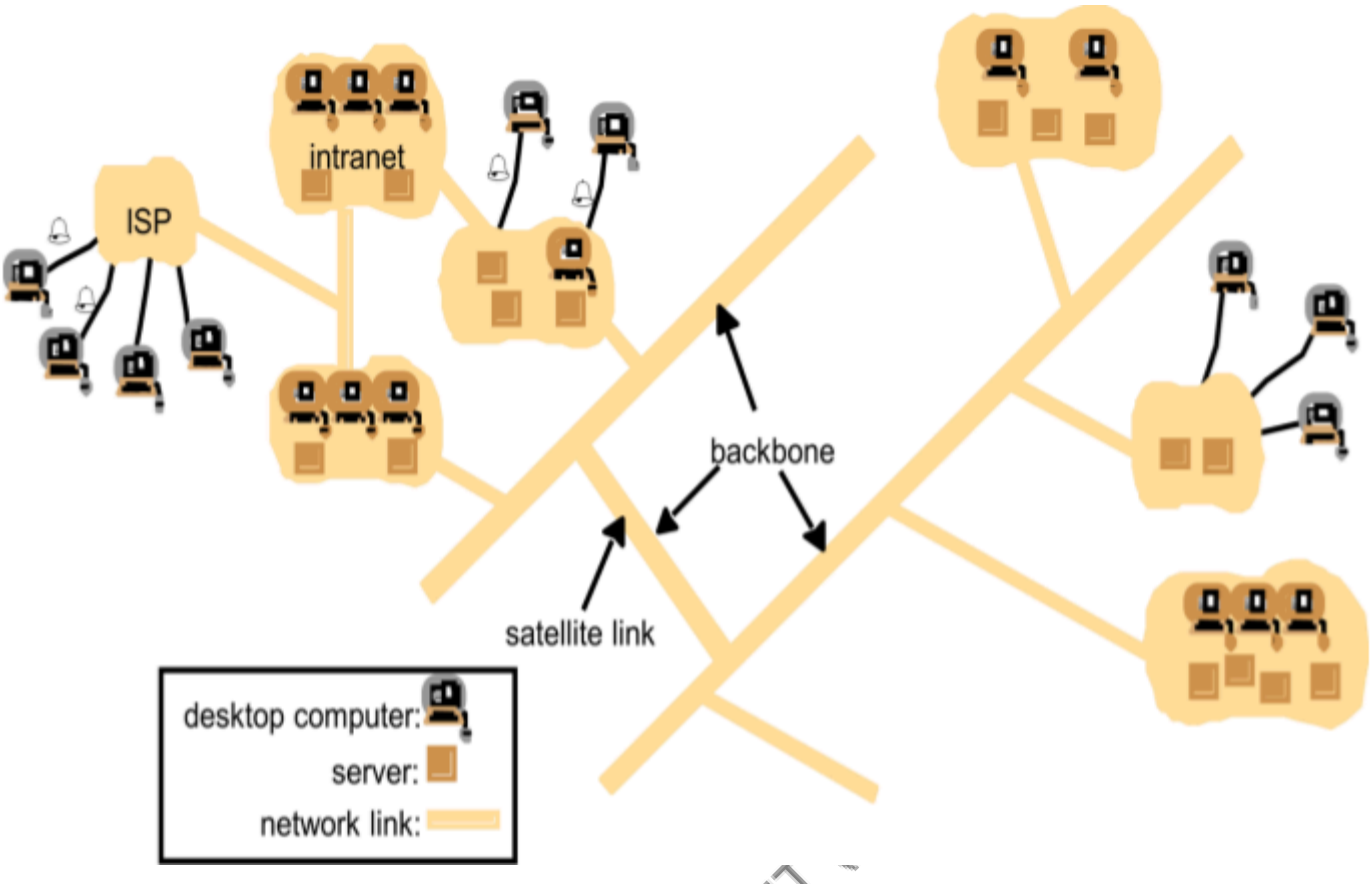
***No global clock*:** When programs need to cooperate they coordinate their actions by exchanging messages. Close coordination often depends on a shared idea of the time at which the programs' actions occur. But it turns out that there are limits to the accuracy with which the computers in a network can synchronize their clocks – 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 a network.

***Independent failures*:** All computer systems can fail, and it is the responsibility of system designers to plan for the consequences of possible failures. Distributed systems can fail in new ways. Faults in the network result in the isolation of the computers that are connected to it, but that doesn't mean that they 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 the unexpected termination of a program somewhere in the system (a *crash*), is not immediately made known to the other components with which it communicates. Each component of the system can fail independently, leaving the others still running.

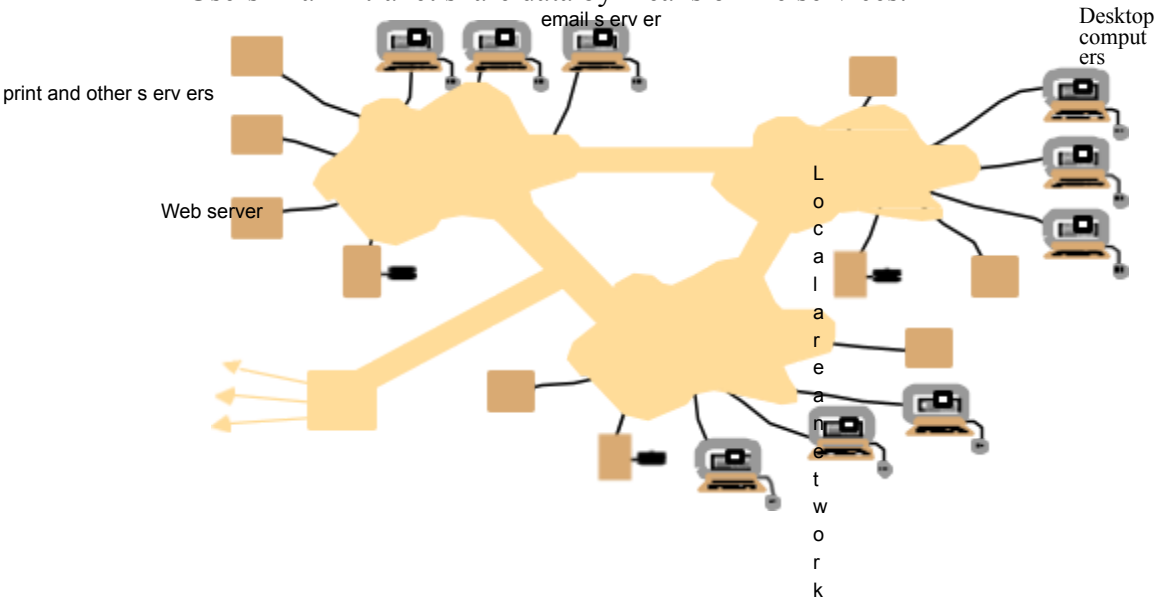
Examples of Distributed systems

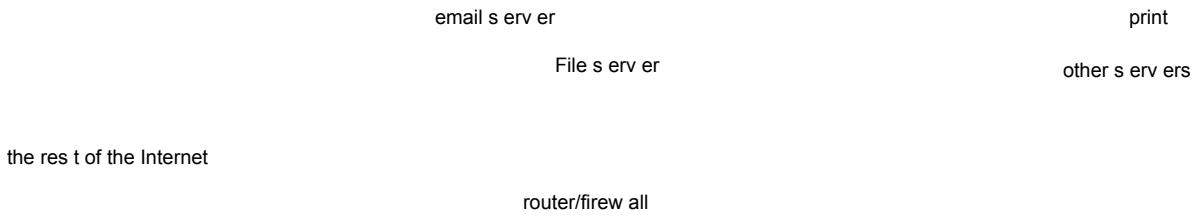
To place distributed systems in a realistic context through examples: the Internet, an intranet and mobile computing.

1. The Internet (Figure 1) :
    - A vast interconnected collection of computer networks of many different types.
    - Passing message by employing a common means of communication (Internet Protocol).
    - The web is not equal to the Internet.



2. **Intranets (Figure 2):**
    - An intranet is a private network that is contained within an enterprise.
    - It may consist of many interlinked local area networks and also use leased lines in the Wide Area Network.
    - It separately administrated and enforces local security policies.
    - It is connected to the Internet via a router
    - It uses firewall to protect an Intranet by preventing unauthorized messages leaving or entering
    - Some are isolated from the Internet
    - Users in an intranet share data by means of file services.




email s erv er
print
File s erv er
other s erv ers
the res t of the Internet
router/firew all

3. **Mobile and Ubiquitous Computing (Figure 1.3)**

   a. Distributed systems techniques are equally applicable to mobile computing involving laptops, PDAs and wearable computing devices.

   b. Mobile computing (nomadic computing) - perform of computing tasks while moving (nomadic computing)

   c. Ubiquitous computing - small computers embedded in appliances

      i. harness of many small, cheap computation devices

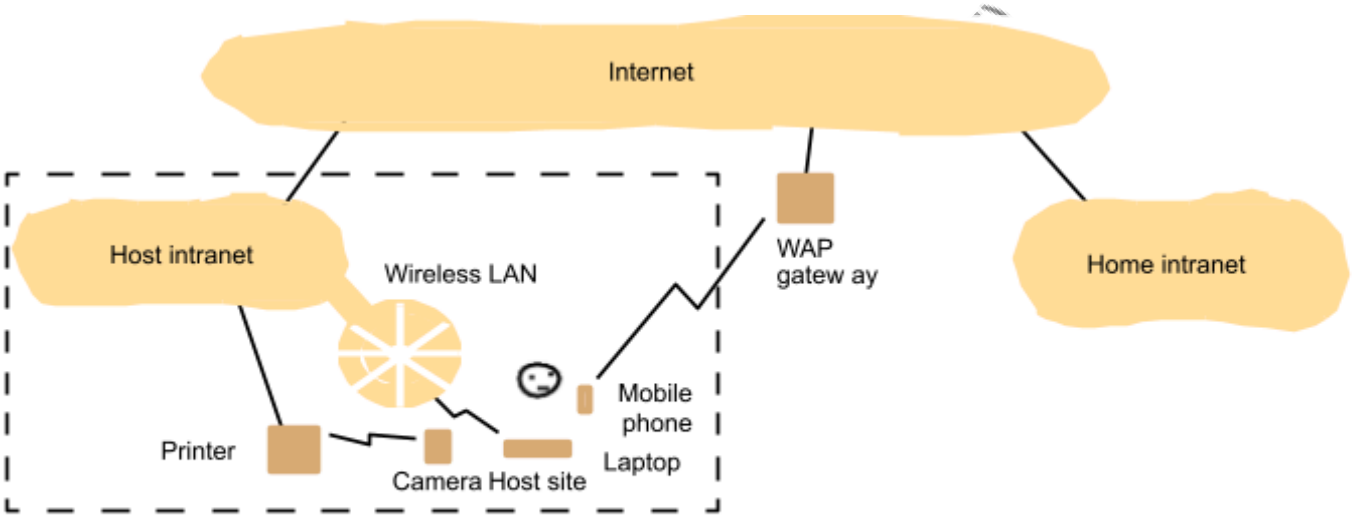
      ii. It benefits users while they remain in a single environment such as

home. Distributed In Figure 3 user has access to three forms of wireless connection**:**

   d. A laptop is connected to host's wireless LAN.

   e. A mobile (cellular) phone is connected to Internet using Wireless Application Protocol (WAP) via a gateway.

   f. A digital camera is connected to a printer over an infra-red link.



Resource Sharing and Web

- Equipments are shared to reduce cost. Data shared in database or web pages are high-level resources which are more significant to users without regard for the server or servers that provide these.
- Patterns of resource sharing vary widely in their scope and in how closely users work together:
  - Search Engine: Users need no contact between users
  - Computer Supported Cooperative Working (CSCW): Users cooperate directly share resources. Mechanisms to coordinate users' action are determined by the pattern of sharing and the geographic distribution.
- 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.
- Server is 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 a client.

- An executing web browser is a client. It communicates with a web server to request web pages from it.
- When a client invokes an operation upon the server, it is called the remote invocation.
- Resources may be encapsulated as objects and accessed by client objects. In this case a client object invokes a method upon a server object.

The World Wide Web (WWW)

- WWW is an evolving system for publishing and accessing resources and services across Internet. Web is an open system. Its operations are based on freely published communication standards and documents standards.
- Key feature: Web provides a hypertext structure among the documents that it stores. The documents contain links - references to other documents or resources. The structures of links can be arbitrarily complex and the set of resources that can be added is unlimited.
- Three main standard technological components:
    - HTML (Hypertext Makeup Language) specify the contents and layout of web pages.
        - Contents: text, table, form, image, links, information for search engine, ...;
        - Layout: text format, background, frame, ...
    - URL (Uniform Resource Location): identify a resource to let browser find it.
        - scheme : scheme-specific-location
        - http://web.cs.twsu.edu/ (HyperText Transfer Protocol)
    - URL (continued):
        - ftp://ftp.twsu.edu/ (File Transfer Protocol)
        - telnet://kirk.cs.twsu.edu (log into a computer)
        - mailto:chang@cs.twsu.edu (identify a user's email address)
    - HTTP (HyperText Transfer Protocol) defines a standard rule by which browsers and any other types of client interact with web servers. Main features:
        - Request-reply interaction
        - Content types may or may not be handled by browser - using plug-in or external helper One resource per request - Several requests can be made concurrently.
        - Simple access control
        - Services and dynamic pages
        - form - Common Gateway Interface program on server (Perl)
        - JavaScript (download from server and run on local computer)
        - Applet (download from server and run on local computer)

Challenges

As distributed systems are getting complex, developers face a number of challenges:

- Heterogeneity
- Openness
- Security
- Scalability
- Failure handling
- Concurrency
- Transparency
- Quality of service

Heterogeneity:

The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks. Heterogeneity (that is, variety and difference) applies to all of the following:

- Hardware devices: computers, tablets, mobile phones, embedded devices, etc.
- Operating System: Ms Windows, Linux, Mac, Unix, etc.
- Network: Local network, the Internet, wireless network, satellite links, etc.
- Programming languages: Java, C/C++, Python, PHP, etc.
- Different roles of software developers, designers, system managers

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. 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 to happen, standards need to be agreed and adopted – as have the Internet protocols.

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. Most middleware is implemented over the Internet protocols, which themselves mask the differences of the underlying networks, but all middleware deals with the difference in operating systems and hardware

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. Code suitable for running on one computer is not necessarily suitable for running on another because executable programs are normally specific both to the instruction set and to the host operating system.

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. In other words, distributed systems designers must hide the complexity of the systems as much as they can.

- 8 forms of transparency:
  - Access transparency – access to local an remote resources using identical operations
  - Location transparency – access to resources without knowing the physical location of the machine
  - Concurrency transparency – several processes operate concurrently without interfering each other
  - Replication transparency – replication of resources in multiple servers. Users are not aware of the replication
  - Failure transparency – concealment of faults, allows users to complete their tasks without knowing of the failures

- Mobility transparency – movement of resources and clients within a system without affecting users operations
- Performance transparency – systems can be reconfigured to improve performance by considering their loads
- Scaling transparency – systems and applications can be expanded without changing the structure or the application algorithms

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. If the well-defined interfaces for a system are published, it is easier for developers to add new features or replace sub-systems in the future. Example: Twitter and Facebook have API that allows developers to develop their own software interactively.

Concurrency

Both services and applications provide resources that can be shared by clients in a distributed system. There is therefore a possibility that 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. For an object to be safe in a concurrent environment, its operations must be synchronized in such a way that its data remains consistent. This can be achieved by standard techniques such as semaphores, which are used in most operating systems.

Security

Many of the information resources that are made available and 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)

integrity (protection against alteration or corruption),

availability for the authorized (protection against interference with the means to access the resources).

Scalability

Distributed systems must be scalable as the number of user increases. The scalability is defined by B. Clifford Neuman as

A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity

Scalability has 3 dimensions:

- Size
  - Number of users and resources to be processed. Problem associated is overloading
- Geography
  - Distance between users and resources. Problem associated is communication reliability
- Administration
  - As the size of distributed systems increases, many of the system needs to be controlled. Problem associated is administrative mess

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 intended computation. The handling of failures is particularly difficult.

- Dealing with failures in distributed systems:

- Detecting failures – known/unknown failures
- Masking failures – hide the failure from become severe. E.g. retransmit messages, backup of file data
- Tolerating failures – clients can be designed to tolerate failures – e.g. inform users of failure and ask them to try later
- Recovery from failures - recover and rollback data after a server has crashed
- Redundancy- the way to tolerate failures – replication of services and data in multiple servers

Quality of service

- The main nonfunctional properties of distributed systems that affect the quality of service experienced by users or clients are: reliability, security, performance, adaptability.
- Reliability
- Security
- Performance
- Adaptability

System Models: Introduction

- Architectural Models
  - Client-Server Model
  - Peer-Peer Model
- Fundamental Models
  - Interaction Model
  - Failure Model
  - Security Model

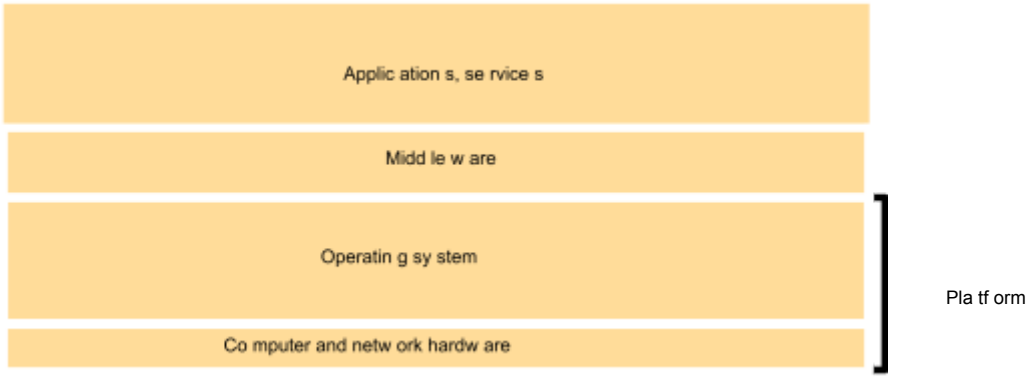
Architectural Models:

- An architectural model of a distributed system is concerned with the placement of its parts and the relationships between them.
- 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.
- Major concerns are to make the system:
  - Reliable
  - Manageable
  - Adaptable
  - Cost-effective
- An architectural Model of a distributed system first simplifies and abstracts the functions of the individual components of a distributed system.
- An initial simplification is achieved by classifying processes as:
  - Server processes
  - Client processes
  - Peer processes
    - Cooperate and communicate in a symmetric manner to perform a task.

Software Layers

- Software architecture referred to:
  - The structure of software as layers or modules in a single computer.
  - The services offered and requested between processes located in the same or different computers.

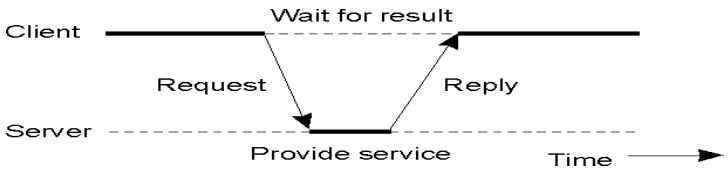
- Software architecture is breaking up the complexity of systems by designing them through layers and services.
  - Layer: a group of related functional components.
  - Service: functionality provided to the next layer.



- Platform
  - The lowest-level hardware and software layers are often referred to as a platform for distributed systems and applications.
    - These low-level layers provide services to the layers above them, which are implemented independently in each computer.
    - These low-level layers bring the system's programming interface up to a level that facilitates communication and coordination between processes.
- Middleware
  - A layer of software whose purpose is
    - to mask heterogeneity presented in distributed systems.
    - To provide a convenient programming model to application developers.
  - Major Examples of middleware are:
    - Sun RPC (Remote Procedure Calls)
    - OMG CORBA (Common Request Broker Architecture)
    - Microsoft D-COM (Distributed Component Object Model)
    - Sun Java RMI

Client-Server model

- Most often architecture for distributed systems.
- Client process interact with individual server processes in a separate host computers in order to access the shared resources
- 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.
  - E.g. a search engine can be both a server and a client: it responds to queries from browser clients and it runs web crawlers that act as clients of other web servers.



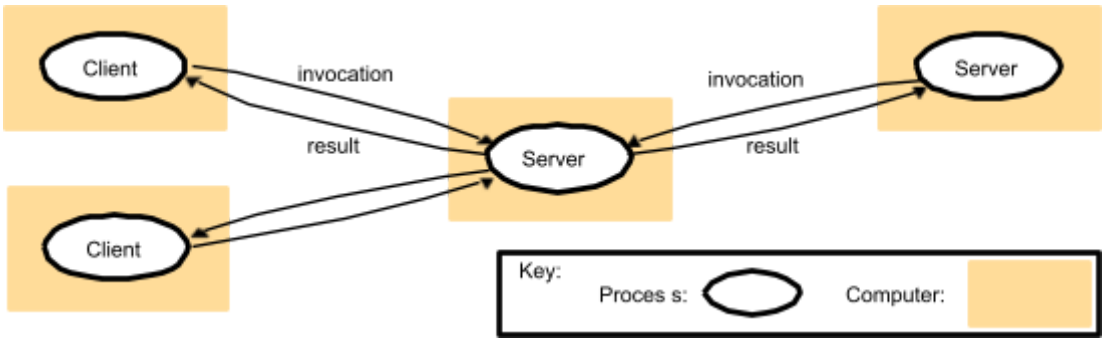
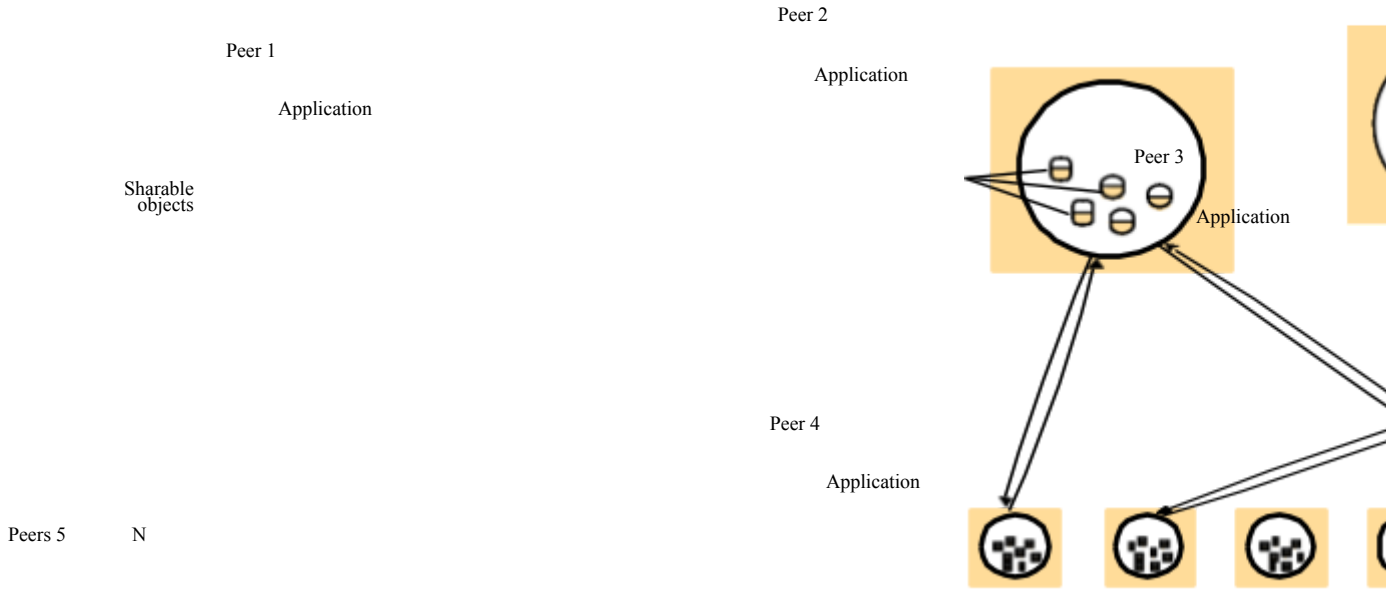


Figure 4. Clients invoke individual servers

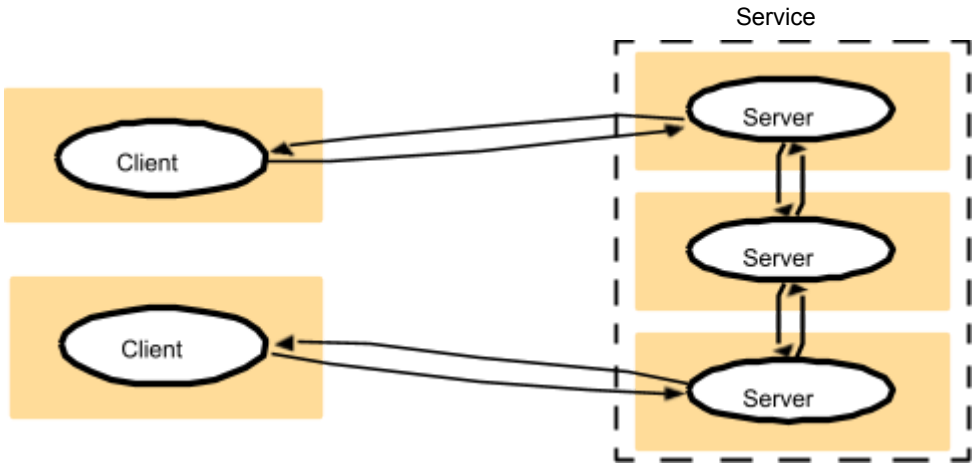
Peer-to-Peer model

- All of the processes play similar roles, interacting cooperatively as peers to perform a distributed activities or computations without any distinction between clients and servers or the computers that they run on.
- E.g., music sharing systems Napster

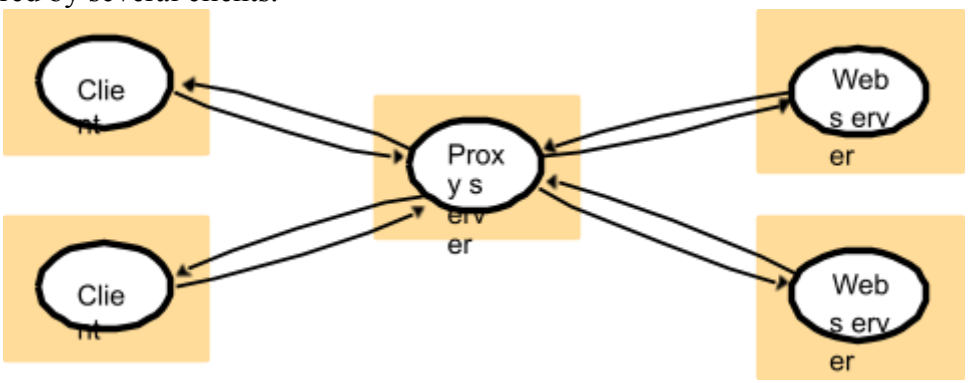


Variants of Client Sever Model

- The problem of client-server model is placing a service in a server at a single address that does not scale well beyond the capacity of computer host and bandwidth of network connections.
- To address this problem, several variations of client-server model have been proposed.
- Some of these variations are discussed in the next slide.
- **Services provided by multiple servers**
  - Services may be implemented as several server processes in separate host computers interacting as necessary to provide a service to client processes.
  - E.g. cluster that can be used for search engines.



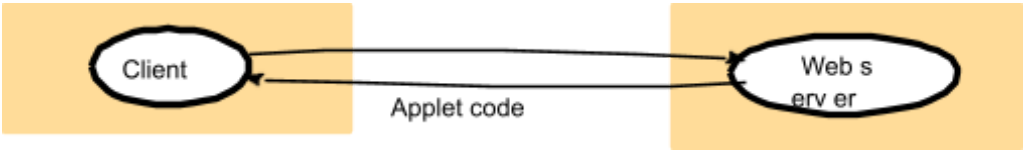
- **Proxy servers and caches**
  - A cache is a store of recently used data objects.
  - When a new object is received at a computer it is added to the cache store, replacing some existing objects if necessary.
  - When an object is needed by a client process the caching service first checks the cache and supplies the object from there if an up-to-date copy is available.
  - If not, an up-to-data copy is fetched.
  - Caches may be collected with each client or they may be located in a proxy server that can be shared by several clients.



- **Mobile code**
  - Applets are a well-known and widely used example of mobile code.
  - Applets downloaded to clients give good interactive response
  - Mobile codes such as Applets are a potential security threat to the local resources in the destination computer.
  - Browsers give applets limited access to local resources. For example, by providing no access to local user file system.

E.g. a stockbroker might provide a customized service to notify customers of changes in the prices of shares; to use the service, each customer would have to download a special applet that receives updates from the broker's server, display them to the user and perhaps performs automatic to buy and sell operations triggered by conditions set up by the customer and stored locally in the customer's computer

a) client request res ults in t he dow nloading of applet code



b) client interac ts w ith the applet

- **Mobile agents**
  - A running program (code and data) that travels from one computer to another in a network carrying out of a task, usually on behalf of some other process.
  - Examples of the tasks that can be done by mobile agents are:
    - To collecting information.
    - To install and maintain software maintain on the Computers within an organization.
    - To compare the prices of products from a number of vendors.
    - Mobile agents are a potential security threat to the resources in computers that they visit.
    - The environment receiving a mobile agent should decide on which of the local resources to be allowed to use.
    - Mobile agents themselves can be vulnerable
    - They may not be able to complete their task if they are refused access to the information they need.

Network computers

- It downloads its operating system and any application software needed by the user from a remote file server.
- Applications are run locally but the file are managed by a remote file server.
- Network applications such as a Web browser can also be run.

Thin clients

- It is 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.
- This architecture has the same low management and hardware costs as the network computer scheme.
- Instead of downloading the code of applications into the user's computer, it runs them on a compute server.
- Compute server is a powerful computer that has the capacity to run large numbers of application simultaneously.
- The compute server will be a multiprocessor or cluster computer running a multiprocessor version of an operation system such as UNIX or Windows.

Performance Issues

- Performance issues arising from the limited processing and communication capacities of computers and networks are considered under the following subheading:
  - Responsiveness

- ❖ E.g. a web browser can access the cached pages faster than the non-cached pages.
- ❖ Throughput
- ❖ Load balancing
  - ❖ E.g. using applets on clients, remove the load on the server.

Quality of service

- The ability of systems to meet deadlines.
- It depends on availability of the necessary Computing and network resources at the appropriate time.
- This implies a requirement for the system to provide guaranteed computing and communication resources that are sufficient to enable applications to complete each task on time.
  - ❖ E.g. the task of displaying a frame of video

Fundamental Models:

- Fundamental Models deal with a more formal description of the properties that are common in all of the architectural models.
- Fundamental Models are concerned with a more formal description of the properties that are common in all of the architectural models.
- All architectural models are composed of processes that communicate with each other by sending messages over a computer networks.
- Aspects of distributed systems that are discussed in fundamental models are:

Interaction model:

- Computation occurs within processes.
- The processes interact by passing messages, resulting in:
- Communication (information flow)
- Coordination (synchronization and ordering of activities) between processes
- Interaction model reflects the facts that communication takes place with delays.

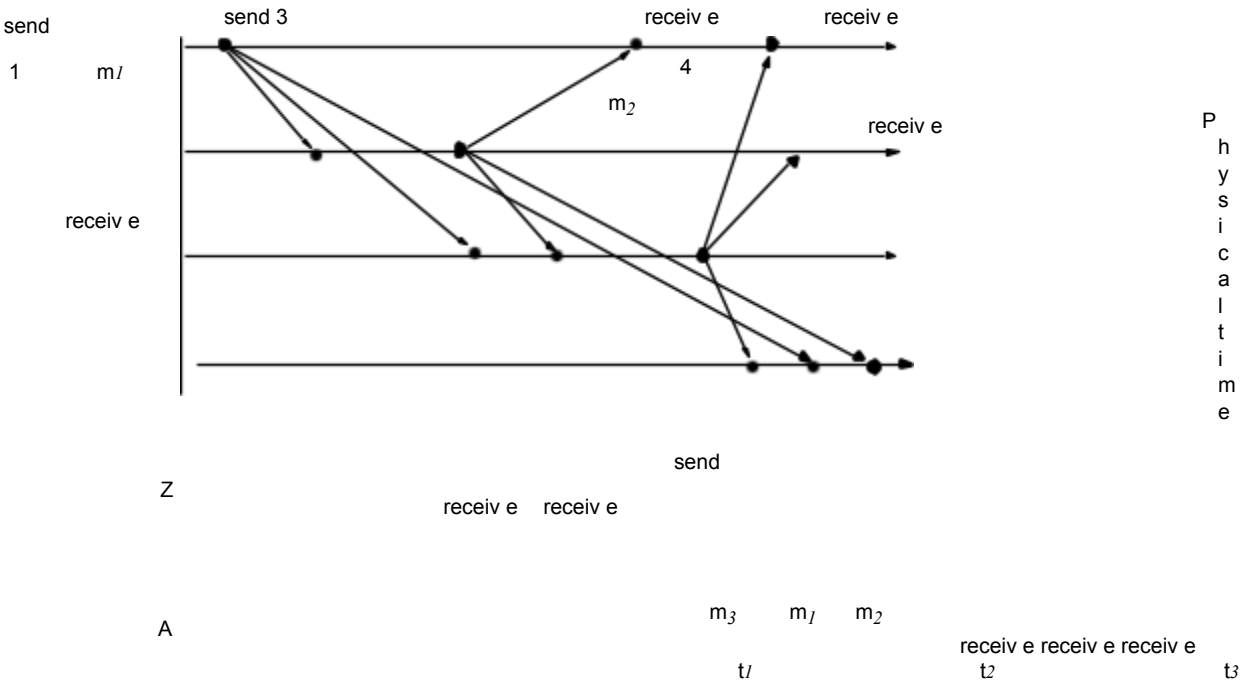
- Distributed systems are composed of many processes, interacting in the following ways:
  - Multiple server processes may cooperate with one another to provide a service
    - E.g. Domain Name Service
  - A set of peer processes may cooperate with one another to achieve a common goal
    - E.g. voice conferencing
- Two significant factors affecting interacting processes in a distributed system are:
  - Communication performance is often a limiting characteristic.
  - It is impossible to maintain a single global notion of time.
- Performance of communication channels
  - The communication channels in our model are realized in a variety of ways in distributed systems, for example
    - By an implementation of streams
    - By simple message passing over a computer network
  - Communication over a computer network has the performance characteristics such as:
    - Latency
      - The delay between the start of a message's transmission from one process to the beginning of its receipt by another.

- Bandwidth
  - The total amount of information that can be transmitted over a computer network in a given time.
  - Communication channels using the same network, have to share the available bandwidth.
- Jitter
  - The variation in the time taken to deliver a series of messages.
  - It is relevant to multimedia data.
    - For example, if consecutive samples of audio data are played with differing time intervals then the sound will be badly distorted.

- **Two variants of the interaction model**
  - In a distributed system it is hard to set time limits on the time taken for process execution, message delivery or clock drift.
  - Two models of time assumption in distributed systems are:
    - Synchronous distributed systems
      - It has a strong assumption of time
      - The time to execute each step of a process has known lower and upper bounds.
      - Each message transmitted over a channel is received within a known bounded time.
      - Each process has a local clock whose drift rate from real time has a known bound.
    - Asynchronous distributed system
      - It has no assumption about time.
      - There is no bound on process execution speeds.
        - Each step may take an arbitrary long time.
      - There is no bound on message transmission delays.
        - A message may be received after an arbitrary long time.
      - There is no bound on clock drift rates.
        - The drift rate of a clock is arbitrary.

- **Event ordering**
  - In many cases, we are interested in knowing whether an event (sending or receiving a message) at one process occurred before, after, or concurrently with another event at another process.
  - The execution of a system can be described in terms of events and their ordering despite the lack of accurate clocks.
  - For example, consider a mailing list with users X, Y, Z, and A.
  - User X sends a message with the subject Meeting.
  1. Users Y and Z reply by sending a message with the subject RE: Meeting.
  - In real time, X's message was sent first, Y reads it and replies; Z reads both X's message and Y's reply and then sends another reply, which references both X's and Y's messages.
  - But due to the independent delays in message delivery, the messages may be delivered in the order is shown in figure 10.

- It shows user A might see the two messages in the wrong order.

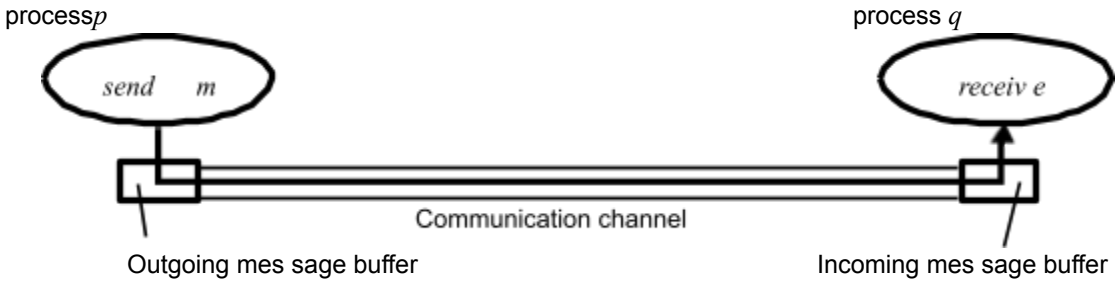


Failure model

- Failure model defines and classifies the faults.

- In a distributed system both processes and communication channels may fail – That is, they may depart from what is considered to be correct or desirable behavior.
- Types of failures:
  - Omission Failures
  - Arbitrary Failures
  - Timing Failures

- **Omission failure**
  - Omission failures refer to cases when a process or communication channel fails to perform actions that it is supposed to do.
  - The chief omission failure of a process is to crash. In case of the crash, the process has halted and will not execute any further steps of its program.
  - Another type of omission failure is related to the communication which is called communication omission failure shown in



- The communication channel produces an omission failure if it does not transport a message from "p"s outgoing message buffer to "q"'s incoming message buffer.
- This is known as "dropping messages" and is generally caused by lack of buffer space at the receiver or at an gateway or by a network transmission error, detected by a checksum carried with the message data.

- **Arbitrary failure**
  - Arbitrary failure is used to describe the worst possible failure semantics, in which any type of

error may occur.

- ❖ E.g. a process may set a wrong values in its data items, or it may return a wrong value in response to an invocation.

- Communication channel can suffer from arbitrary failures.

- ❖ E.g. message contents may be corrupted or non-existent messages may be delivered or real messages may be delivered more than once.
- ❖ The omission failures are classified together with arbitrary failures shown in

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.

- **Timing failure**
  - Timing failures are applicable in synchronized distributed systems where time limits are set on process execution time, message delivery time and clock drift rate.

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.

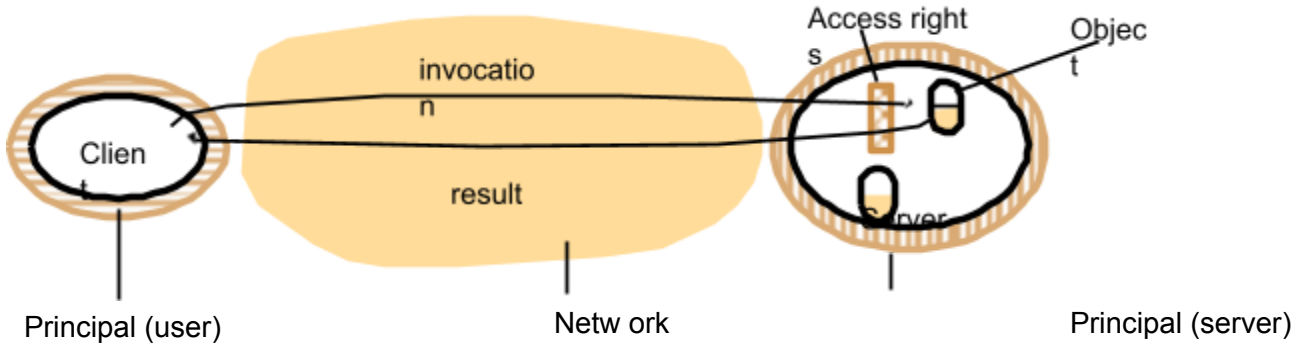
- **Masking failure**
  - It is possible to construct reliable services from components that exhibit failure.
    - ❖ E.g. multiple servers that hold replicas of data can continue to provide a service when one of them crashes.
  - A service masks a failure, either by hiding it altogether or by converting it into a more acceptable type of failure.
    - ❖ E.g. checksums are used to mask corrupted messages- effectively converting an arbitrary failure into an omission failure.

Security model

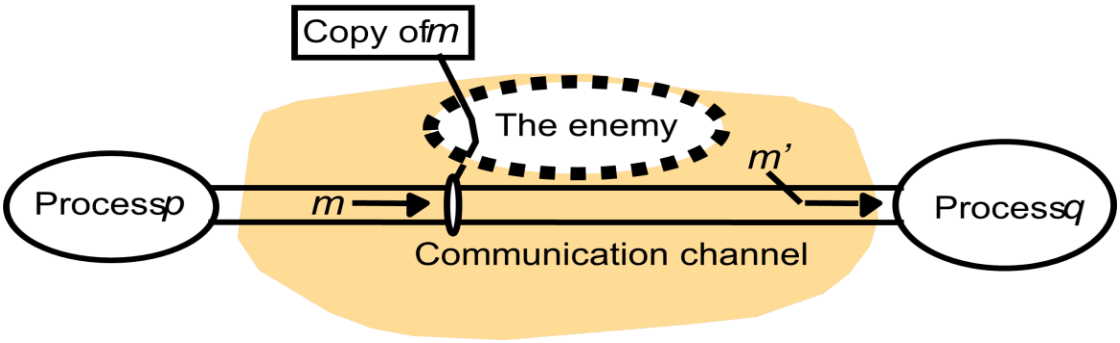
- Security model defines and classifies the forms of attacks.
- It provides a basis for analysis of threats to a system
- It is used to design of systems that are able to resist threats.

- The security of a distributed system can be achieved by securing the processes and the channels used in their interactions.
- Also, by protecting the objects that they encapsulate against unauthorized access.
- Protecting Objects
  - Access rights
    - Access rights specify who is allowed to perform the operations on a object.
      - Who is allowed to read or write its state.

- Principal
  - Principal is the authority associated with each invocation and each result.
  - A principal may be a user or a process.
  - The invocation comes from a user and the result from a server.
- The sever is responsible for
  - Verifying the identity of the principal (user) behind each invocation.
  - Checking that they have sufficient access rights to perform the requested operation on the particular object invoked.
  - Rejecting those that do not.



- The enemy
  - To model security threats, we assume an enemy that is capable of sending any message to any process and reading or copying any message between a pair of processes.



  - Threats from a potential enemy are classified as:
    - Threats to processes
    - Threats to communication channels
    - Denial of service
  - Defeating security threats
  - Secure systems are based on the following main techniques:
    - Cryptography and shared secrets
      - Cryptography is the science of keeping message secure.
      - Encryption is the process of scrambling a message in such a way as to hide its contents.
    - Authentication
      - The use of shared secrets and encryption provides the basis for the authentication of messages.
    - Secure channels
      - Encryption and authentication are use to build secure channels as a service layer on top of the existing communication services.

- A secure channel is a communication channel connecting a pair of processes, each of which acts on behalf of a principal.
- VPN (Virtual Private Network) and secure socket layer (SSL) protocols are instances of secure channel.

- A secure channel has the following properties:
  - » Each of the processes knows the identity of the principal on whose behalf the other process is executing.
  - » In a secure channel, the server knows the identity of the principal behind the invocations and can check their access rights before performing an operation.
  - » A secure channel ensures the privacy and integrity of the data transmitted across it.
  - » Each message includes a physical or logical time stamp to prevent messages from being replayed or reordered.

- Other possible threats from an enemy
  - Denial of service
    - ❖ This is a form of attack in which the enemy interferes with the activities of authorized users by making excessive and pointless invocations on services of message transmissions in a network.
    - ❖ It results in overloading of physical resources (network bandwidth, server processing capacity).
  - Mobile code
    - ❖ Mobile code is security problem for any process that receives and executes program code from elsewhere, such as the email attachment.
    - ❖ Such attachment may include a code that accesses or modifies resources that are available to the host process but not to the originator of the code