Unit-2 SYSTEM MODELS

Syllabus: Architectural Models-Software Layers, System Architecture, Variations, Interface and Objects, Design Requirements for Distributed Architectures, Fundamental models-Interaction Model, Failure ,Security Models

Topic: 01 System Model Introduction

An architectural model of a distributed system is concerned with the placement of its parts and the relationships between them.

Examples include: Client-Server model, Peer-to-Peer model

- Variations of client-sever model can be formed by:
  - The partition of data or replication at cooperative servers
  - The caching of data by proxy servers and clients
  - The use of mobile code and mobile agents
  - The requirement to add or remove mobile devices in a convenient manner
- Fundamental Models deal with a more formal description of the properties that are common in all of the architectural models.
- Some of these properties in distributed systems are:
  - There is no global time in a distributed system.
  - All communication between processes is achieved by means of messages.
- Message communication in distributed systems has the following properties:
  - Delay
  - Failure
  - Security attacks
- Message communication issues are addressed by three models:
  - Interaction Model
    - It deals with performance and with the difficulty of setting of time limits in a distributed system.
  - Failure Model
    - It attempts to give a precise specification of the faults that can be exhibited by processes and communication channels.
  - Security Model
    - It discusses possible threats to processes and communication

channels.

Topic No 02: **Architectural Models**

Architectural Models-Introduction

- 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.

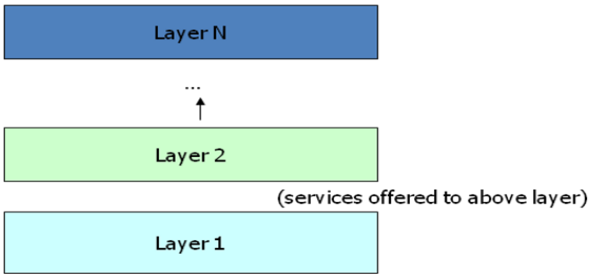


Figure 1. Software layers

- 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.

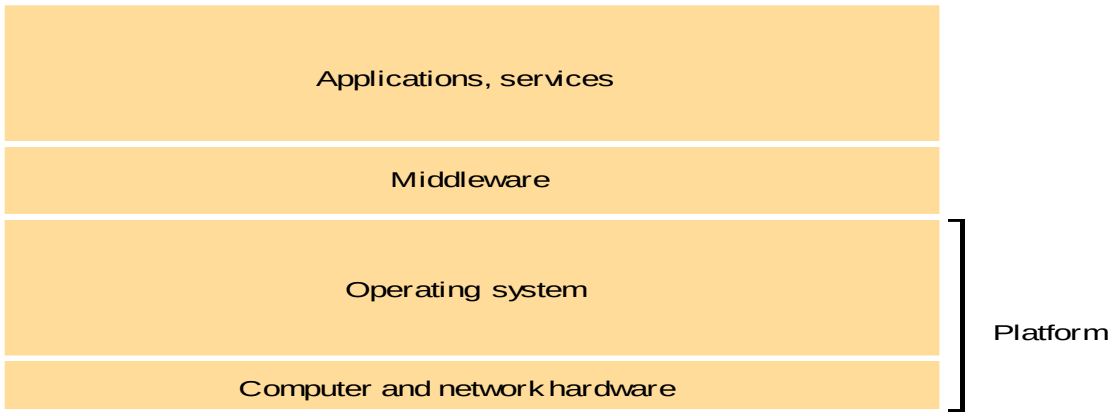


Figure. Software and hardware service layers in distributed systems

- Common examples of platform are:
  - Intel x86/Windows
  - Intel x86/Linux

- Intel x86/Solaris
- SPARC/SunOS
- PowePC/MacOS

- 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

Topic No 03: System Architectures

- The most evident aspect of distributed system design is the division of responsibilities between system components (applications, servers, and other processes) and the placement of the components on computers in the network.
- It has major implication for:
  - Performance
  - Reliability
  - Security
- 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.

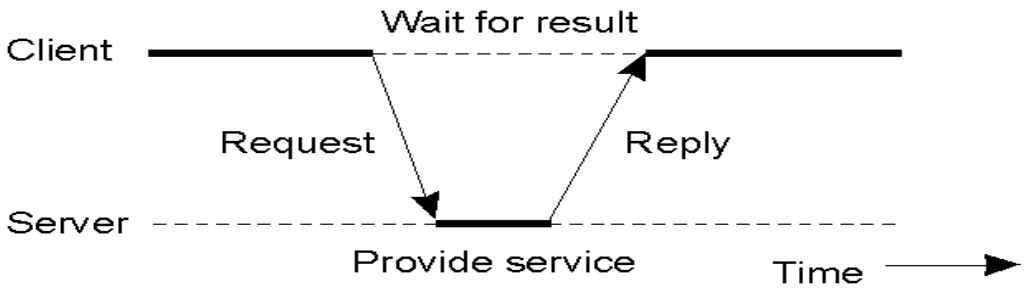


Figure. General interaction between a client and a server.

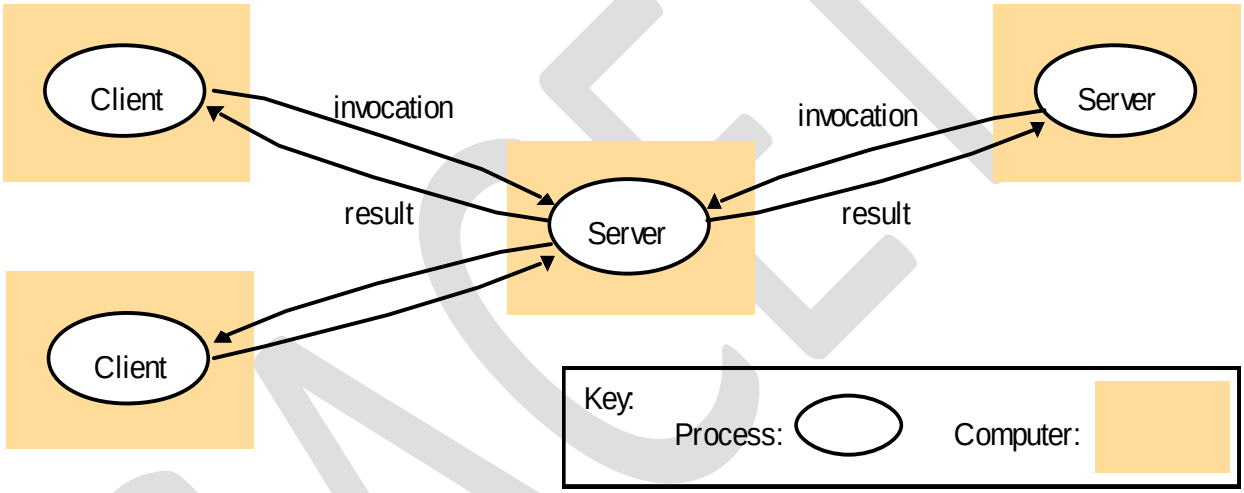


Figure. Clients invoke individual servers

- 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

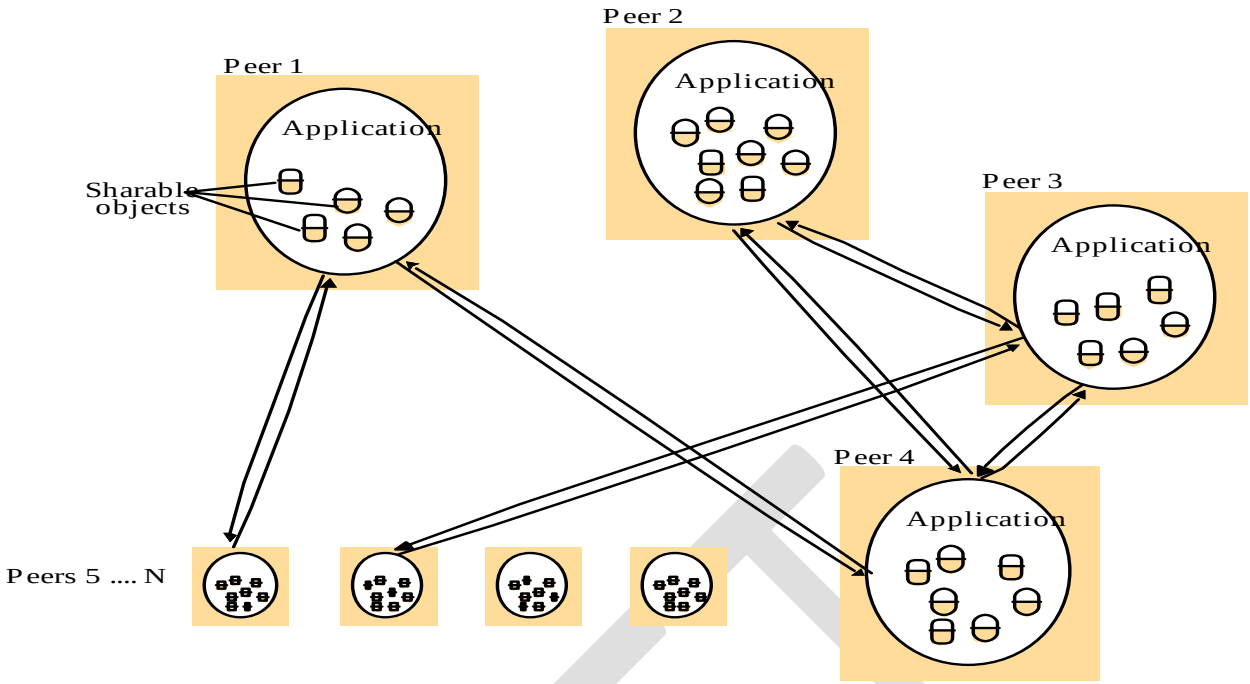


Figure. A distributed application based on the peer-to-peer architecture

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.

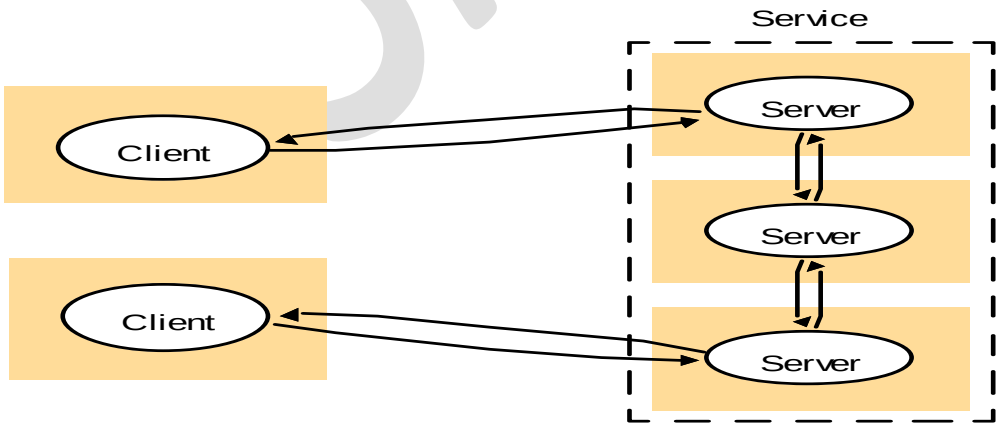


Figure. A service provided by multiple servers

- 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.

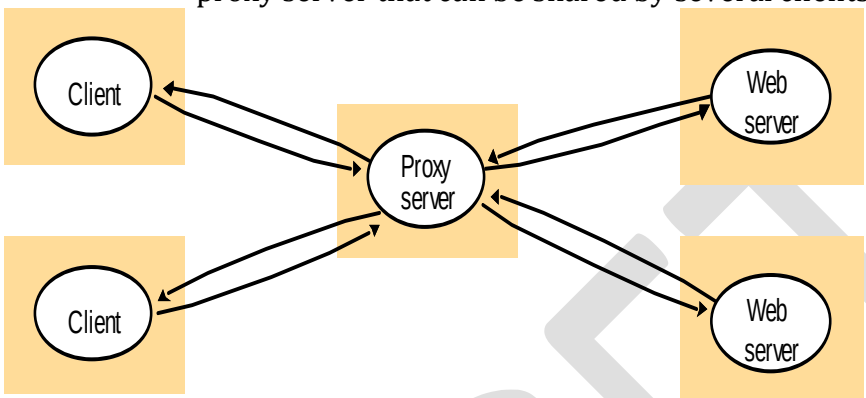
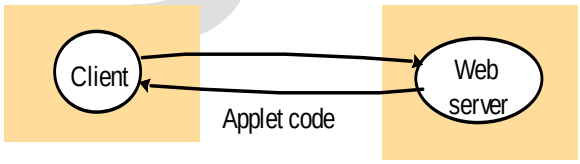


Figure. Web proxy server

- 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 results in the downloading of applet code



b) client interacts with the applet

Figure. Web applets

- 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.
  - 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.

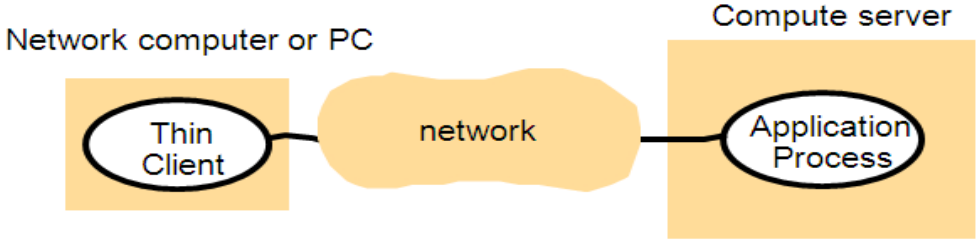


Figure. Thin clients and compute servers

- Thin client implementations
  - Thin client systems are simple in concept and their implementation is straightforward in some environment.
  - For example, most variants of UNIX include the X-11 window system.
- The X-11 window system
  - The X-11 window system is a process that manages the display and interactive input devices (keyboard, mouse) of the computer on which it runs.
  - X-11 provides an extensive library of procedures (the X-11 protocol) for displaying and modifying graphical objects in windows as well as the creation and manipulation of the windows themselves.
  - The X-11 system is referred to as a window server process.
  - The clients of X-11 server are the application programs that the user is currently interacting with.
  - The client programs communicate with the server by invoking operations in the X-11 protocol.
  - These include operations to draw text and graphical objects in windows.
- Mobile devices and spontaneous interoperation
  - Mobile devices are hardware computing components that move between physical locations and thus networks, carrying software component with them.
  - Many of these devices are capable of wireless networking ranges of hundreds of meters such as WiFi (IEEE 802.11), or about 10 meters such as Bluetooth.
  - Mobile devices include:
    - Laptops
    - Personal digital assistants (PDAs)
    - Mobile phones
    - Digital cameras
    - Wearable computers such as smart watches

Design Requirements for distributed architectures

- 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

- The main properties of the quality of the service are:
  - Reliability
  - Security
  - Performance
  - Adaptability

- Use of caching and replication
  - Distributed systems overcome the performance issues by the use of data replication and caching.
- Dependability issues
  - Dependability of computer systems is defined as:
    - Correctness
    - Security
      - Security is locating sensitive data and other resources only in computers that can be secured effectively against attack.
      - E.g. a hospital database
    - Fault tolerance
      - Dependable applications should continue to function in the presence of faults in hardware, software, and networks.
      - Reliability is achieved by redundancy.

Topic no 5: Fundamental Models

Fundamental Models-Introduction

- 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.
  - Failure model
    - Failure model defines and classifies the faults.
  - 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.

Interaction Model

- 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.

Interaction Model-Communication Channels

- 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.

Interaction Model-Computer Clock

- Computer clocks and timing events
  - Each computer in a distributed system has its own internal clock, which can be used by local processes to obtain the value of the current time.
  - Two processes running on different computers can associate timestamp with their events.
  - Even if two processes read their clock at the same time, their local clocks may supply different time.
  - This is because computer clock drift from perfect time and their drift rates differ from one another.
  - Clock drift rate refers to the relative amount that a computer clock differs from a perfect reference clock.
  - Even if the clocks on all the computers in a distributed system are set to the same time initially, their clocks would eventually vary quite significantly unless corrections are applied.
  - There are several techniques to correcting time on computer clocks.
    - For example, computers may use radio signal receivers to get readings from GPS (Global Positioning System) with an accuracy about 1 microsecond.

Interaction Model-Variations

- 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.

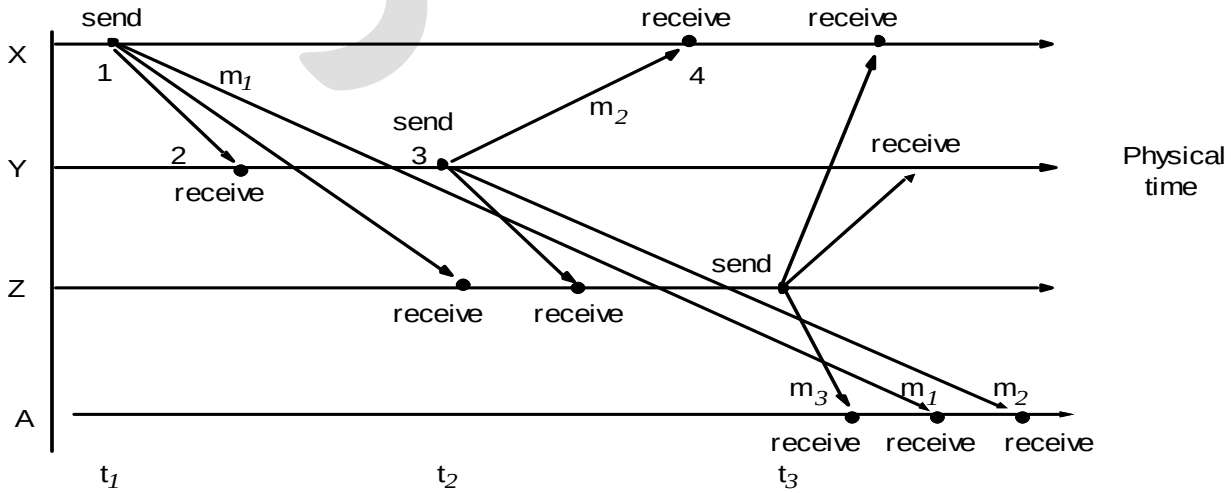


Figure. Real-time ordering of events

- Some users may view two messages in the wrong order, for example, user A might see
- *Item* is a sequence number that shows the order of receiving emails.

Item	*From*	*Subject*
23	Z	Re: Meeting
24	X	Meeting
26	Y	Re: Meeting

Failure Model

- 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 Figure.

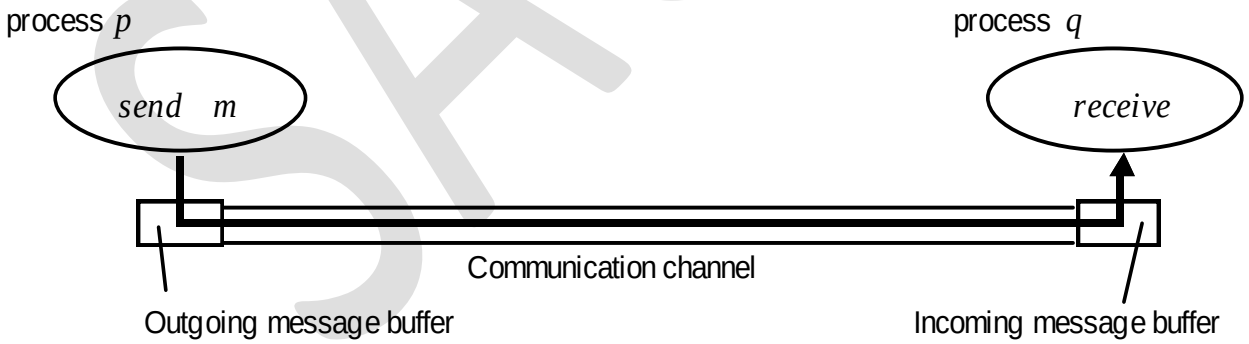


Figure: Processes and channels.

  - 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 Figure

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.

Figure: Omission and arbitrary failures

- **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.

Figure : Timing failures

- 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

- 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.

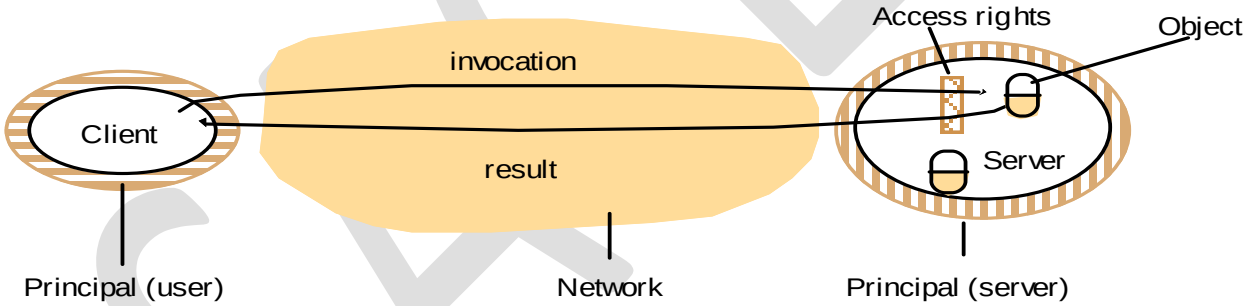


Figure . Objects and principals.

- 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.

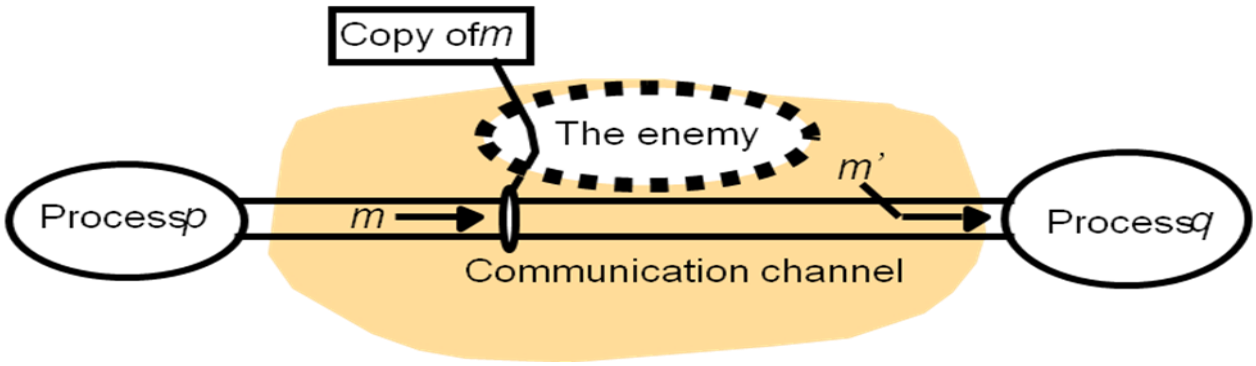


Figure . The enemy

- 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.

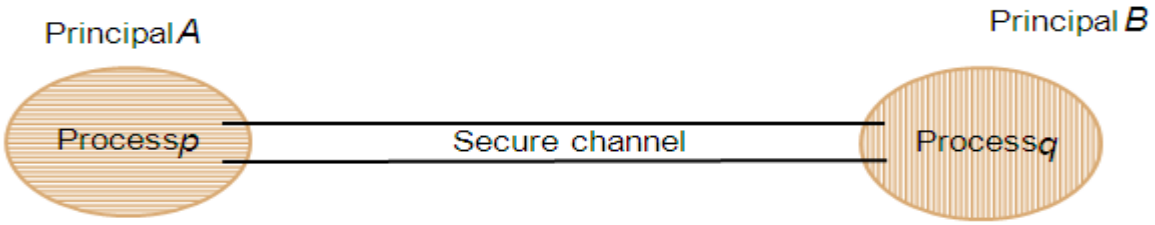


Figure. Secure channels

- 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.