Distributed Systems

By

Dr Vidya Raj C
Professor, CSE and Dean AA

Textbook:

George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair: Distributed Systems – Concepts and Design, Fifth Edition, Pearson Publications, 2012.

Introduction

- A distributed system is a collection of autonomous computers linked by a computer network and equipped with distributed system software

- Components located at networked computers can communicate and coordinate their actions only by passing messages

- Examples of modern distributed applications
  - web search, multiplayer online games and financial trading systems, mobile and ubiquitous computing

Advantages of Distributed Systems

- Economics
  - Better price/performance ratio
- Speed
  - More total computing power than CS
- Inherent distribution
  - Some applications involve spatially separated machines
- Fault tolerance
  - Better fault tolerance

Significant Consequences of DS

- **Concurrency**
  - In network of computers, concurrent program execution is the norm.
  - The capacity of the system to handle shared resources can be increased by adding more resources to the network
- **No global clock**
  - No single global notion of the correct time.
  - Difficulty in establishing the temporal order of events in DS
- **Independent failures**
  - Over the network any computer can fail
  - Faults in a network results in isolation of computers
  - Failure transparency

Example of DS

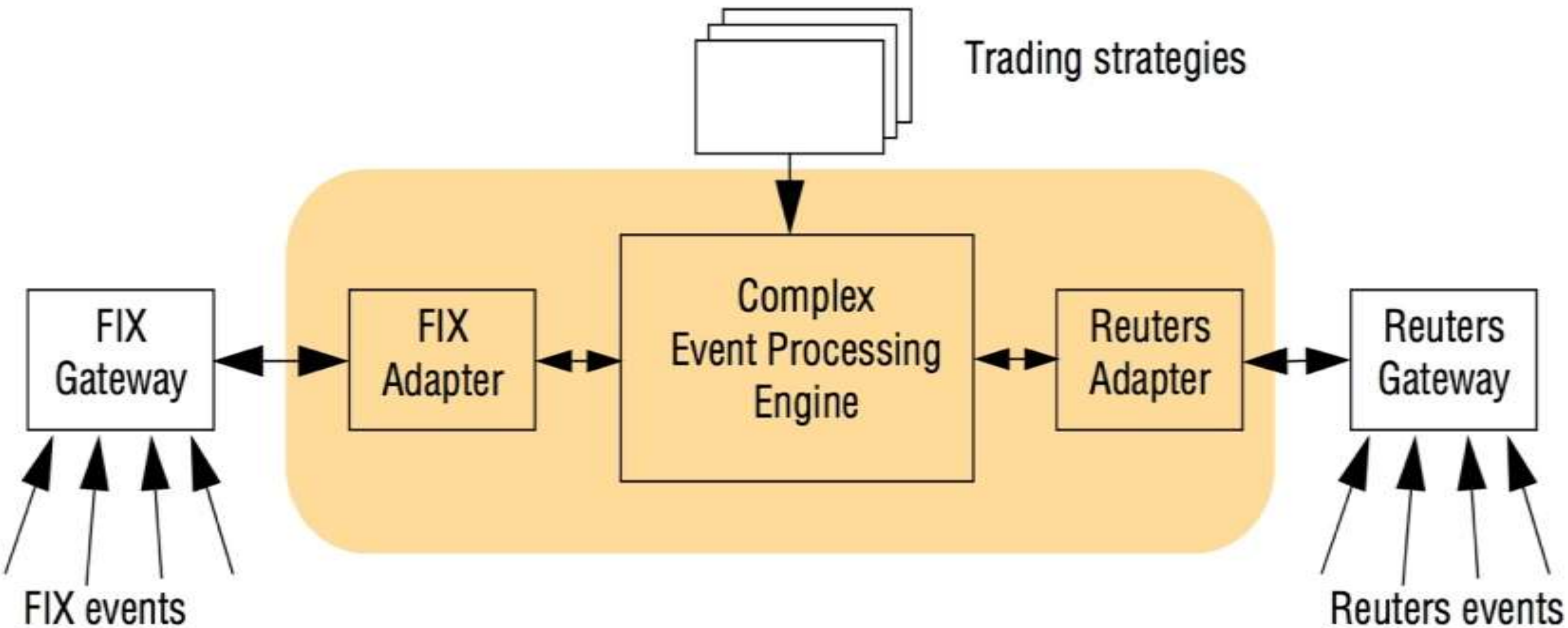
- Web search
  - The task of a web search engine is to index the entire contents of the World Wide Web, encompassing a wide range of information styles including web pages, multimedia sources and (scanned) books.

- Massively multiplayer online games (MMOGs)
  - Offer an immersive experience whereby very large numbers of users interact through the Internet with a persistent virtual world

Application domains and applications

Finance and commerce	eCommerce e.g. Amazon and eBay, PayPal, online banking and trading
The information society	Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace.
Creative industries and entertainment	online gaming, music and film in the home, user-generated content, e.g. YouTube, Flickr
Healthcare	health informatics, on online patient records, monitoring patients
Education	e-learning, virtual learning environments; distance learning
Transport and logistics	GPS in route finding systems, map services: Google Maps, Google Earth
Science	The Grid as an enabling technology for collaboration between scientists
Environmental management	sensor technology to monitor earthquakes, floods or tsunamis

An example financial trading system



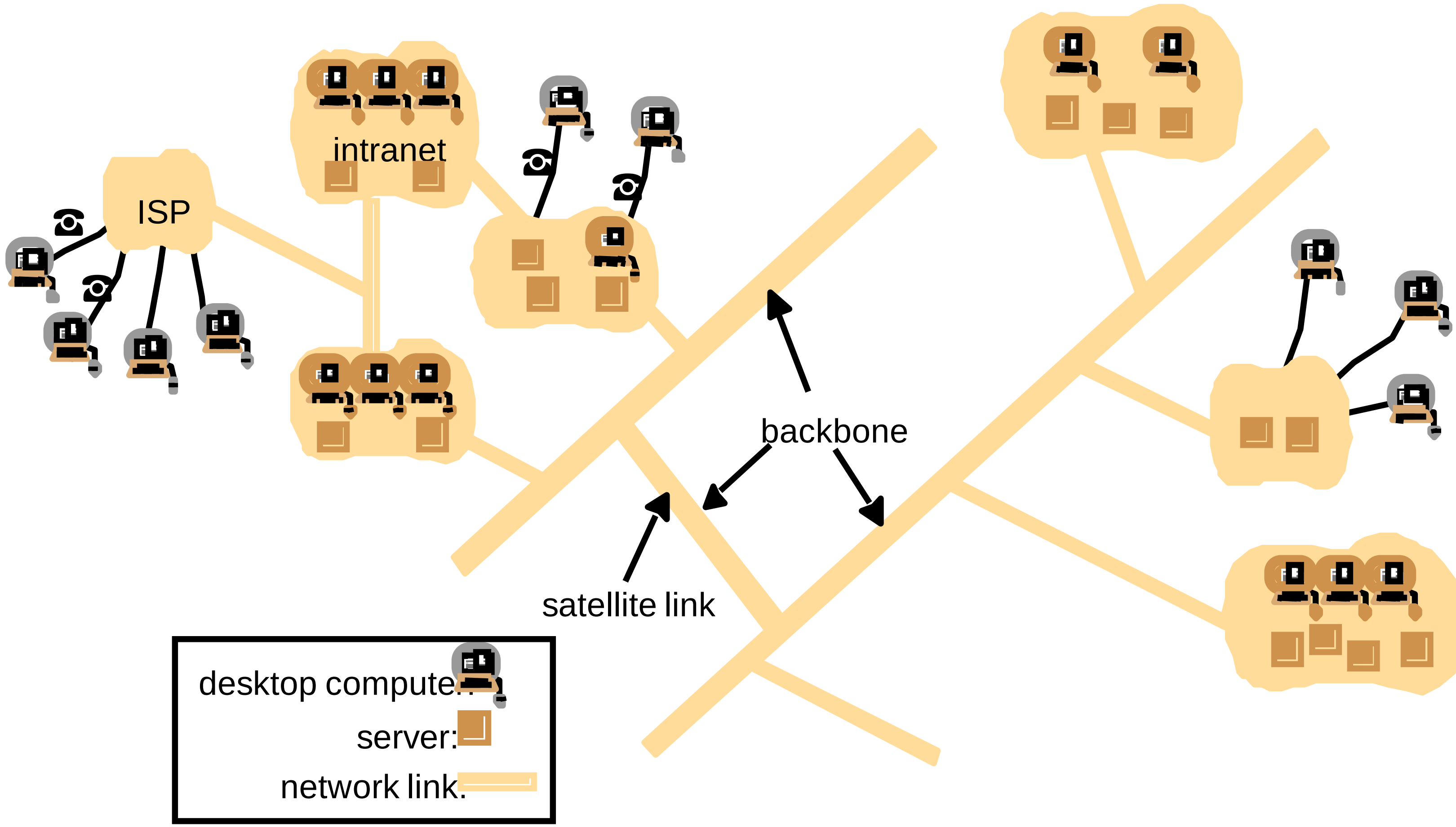
Trends in Distributed Systems

1. Pervasive networking and the modern Internet
   - Internet is a vast Distributed system
   - Network has become a pervasive resource and devices can be connected any time and anywhere

2. Mobile and ubiquitous computing
   - Device miniaturization and wireless networking
   - Portable and Wearable devices
   - Eventually becoming a part of daily life

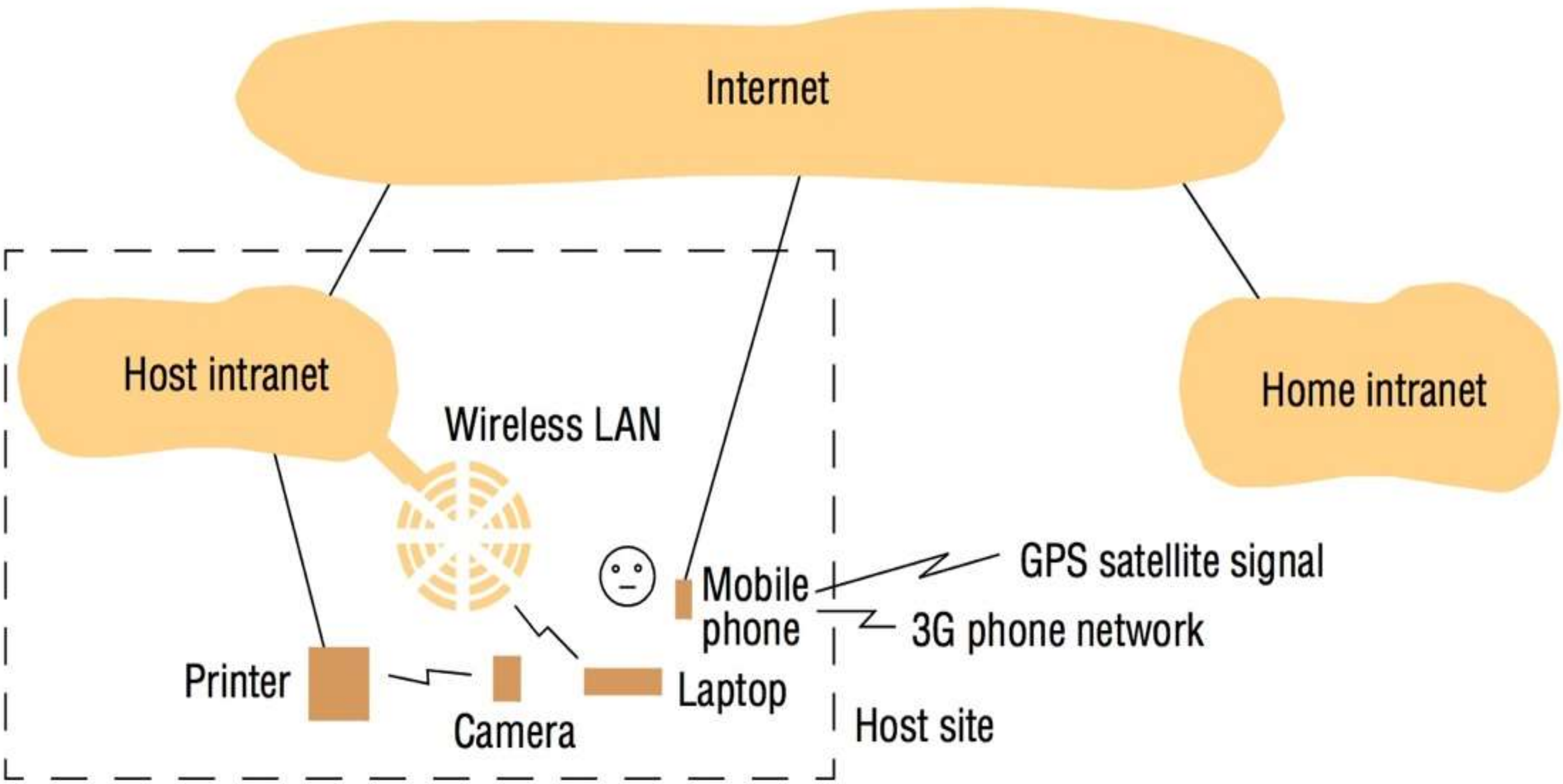
Pervasive networking and the modern Internet

A typical portion of the Internet



Mobile and ubiquitous computing

Portable and handheld devices in a DS

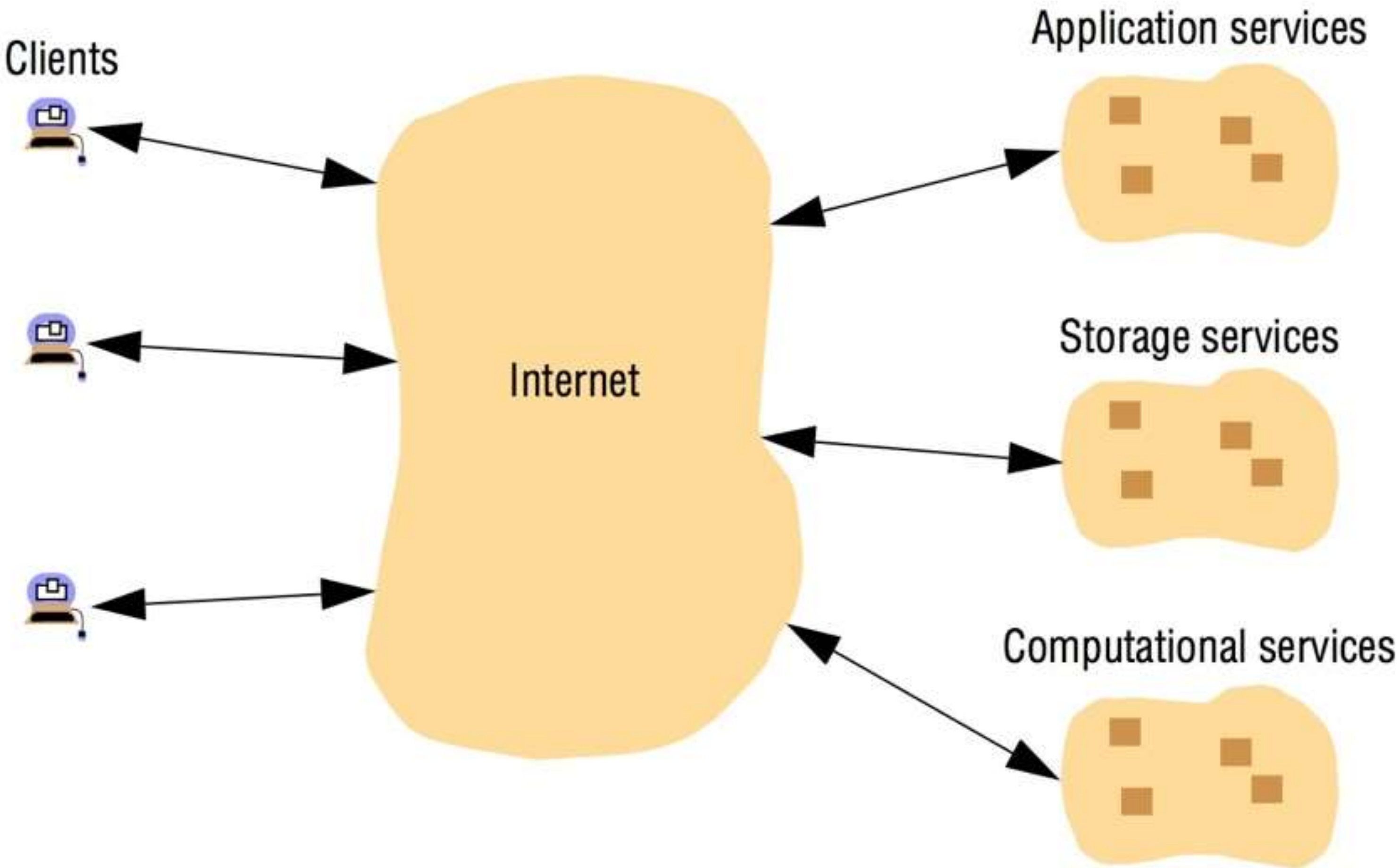


Trends in Distributed Systems

- Distributed multimedia systems
  - Distributed system to support the storage, transmission and presentation of discrete media types, such as pictures or text messages.
  - Live or pre-recorded television broadcasts, Video on demand, web casting
- Distributed computing as a utility
  - distributed resources – physical and software services are used as a commodity or utility
  - Cloud computing is used to capture this vision of computing as a utility

Distributed computing as a utility

Cloud computing



Resource Sharing

- Web pages, databases
- Hardware resources such as printers, scanners
- Data resources such as files, and resources with more specific functionality such as search engines are shared
- Computer-supported cooperative working (CSCW), a group of users who cooperate directly share resources such as documents in a small, closed group

Challenges

- Heterogeneity
- Openness
- Scalability
- Security
- Failure handling
- Concurrency
- Transparency

Heterogeneity

- Applies to hardware, OS, Programming languages, different developers
- Internet has different types of networks
- Data types with different representations
- Heterogeneity and mobile code
  - Mobile code - code that can be sent from one computer to another and run at the destination
  - ISA depends on its hardware architecture
  - Improvement - Virtual machine approach

Openness

- It is the characteristic that determine whether a system can be extended and re-implemented in various ways
- It is achieved by publishing the key interfaces
- Request for Comments – RFC, basis for technical document of Internet

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

- Design challenges
  - Controlling the cost of physical resources
  - Controlling the performance loss
  - Preventing software resources running out
  - Avoiding performance bottlenecks

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 3 components:
  - **Confidentiality :** *protection against disclosure to unauthorized individuals*
  - **Integrity**: *protection against alteration or corruption*
  - **Availability** : *protection against interference with the means to access the resources*
- Usage of Fire wall still a challenge
  - Denial of service attack, Security of mobile code

Failure handling

- 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

- Techniques for dealing with failures:
  - Detecting failures
  - Masking failures
  - Tolerating failure
  - Recovery from failures
  - Redundancy

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

- The shared resources in a DS must be responsible for ensuring that it operates correctly in the concurrent environment, using synchronization mechanisms such as semaphores, which are used in most operating systems

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 reference model for Open Distributed Processing defines 8 types of transparencies:
  - Access transparency, Location, Concurrency, Replication, Failure, Mobility, Performance, Scaling transparency

Transparency

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

- **Physical models**
  - Describe the types of computers and devices that constitute a system and their interconnectivity, without details of specific technologies
- **Architectural models**
  - Defines the way the components of the system interact with one another and their mapping with the underlying network
  - Computational elements include individual computers, servers, clients or aggregates of them supported by appropriate network interconnections.
  - Client-server and peer-to-peer are two of the most commonly used forms of architectural model for distributed systems
- **Fundamental models**
  - Concerned with formal description of the properties that are common in all architectural models
  - Specify design issues, difficulties and threats faced during the development and describe solutions to individual issues faced by most distributed systems
  - 3 models: the interaction model, failure model and security model

Physical models

- Representation of the underlying hardware elements of a distributed system that abstracts away from specific details of the computer and networking technologies employed

- **Baseline physical model**: is one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages

- Three Generations of distributed systems: Early, Internet-scale, Contemporary

Generations of distributed systems

Distributed systems:	*Early*	*Internet-scale*	*Contemporary*
Scale	Small	Large	Ultra-large
Heterogeneity	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
Openness	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
Quality of service	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

Architectural models

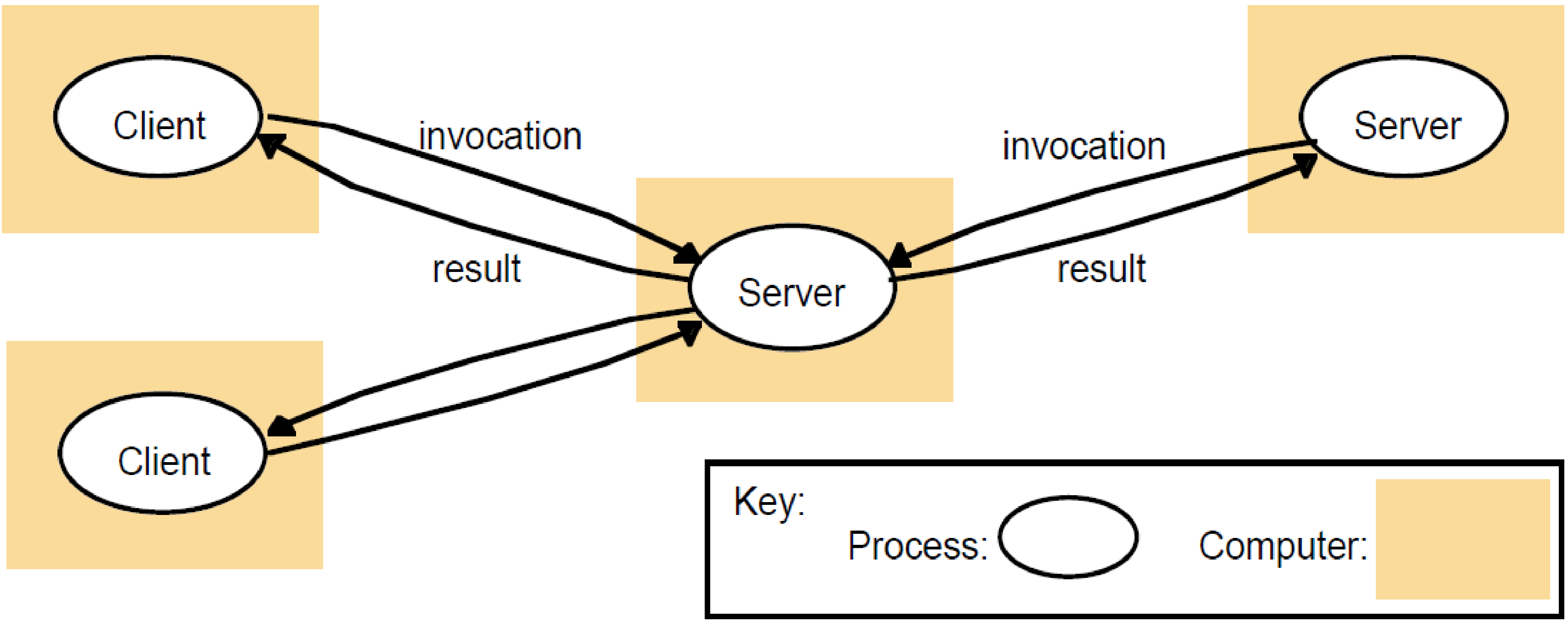
- The architecture of a system is its structure in terms of separately specified components and their interrelationships

- Two important models
  - Client-Server model: Clients interact with the server processes to access the shared resources that the server manages
  - Peer-to-Peer model: All process play similar role, no distinction between client and server

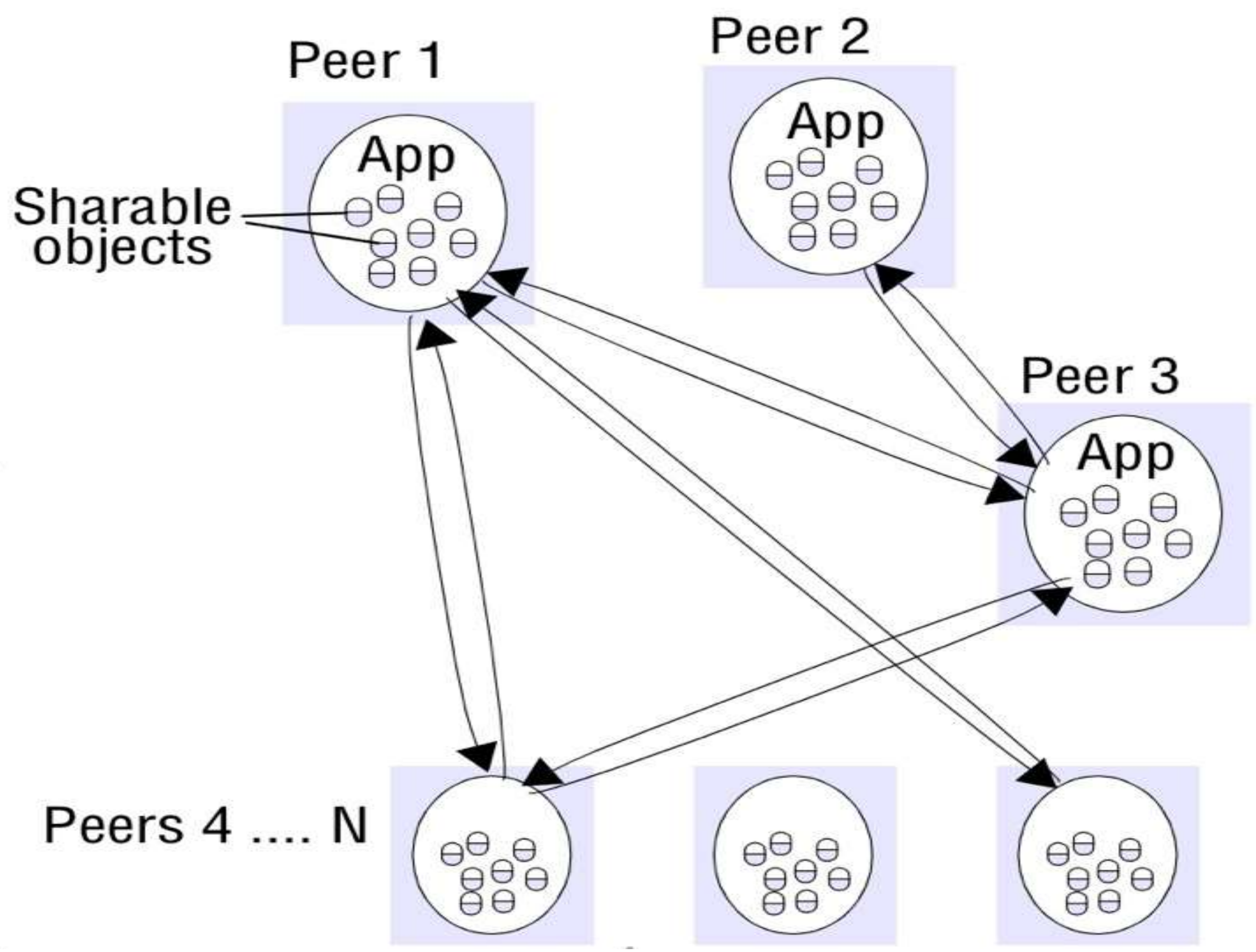
Communicating entities & Communication paradigms

Communicating entities (what is communicating)		*Communication paradigms (how they communicate)*		
System-oriented entities	*Problem-oriented entities*	*Interprocess communication*	*Remote invocation*	*Indirect communication*
Nodes	Objects	Message passing	Request-reply	Group communication
Processes	Components	Sockets	RPC	Publish-subscribe
	Web services	Multicast	RMI	Message queues
				Tuple spaces
				DSM

Client- server architecture:
Clients invoke individual servers



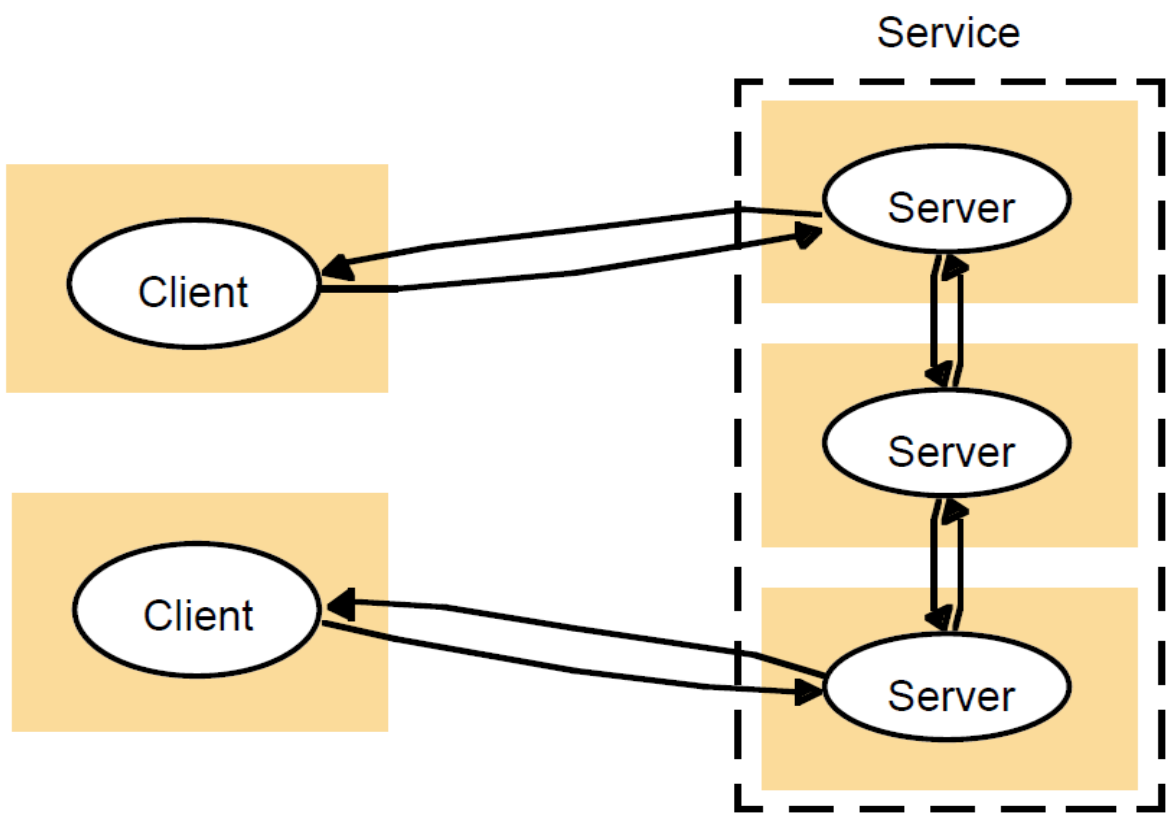
Peer-to-Peer architecture



Peer-to-Peer architecture:

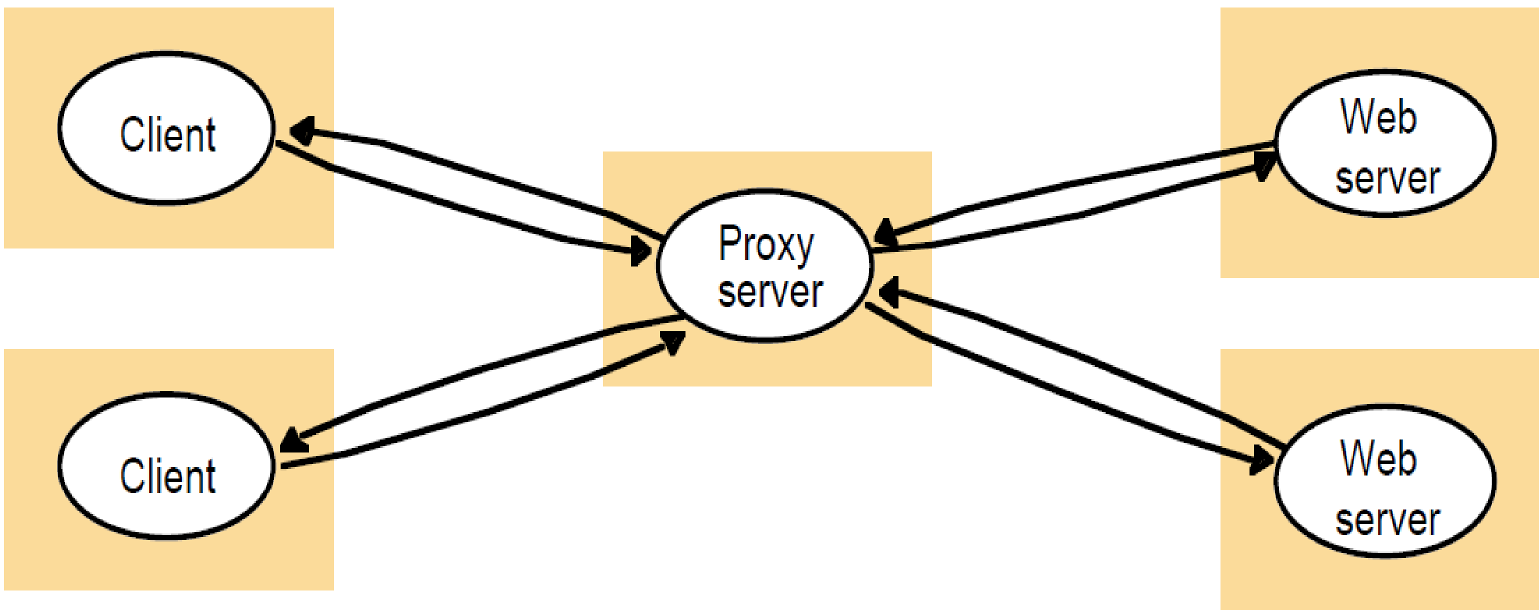
A ***service provided by multiple servers*** may be implemented as several server processes in separate host computers interacting as necessary to provide a service to client processes

Ex: Web, Sun Network Information Service



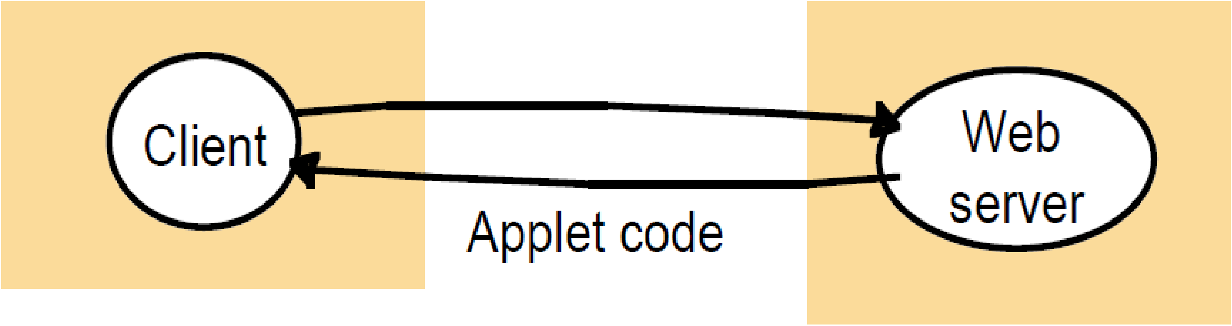
Web proxy server:

provide a shared cache of web resources for the client machines at a site or across several sites to increase the availability and performance of the service by reducing the load on the wide area network and web servers



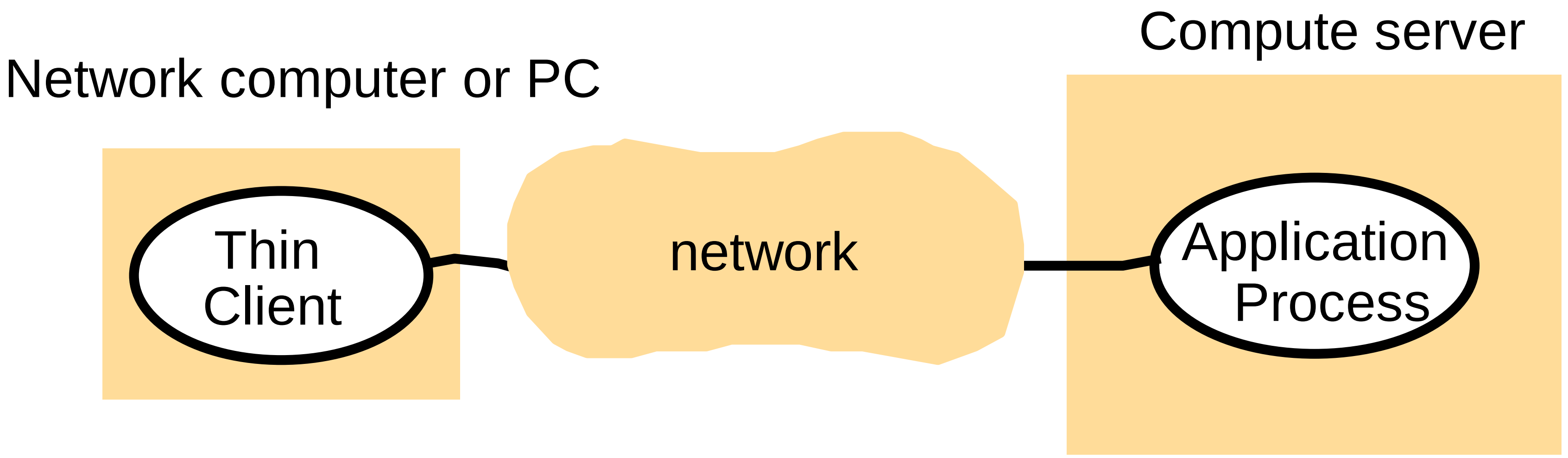
Web applets

a) client request results in the downloading of applet code



b) client interacts with the applet

Thin clients and compute servers



Fundamental models

- Models based on the fundamental properties that allow us to be more specific about their characteristics and the failures and security risks they might exhibit

- ***Interaction:*** *Computation occurs within processes; the processes interact by passing* messages, resulting in communication and coordination between processes.
- ***Failure:*** *Defines and classifies the faults. This provides a basis for the analysis of their potential effects and for the design of systems that are able to tolerate faults of each type while continuing to run correctly*
- ***Security****: S*ecurity model defines and classifies the forms that such attacks may take, providing a basis for the analysis of threats to a system and for the design of systems that are able to resist them.

Interaction model – 2 variants

- *Synchronous distributed systems:*
  - 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 systems:*
  - Process execution speeds may take an arbitrarily long time
  - Message transmission delays
  - Clock drift rates

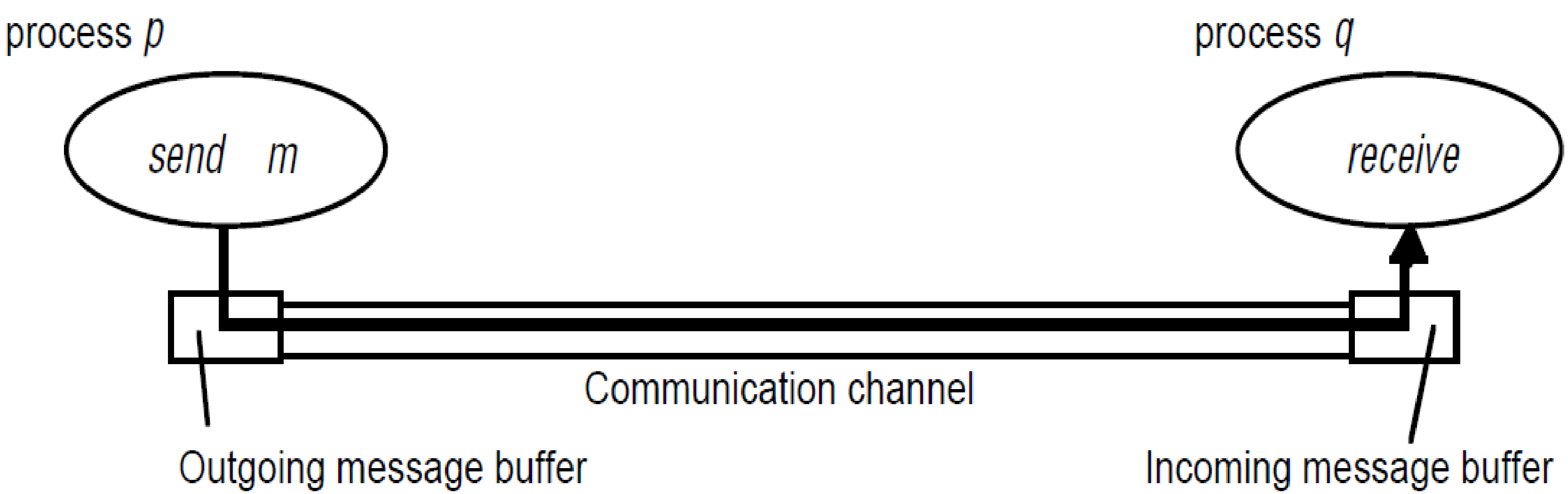
Failure model

- In a distributed system both processes and communication channels may fail:

- **Omission failures -** The faults classified as *omission failures refer to cases when a* process or communication channel fails to perform actions that it is supposed to do

- **Arbitrary failures -** The term *arbitrary or Byzantine failure is used to describe the worst* possible failure semantics, in which any type of error may occur.

Processes and channels



Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Fail-stop	**Proce**ss	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 behavior: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Timing failures

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

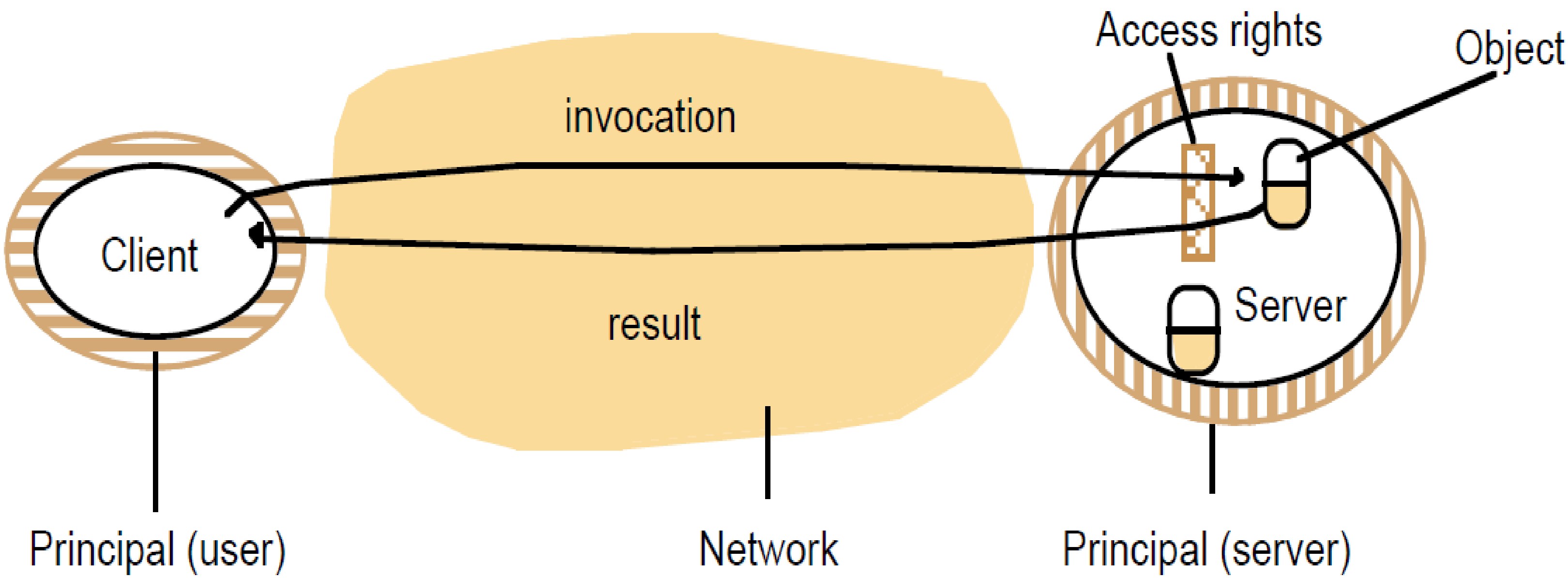
Timing failures

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.

Security model

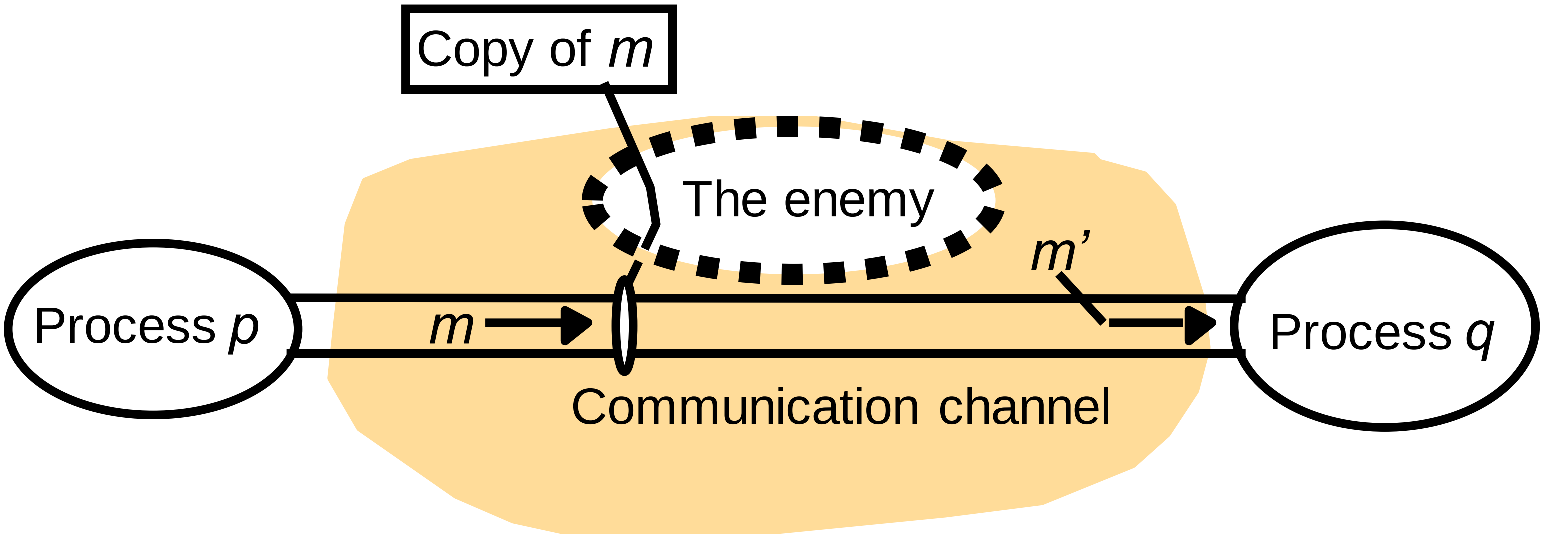
- The security of a distributed system can be achieved by securing the processes and the channels used for their interactions and by protecting the objects that they encapsulate against unauthorized access

Objects and principals



The enemy

The threats from a potential enemy include *threats to processes and threats to communication channels.*



Secure Channel

is a communication channel connecting a pair of communication processes, each of which acts on behalf of a principal