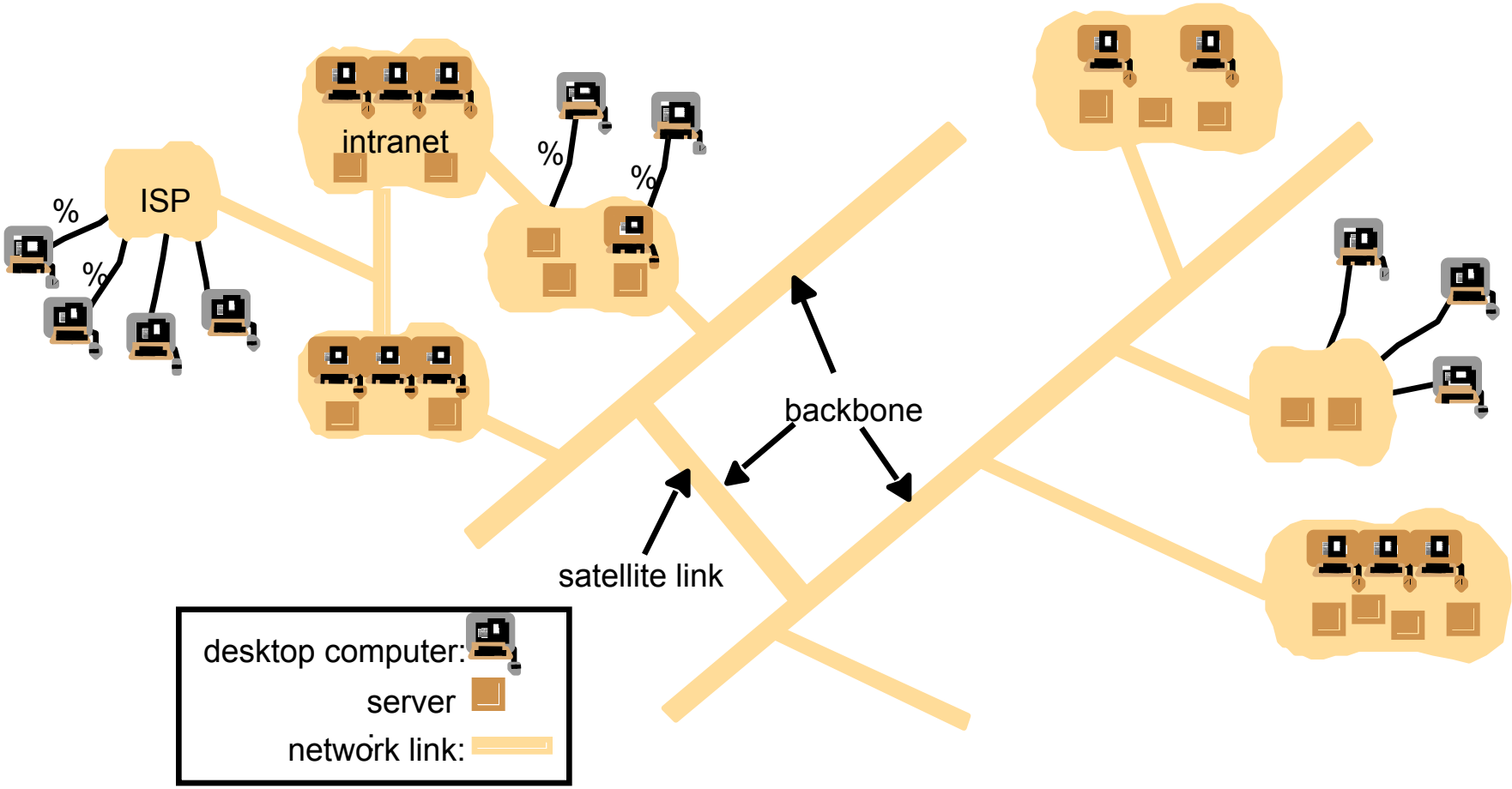
Characterization of Distributed Systems

Introduction

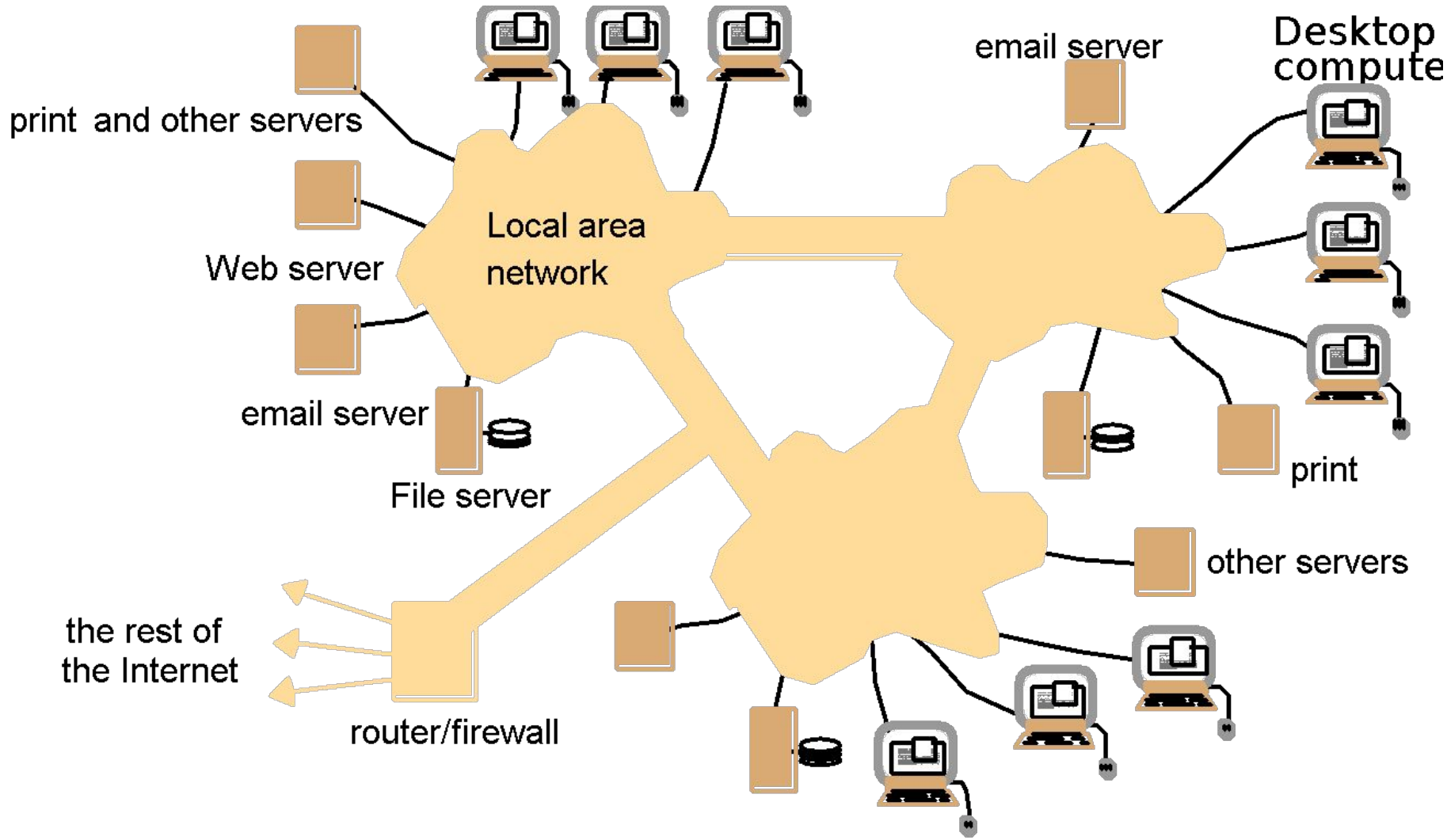
- Distributed system
  - Definition: A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages.
- Consequences:
  - Concurrency
  - No global clock
  - Independent failures

Examples of Distributed Systems (1/3)



The Internet

Examples of Distributed Systems (2/3)

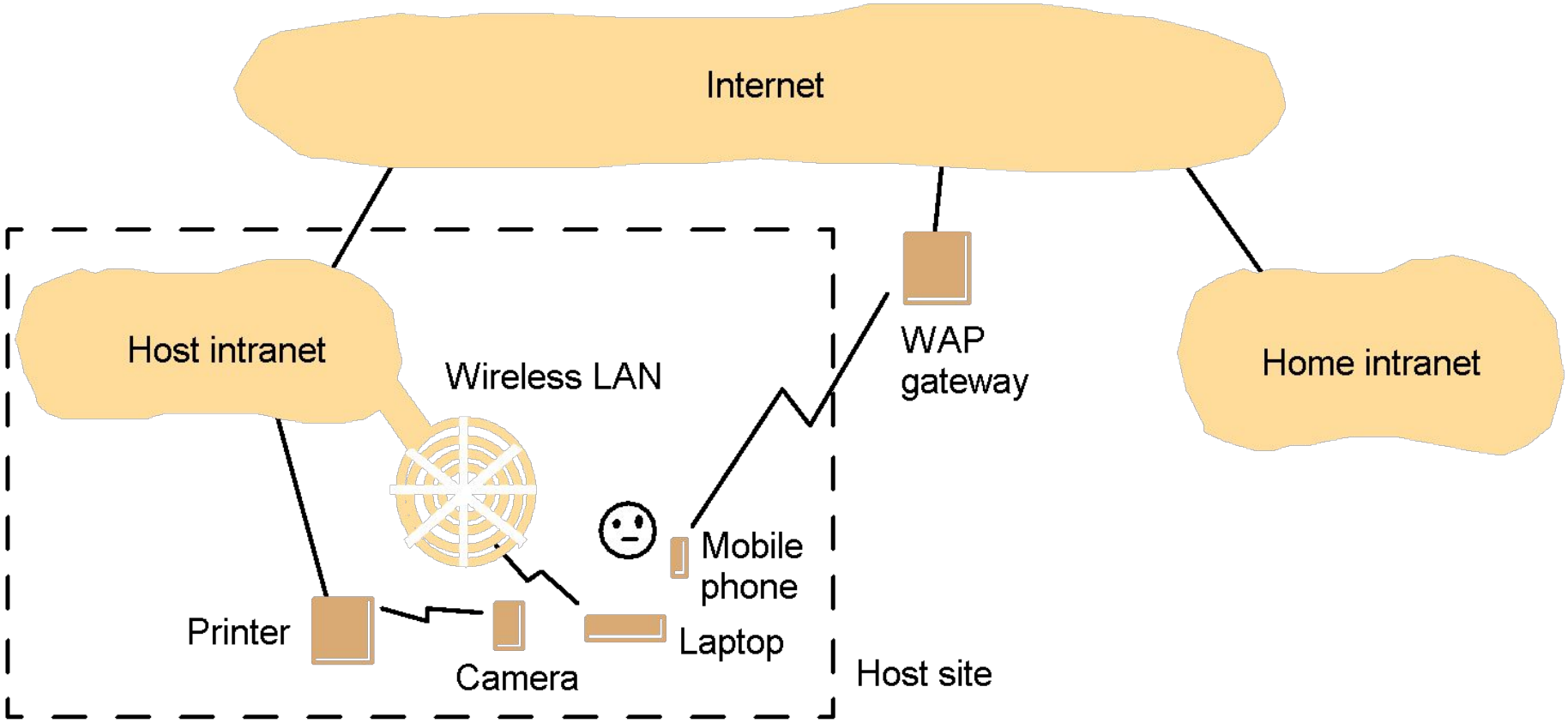


The Intranet

Examples of Distributed Systems (3/3)

- The role of a firewall is to protect an intranet by preventing unauthorized message leaving or entering
- Design of Components for Use in Intranet
  - File services are needed enable users to share data
  - Firewalls tend to impede legitimate access to services
  - The cost of software installation and support is an important issue

Mobile and Ubiquitous Computing

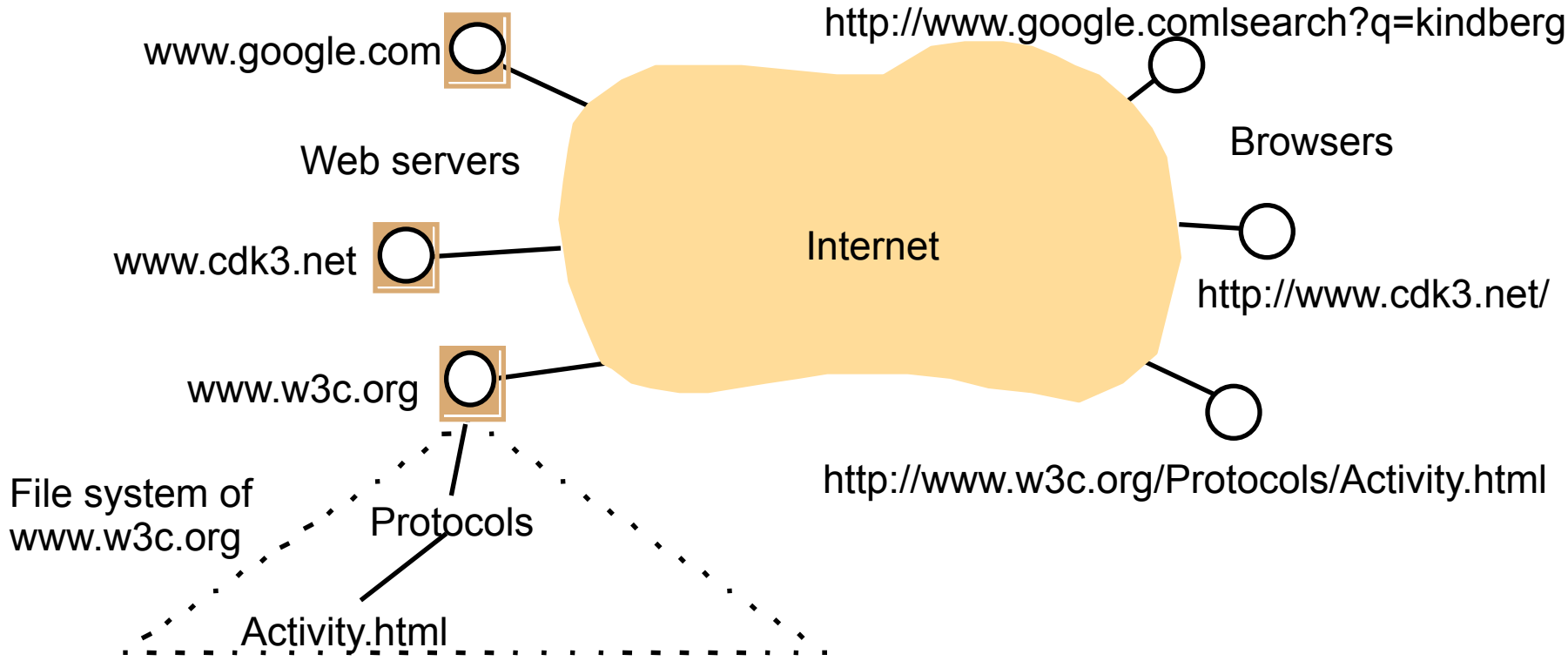


Portable and handheld devices in a distributed system

Resource Sharing and the Web

- The World Wide Web
- HyperText Transfer Protocol (HTTP)
  - HyperText Markup Language (HTML)
- Uniform Resource Locators (URLs)

Web servers and web browsers



Challenges

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

Heterogeneity

- Heterogeneity
  - Networks
  - Computer hardware
  - Operating systems
  - Programming languages
  - Implementations by different developers
- Middleware
- Heterogeneity and mobile code
  - Java applets
  - Virtual machine

Openness

Security

- Encryption can be used to provide adequate protection of shared resources and to keep sensitive information secret when is transmitted in messages over a network.
- Denial of service attacks
  - This can be achieved by bombarding the service with such a large number of pointless requests that serious users are unable to user it.
- Security of mobile code
  - Executable program as an electronic mail attachment

Scalability

- 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.
- Controlling the performance loss
- Preventing software resources running out
  - IPv4 (32 bits address) -> IPv6 (128 bits address)
- Avoiding performance bottlenecks

Failure Handling

- Detecting failures
- Masking failures
  - Messages can be retransmitted when they fail to arrive
  - File data can be written to a pair
- Tolerating failures
- Recovery from failures
- Redundancy

Concurrency

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.

Transparencies

Access transparency: enables local and remote resources to be accessed using identical operations.
Location transparency: enables resources to be accessed without knowledge of their location.
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.
Performance transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms.

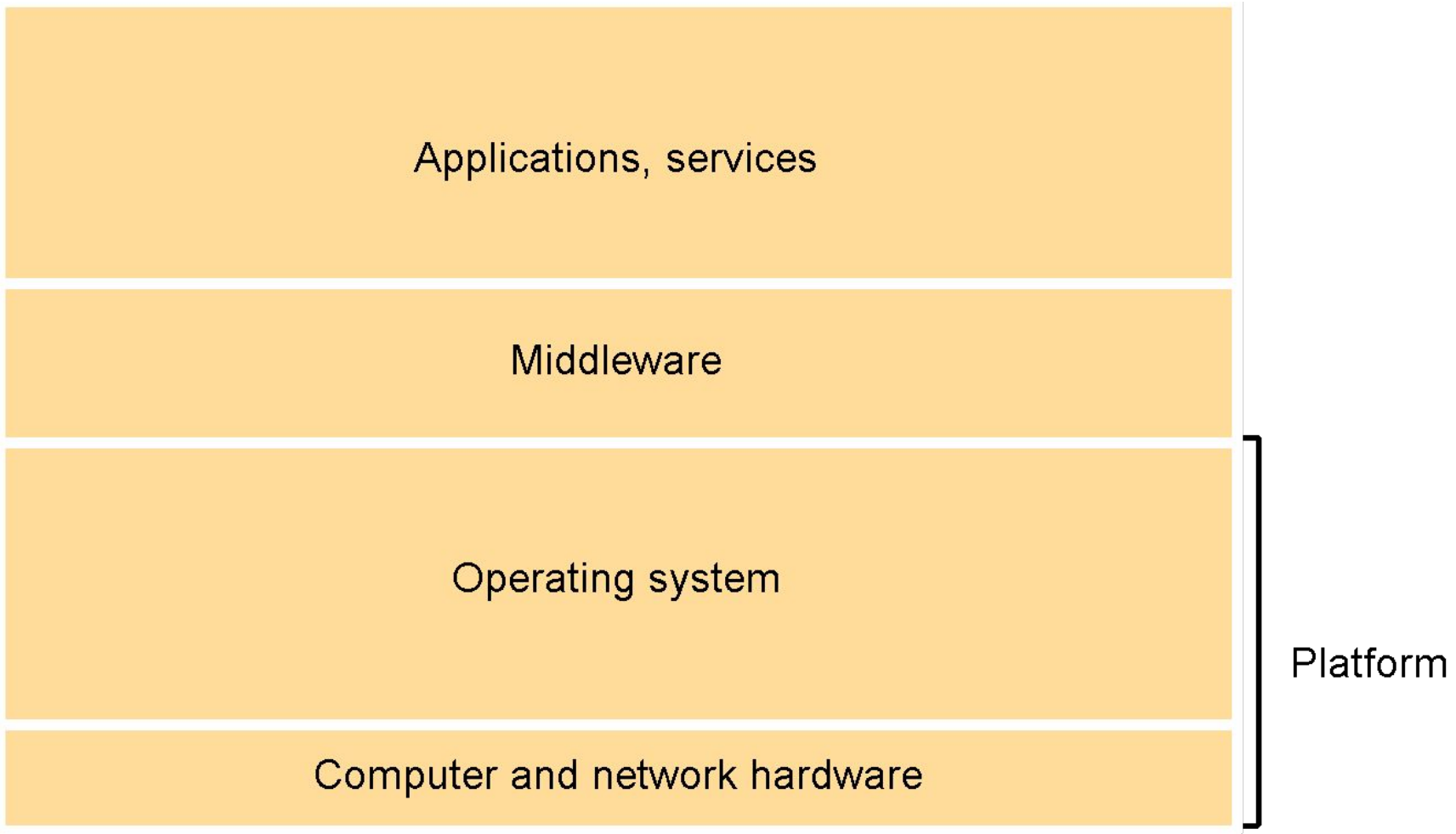
Three types of Model

- Physical – h/w composition of System and interconnection
- Architectural – Main component and their role
  - s/w architecture – how components interact
  - System architecture – How components are deployed in underlying network
- Fundamental – concerned with properties that are common in all of the architectural model
  - Interaction model-deals with the difficulty in setting time limits
  - Failure model-defines the ways in which failure may occur
  - Security model-discuss possible threats and their solutions

Architectural Model

- Deals with placement of components and interaction between them
- For this purpose, the process is classified as – Server, Client, and peer. And work is allotted to them.
- Software Layers:
  - Each layer take services from lower layer and offer services to upper layer
  - Middle layer – Services: Communication, data sharing, naming, security, transaction, storage, manage heterogeneity of DS, provide common programming abstraction
  - Platform – Provide programming interface for communication and coordination

Software and hardware service layers in distributed systems



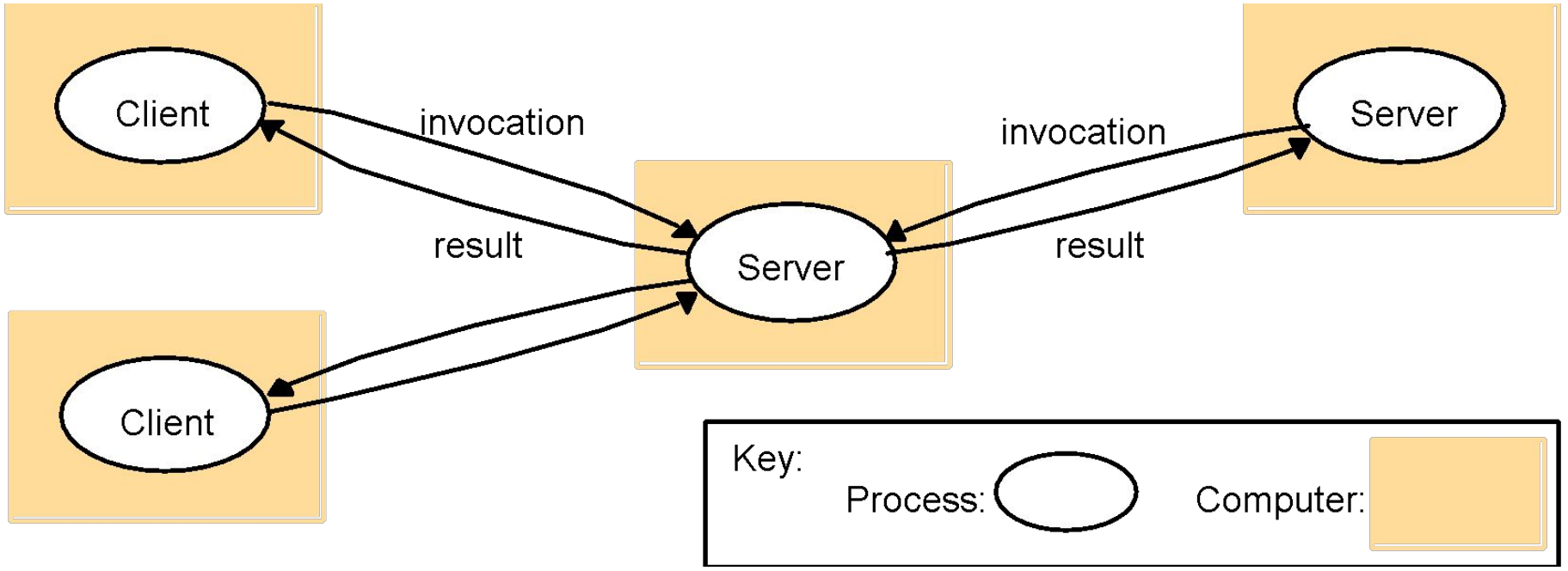
System Architecture

- Deals with division of responsibilities of system components.
- Placement of components in the network.
- Client-Server Model –
  - Client- aprocess that request a service from server by sending a request
  - Server- a process that implements a specific service and reply to the client
- A server may be client of other server
  - Web server- a client of file server that manages the file in which the web pages are stored
  - Web server- a client of DNS server which translates domain names in to n/w address.

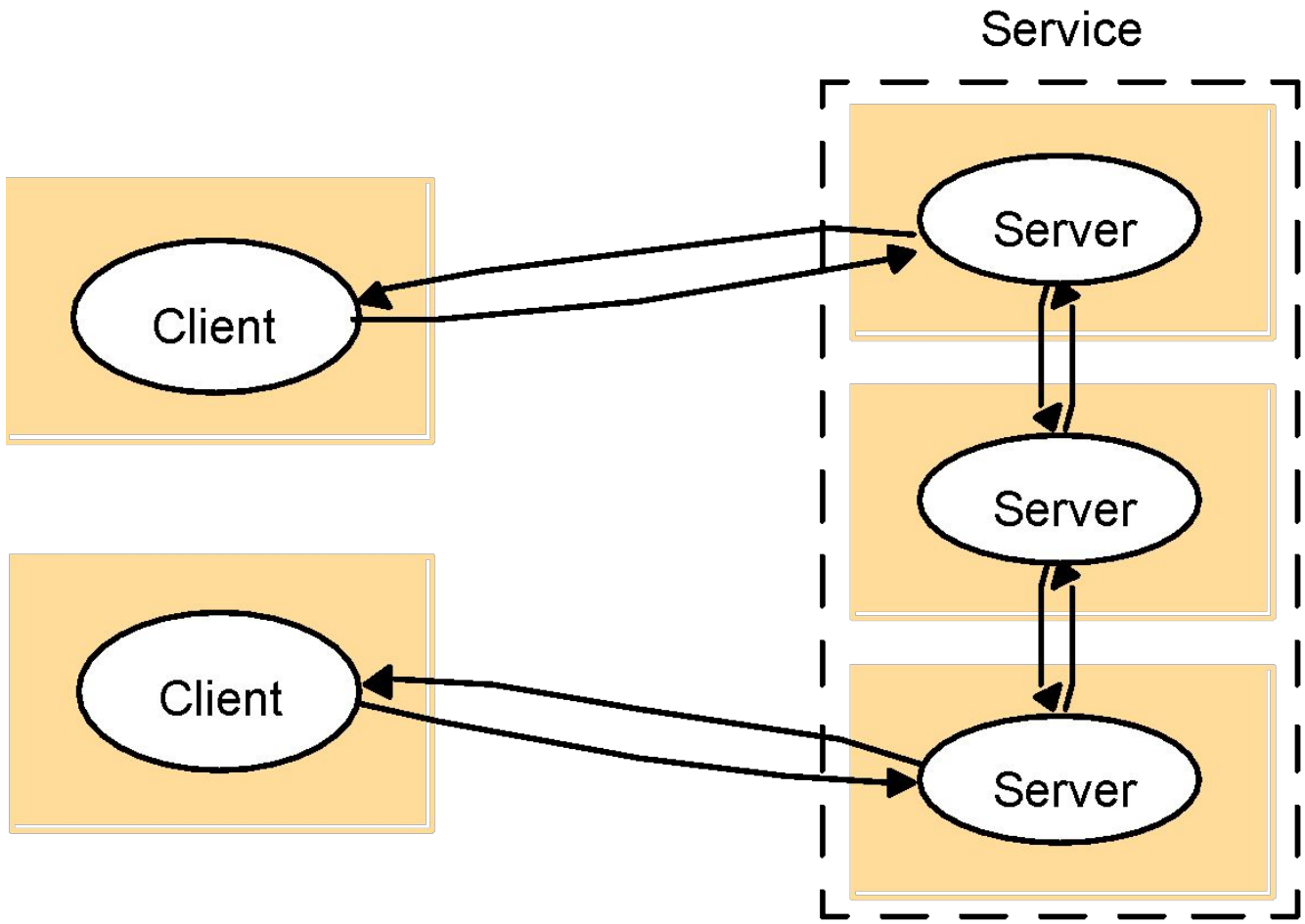
System Architecture

- A server may be partitioned and services are distributed
- A server may be replicated to increase performance and tolerate failing
- Cache- buffer of recently used data obj and supplies the data obj to client when required
- Proxy server-
  - It increases availability & performance of a service by reducing load on n/w and web servers
  - Provide shared cache of web server to a site
  - Used to access remote web server
- Peer process- applications interact with each other

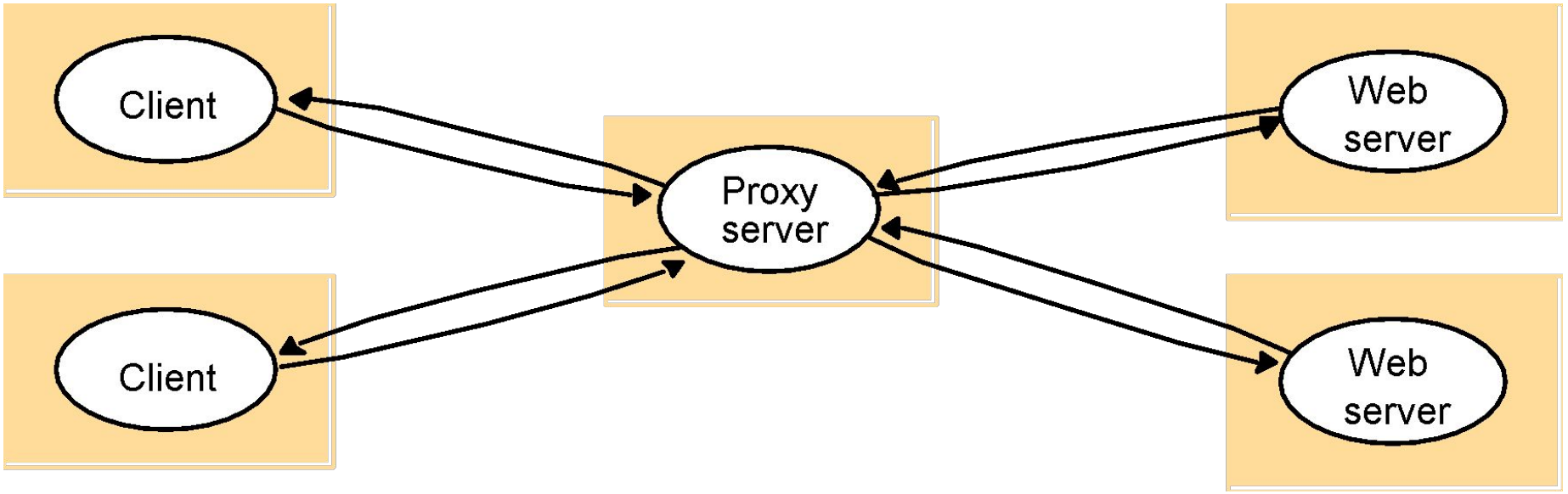
Clients invoke individual servers



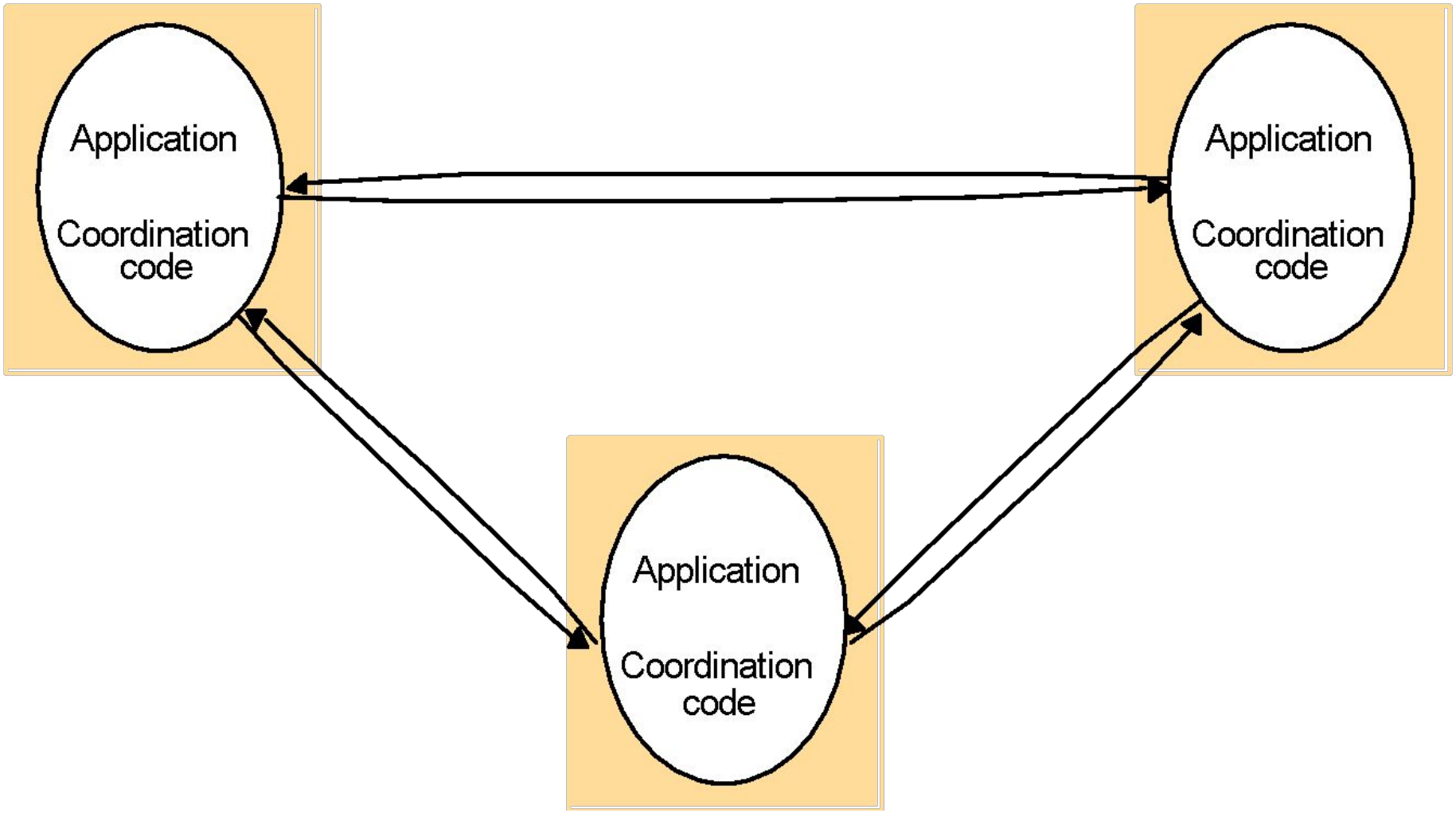
A service provided by multiple servers



Web proxy server

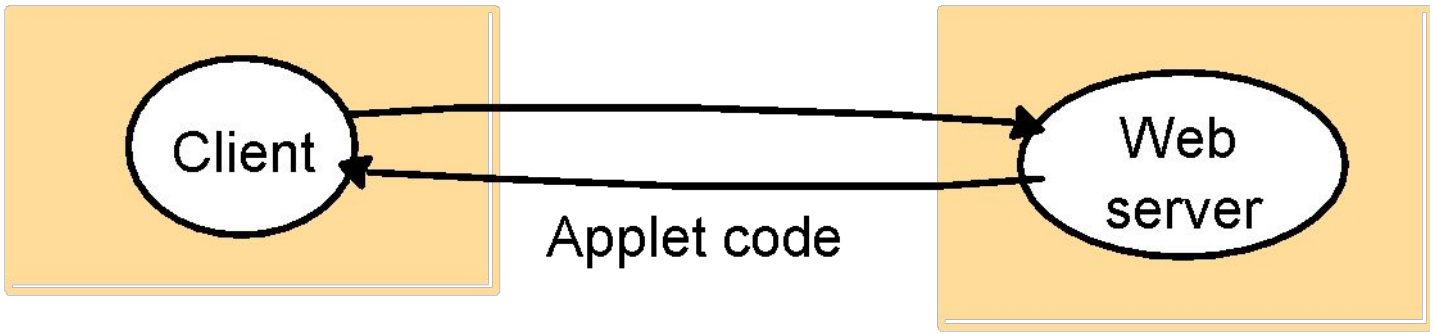


A distributed application based on peer processes



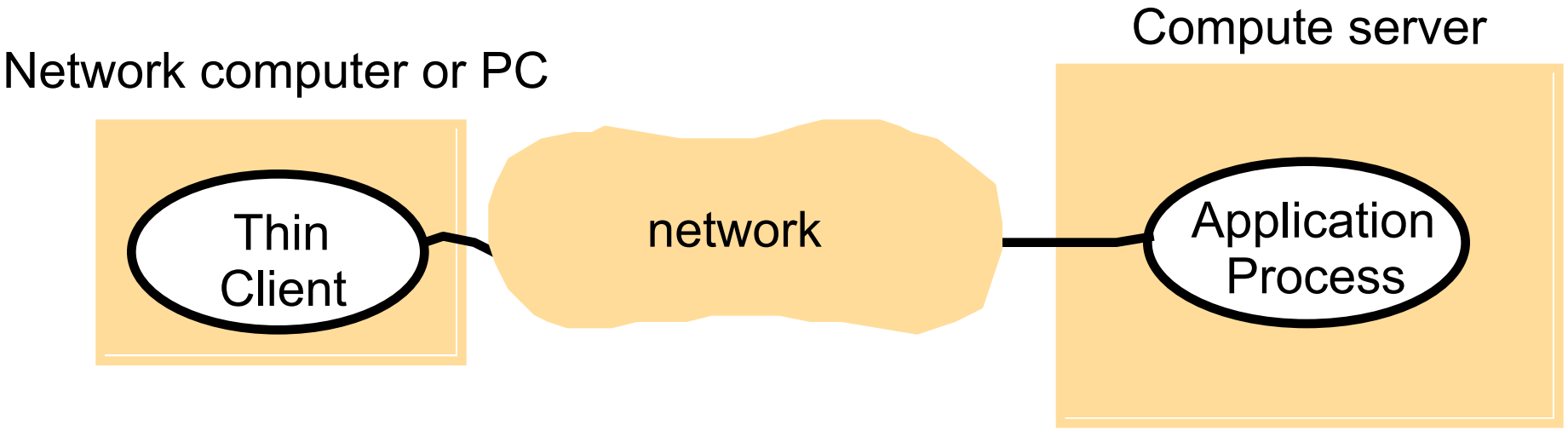
Web applets

a) client request results in the downloading of applet code

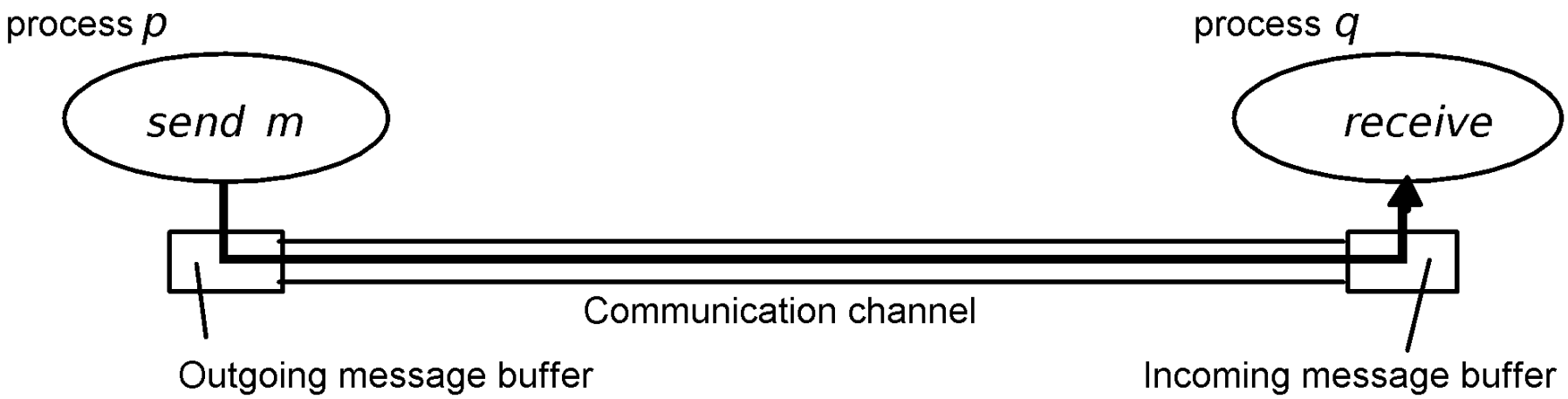


b) client interacts with the applet

Thin clients and compute servers



Processes and channels



Fundamental Model

- All communication is done by msg – Interaction model
- There may be failures – Failure model
- Vulnerable to security attacks – security model

Interaction model

- Processes in DS communicate through msg passing
- Processes are affected by:
  - Delay- total time from sender to receiver (propogation time, transmission time, time take by OS for processing)
  - Bandwidth- amount of info that can be transmitted over given amount of time.
  - Jitter- variation in time taken to reach the destination

Failure model

- Ommision failure – process or channel failure
  - Process failure- if a process fails, it will not respond to request
  - Communication failure-
    - Dropping msg due to lack of buffer space
    - Sending omission failure- loss of msg between sending process and out going buffer at server
    - Receiver omission failure- loss of msg between incoming buffer and receiver process
    - Channel omission failure- loss o msg in between sender and receiver
- Arbitrary failure- any type of failure
- Timing failure- applicable in synchronous DS where time limits are set for all operations.

Security model

- Security can be achieved by securing processes , channels and objects
  - Protecting obj- by giving access rights (server is responsible for verifying)
  - Securing process-
    - Process interact through msg- secured by confidentiality, authentication and integrity
    - A process may receive a msg from any other process
    - An enemy can copy and alter the msg as they travel on media
- Denial of service- huge traffic is generated to overload the resources- so delay is increased

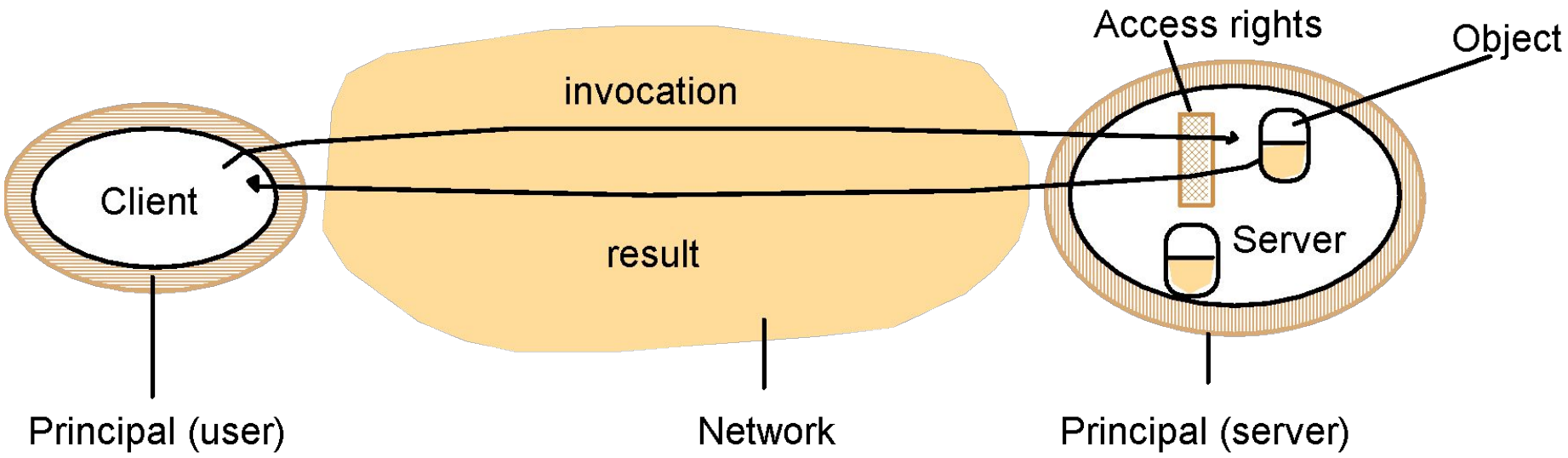
Omission and arbitrary failures

Class of failure	*Affects*	*Descriptio*
Fail-stop	Proces s	Process halts and remains halted. Other processes may detect this state.
Cras h	Proces s	Process halts and remains halted. Other processes may not be able to detect this state.
Omissio n	Channe l	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omissio n	Proces s	A process completes a *send* but the message is not put in its outgoing message buffer.
Receive-omission	Proces s	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrar y (Byzantine)	Process or channe l	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.

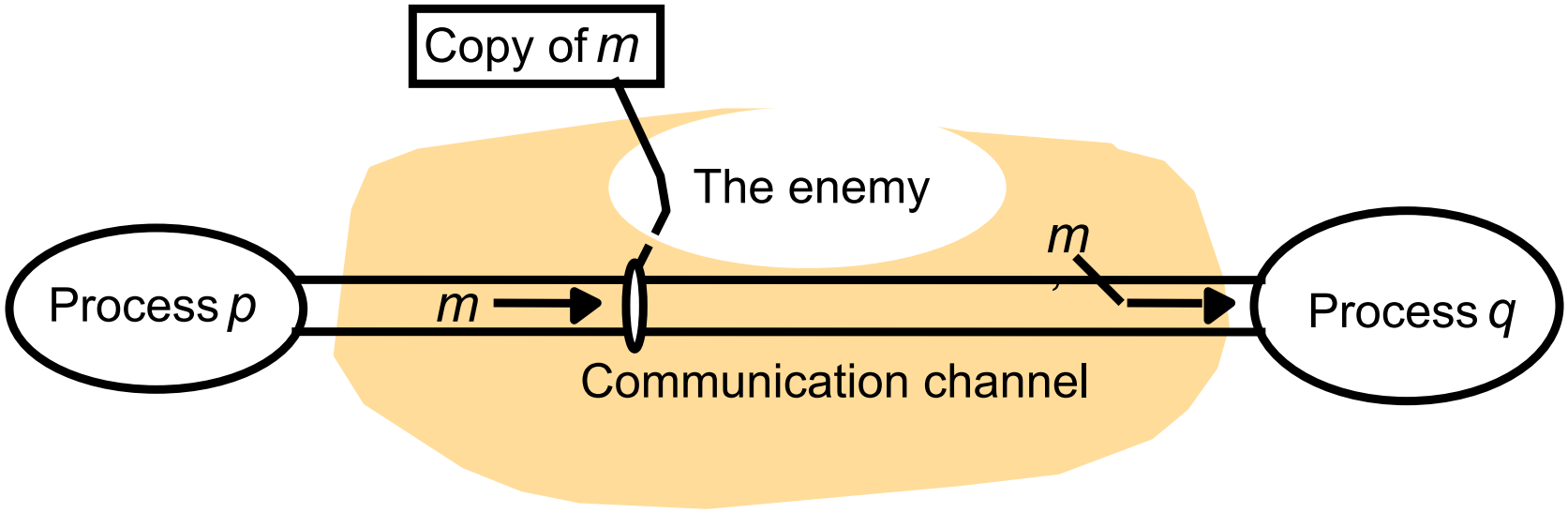
Timing failures

Class of Failure	*Affects*	*Descriptio*
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.

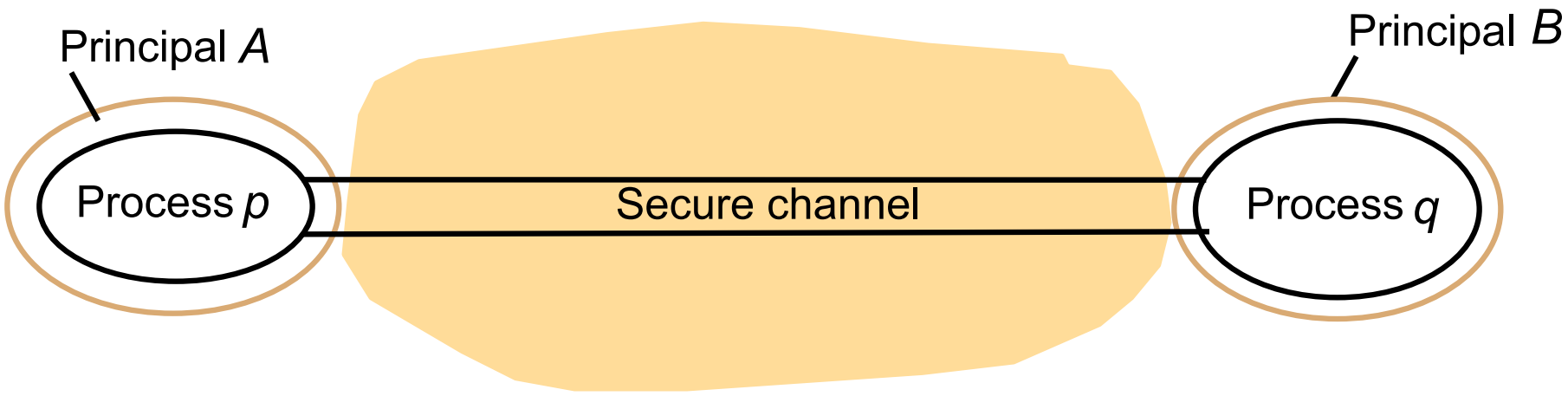
Objects and principals



The enemy


Copy of m
The enemy
m'
Process p
m
Process q
Communication channel

Secure channels



Ordering of Events and Logical Clocks