DISTRIBUTED SYSTEMS

M.Sc(IT)- 1st SEMESTER

GULLAGONG

P.G. DEPT. OF COMPUTER SC & IT

Distributed Systems

Characterization of Distributed Systems

OUTLINE

- ☆ Distributed System Definitions.
- ☆ Distributed Systems Examples:
  - The Internet.
  - Intranets.
  - Mobile and Ubiquitous Computing.
- ☆ Resource Sharing.
- ☆ The World Wide Web.
- ☆ Distributed Systems Challenges.

Why Distributed Systems?

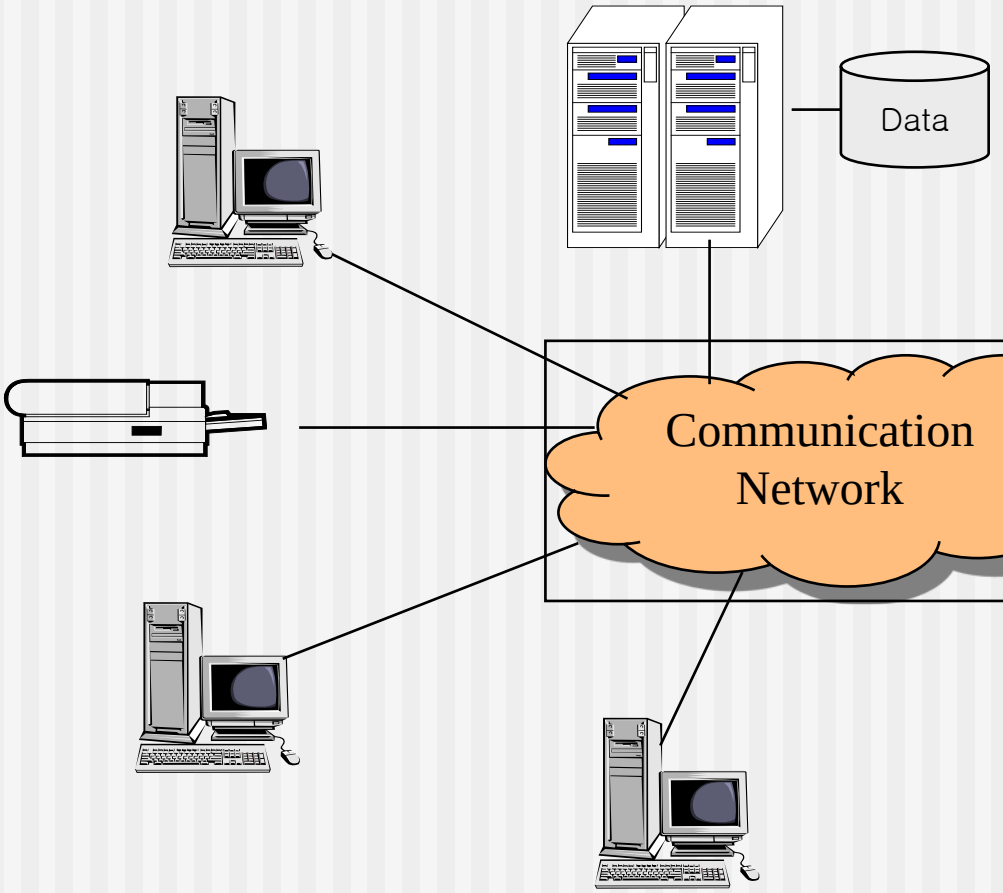
- Main features
  - Geographical distribution of autonomous computers
  - Communication through cable/fiber/wireless/...connections

- Distributed system ?
  - A collection of independent computers that appears to its users as a single coherent system logically(called a single view system).
  - Advantages
    - interaction, co-operation, and sharing of resources
  - Benefits
    - reduced costs, improved availability and performance
    - Scalability, resource sharing, fault tolerance.

Distributed System Definitions

- A system in which hardware or software components located at *networked computers* communicate and coordinate their actions only by *message passing*.
  - Concurrency of components.
  - No global clock.
  - Independent failures.
- A collection of two or more *independent* computers which coordinate their processing through the exchange of *synchronous or asynchronous* message passing.
- A collection of *independent* computers that *appear to the users* of the system as a single computer.

Distributed Systems



0. Distributed system ?
- Autonomous systems[자치 시스템]
- Computer network(유, 무선)
- Distributed software

0. Important consequences
- Concurrency[동시성]
- No global clock (독립적 시간)
- Independent failure

Computer Networks vs. Distributed Systems

- *Computer Network*: the independent computers are explicitly visible (can be explicitly addressed).
- *Distributed System*: existence of multiple independent computers is transparent.
- However,
  - many problems in common,
  - in some sense networks or parts of them (e.g., name services) are also distributed systems, and
  - normally, every distributed system relies on services provided by a computer network.

Why Distributed Systems

- *Functional* distribution: computers have different functional capabilities (i.e., *sharing of resources* with specific functionalities).
- *Load* distribution / balancing: assign tasks to processors such that the overall system performance is optimized.
- Replication of *processing power*: independent processors working on the same task:
  - Distributed systems consisting of collections of microcomputers may have processing powers that no supercomputer will ever achieve.
- *Physical* separation: systems that rely on the fact that computers are physically separated (e.g., to satisfy reliability requirements).
- *Economics*: collections of microprocessors offer a better price/performance ration than large mainframes.
  - mainframes: 10 times faster, 1000 times as expensive

Typical examples

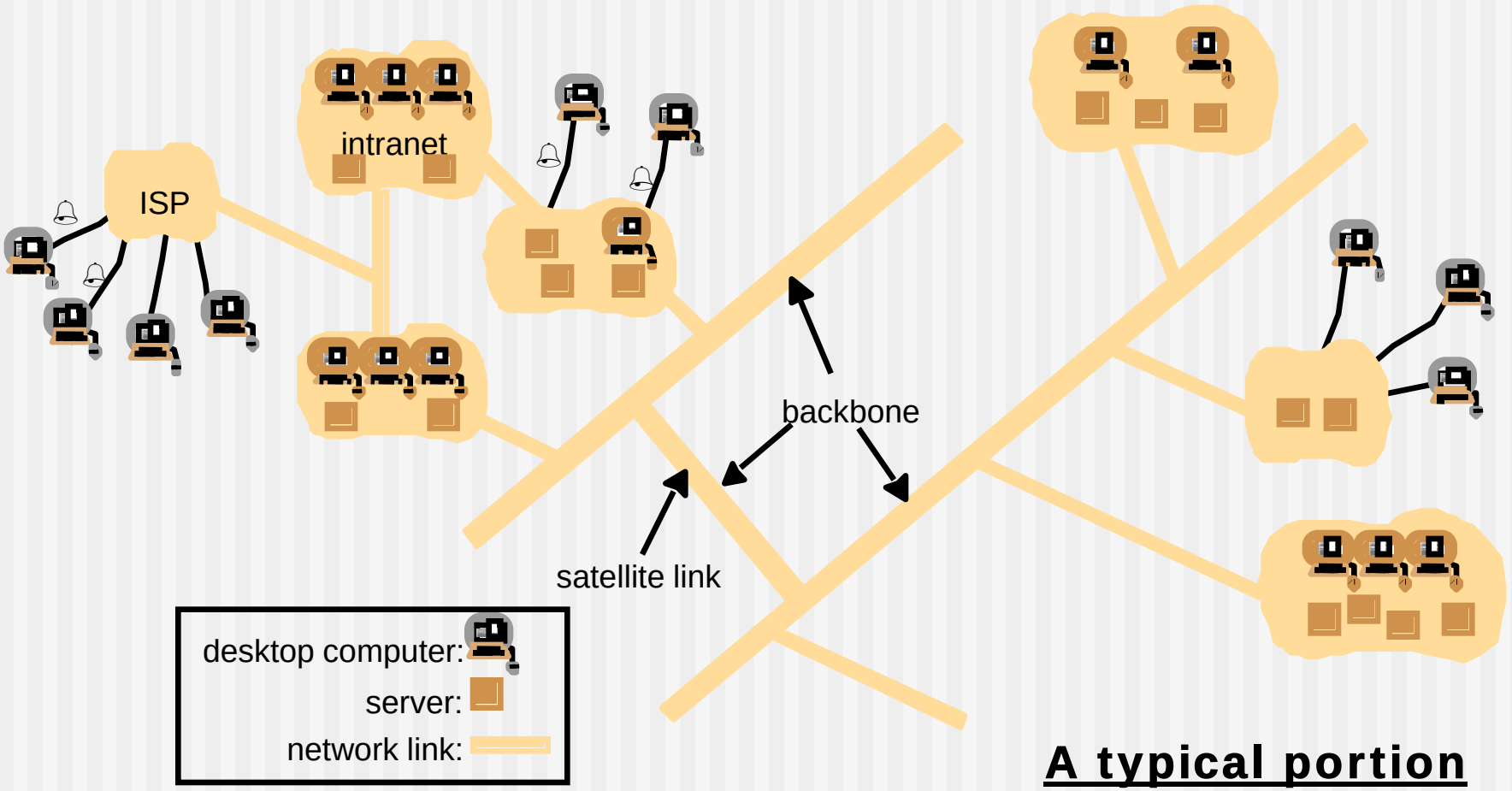
- **Internet**
  - global network of interconnected computers which communicate through IP protocols
- **Intranet**
  - a separately administered network with a boundary that allows to enforce local security policies
- **Mobile and ubiquitous computing**
  - laptops, PDAs, mobile phones, printers, home devices, ...
- **World-Wide Web**
  - system for publishing and accessing resources and services across the Internet

Distributed Systems Examples
(The Internet)

- The Internet is a vast interconnected collection of computer networks of many types.
- Its design enabling a program running anywhere to address messages to programs anywhere else.
- Allowing its users to make use of many services as: WWW, E-Mail, Web hosting, and File transfer.
- Its services can be extended by adding new types of service (*open-ended services*).
- Small organizations and individual users can to access internet services through *Internet Service Providers* (ISPs).
- Independent intranets are linked together by high transmission capacity circuits called *backbones*.

Distributed Systems Examples

(The Internet)



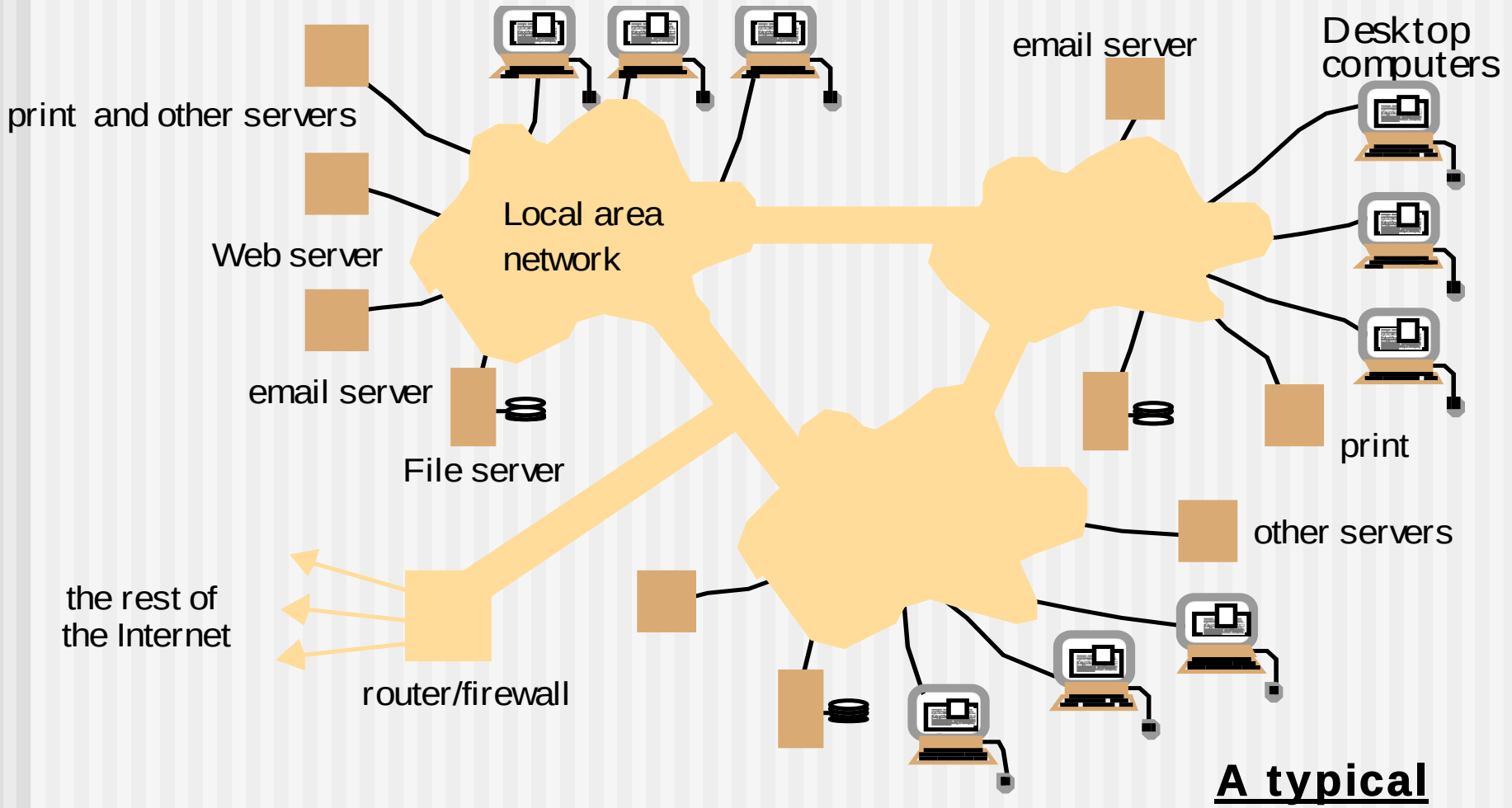
A typical portion of the Internet

Distributed Systems Examples
(Intranets)

- An Intranet is a portion of the internet that is administrated separately and its local security policies are enforced by a configured boundary.
- Composed of several local area networks (LANs) linked by backbone connections to allow its users to access the provided services.
- Connected to the Internet via a *router* which allows its users to make use of the internet services elsewhere.
- Many organization protect their own services from unauthorized use by filtering incoming and outgoing messages using a *firewall*.

Distributed Systems Examples

(Intranets)



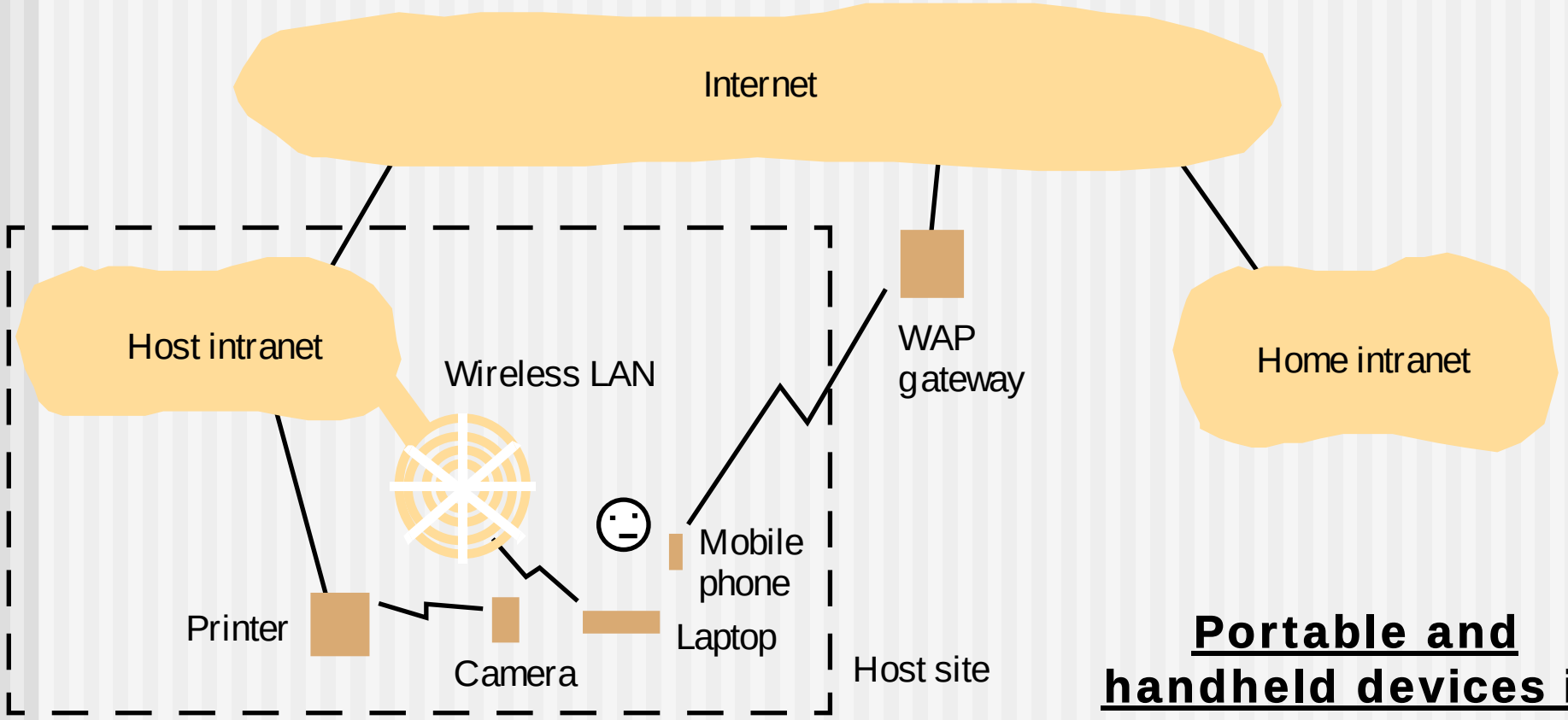
A typical Intranet

Distributed Systems Examples
(Mobile and Ubiquitous Computing)

- The portability of many computing devices and the ability to connect to networks in different places makes mobile computing possible.
- *Mobile computing* is the performance of computing tasks while the users are on the move and away from their residence intranet but still provided with access to resources via the devices they carry with them.
- *Ubiquitous computing* is the harnessing (يستخدم) of many small cheap computational devices that are present in user's physical environments.
- Ubiquitous and mobile computing overlap but they are generally distinct.

Distributed Systems Examples

(Mobile and Ubiquitous Computing)



Portable and handheld devices in a distributed system

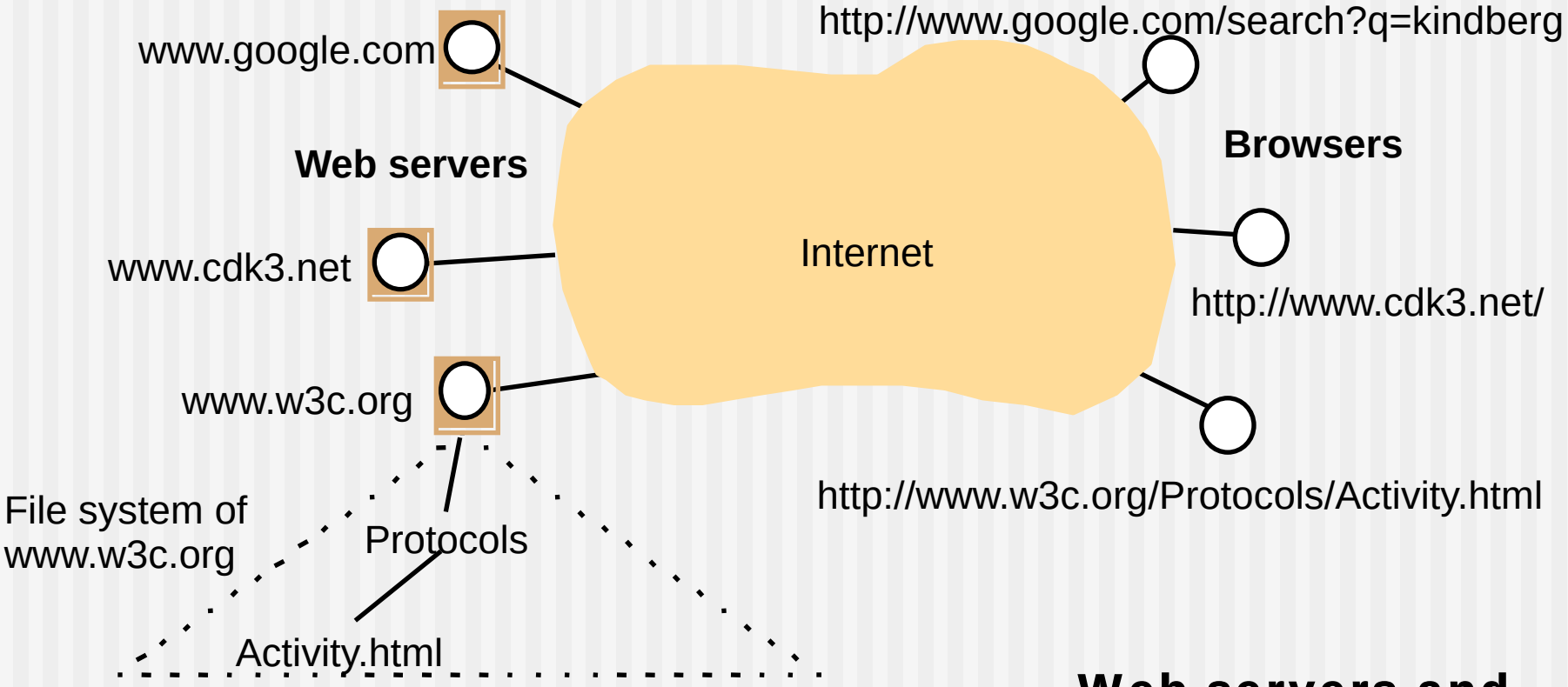
Resource Sharing

- Resources in a distributed system are:
  - Encapsulated within computers.
  - Can only accessed from other computers by communication.
  - Managed by a program that offers a communication interface.
- The term *service* is used for a distinct part of a computer system which manages a collection of related resources and presents their functionality to users and applications via a well-defined set of operations.
- *Server* refers to a running program on a networked computer that accepts request messages from programs, *clients*, running on other computers to perform a service and responds appropriately by reply messages.
- A complete interaction between a client and a server is called a *remote invocation*.

The World Wide Web (WWW)

- An executing web browser is an example of a client communicates with a web server to request web pages from it.
- WWW is an open client-server architecture implemented on top of the Internet.
- Users use the Web through available web browsers to retrieve and view documents of many types and interact with unlimited set of services.
- Web provides a *hypertext* structure among the documents that it stores. (i.e., the documents contain references, *links*, to other related documents and resources stored also in the web).

The World Wide Web (WWW)



Web servers and web browsers

The World Wide Web (WWW)

- Web is an open system and can be extended and implemented in new ways without disturbing its existing functionality.
  - Its operation is based on communication standards and document standards that are freely published and widely implemented.
  - Types of resources that can be published and shared on it is open (plug-ins handle new document formats without changing the web browser).
- Web is based on three main standard technological components:
  - The HyperText Markup Language (*HTML*) specifies the contents and layout of pages as they are displayed by web browser.
  - Uniform Resource Locators (*URL*s) identify documents and other resources accessible via the web.
  - The HyperText Transfer Protocol (*HTTP*) determines the standard rules for interaction between browsers and web servers to fetch documents and other resources.

World Wide Web Components (HTML)

- HTML is used to specify:
  - The text and images that make up the contents of a web page,
  - How web page components are formatted for presentation to the user, and
  - Links and which resources are associated with them.
- HTML is produced using a standard text editor or an HTML tool.
- HTML uses tags to specify content:

<IMG SRC = "http://www.cdk3.com/WebExample/Images/earth.jpg">
<P> Welcome to Earth! Visitor may also be interested in taking a look at the
<A HREF = "http://www.cdk3.com/WebExample/Moon.html"> Moon </A> </P>
.....

- HTML text is stored in a file accessible by a web server.
- A browser retrieves the file from the web server and interprets its HTML text to display the web page in the familiar fashion.

World Wide Web Components (URLs)

- Browsers examine URLs in order to fetch the corresponding resources from web servers.
- A URL is typed by the user into the browser or selected from their bookmarks.
- Also, the browser looks up the corresponding URL when the user clicks on a link or fetches a resource embedded in a web page such an image.
- Every URL in its full form has tow top-level components:

  scheme : scheme-specific-location

- URL examples:
  - *http://www.cdk3.net/*
  - *http://www.google.com/search?q=kindberg.*
  - *ftp://ftp.downloadIt.com/software/aProg.exe*
  - *mailto:metwally@ccis.ksu.edu.sa*

World Wide Web Components (HTTP)

- HTTP defines the ways in which browsers and any other types of client interact with web server.
- HTTP is a request-reply protocol:
  - The client sends a request message to the server containing the URL of the required resource,
  - Then the server looks up the pathname and if it exists, sends back the file's contents in a reply message to the client.
- Browser are not necessarily capable of handling every type of the file's contents:
  - A browser includes a list of preferred types when makes a request.
  - The server take this into account when returns the file content and includes the content type in the reply message so that the browser know how to process it.
- HTTP allows the client to request one resource at a time.

Challenges posed by Distributed Systems

- Due to:
  - Complexity
  - Size
  - changing technologies
  - society's dependence
- Challenges posed by DSs
  - Heterogeneity
  - Openness
  - Security
  - Scalability
  - Fault handling
  - Concurrency
  - Transparency

Distributed Systems Challenges

- A number of challenges are recognized in the design of distributed systems.
- These challenges have been tackled with varying degrees of success in existing systems.
- The list below gives some indication of measures that are employed to meet each challenge:
  - *Heterogeneity :* standards and protocols; middleware; virtual machine;
  - *Openness*: publication of services; notification of interfaces;
  - *Security*: firewalls; encryption;
  - *Scalability*: replication; caching; multiple servers;
  - *Failure Handling*: failure tolerance; recover/roll-back; redundancy;
  - *Concurrency*: concurrency control to ensure data consistency;
  - *Transparency*: middleware; location transparent naming; anonymity

Distributed Systems Challenges (Heterogeneity)

- The internet enables users to access services over a variety and difference:
  - Networks.
  - Operating systems.
  - Implementations by different developers.
  - Computer hardware.
  - Programming languages.
- The different networks are masked by the fact that all of the attached computers are communicate to each other using standard internet protocols.
- The *middleware* software layers (like CORBA) provide a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages.
- *Virtual machines* approach provides a way of making code executable on any hardware – *mobile code*.

Distributed Systems Challenges (Openness)

- The openness of distributed systems is determined by the degree to which new resource-sharing services can be added and be available for use by variety of client programs (*services publication*).
- The specification and documentation of key software interfaces of the system components must be available to software developers (*interfaces notification*).
- Systems that are designed to support resource sharing in this way are termed open distributed systems to emphasize the fact that they are extensible.

Distributed Systems Challenges (Security)

- The security of many information resources available and maintained in distributed systems is importance.
- Security for information resources has three 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.
- A *firewall* can be used to form a barrier around an intranet to protect it from outside users but does not deal with ensuring the appropriate use of resources by users within the intranet.
- *Encryption* can be used to provide adequate protection of shared resources and to keep sensitive information secret when is transmitted in messages over the internet.
- Receiving of an executable program (mobile code) as an electronic mail attachment needs to be handled with care – effects of running the program are unpredictable!

Distributed Systems Challenges (Scalability)

- A system is described as scalable if will remain effective when there is a significant increase in the number of resources and the number of users.

Computers in the internet

Date	*Computers*	*Web servers*
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866

- The design of scalable dist. systems presents many challenges:
  - Controlling the cost of physical resources.
  - Controlling the performance loss.
  - Preventing software resources running out.
  - Avoiding performance bottlenecks.

Distributed Systems Challenges (Failure Handling)

- Failures in a distributed system are partial, therefore the handling of it is particularly difficult.
- Some failures can be detected but others are difficult or impossible to detect it.
- Some detected failures can be hidden or made less severe.
- Recovery from failures involves the design of software so that the state of permanent data can be rolled back after a server has crashed.
- Services can be made to tolerate failures by the use of redundant components:
  - At least two different routes between any two routers in the internet.
  - Databases and domain name tables must be replicated in several servers.

Distributed Systems Challenges (Concurrency)

- Several clients in a distributed system can access a shared resource at the same time
- Any object that represents a shared resource in a distributed system (a programmer that implement it) must be responsible for ensuring that it operate correctly in a concurrent environment.
- Operations of objects in a concurrent environment must be synchronized in a way that its data remains consistent.
- The synchronization of an object operations can be achieved by standard techniques such as semaphores.

Distributed Systems Challenges (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 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.
- *Scaling transparency*: allows the system and applications to expand in scale without change to the system structure or the application algorithms.
- Access and location transparency together provide *network transparency*.

System Models

Distributed system models

- Model: “a simplified representation of a system or phenomenon, as in the sciences or economics, with any hypotheses required to describe the system or explain the phenomenon, often mathematically.”

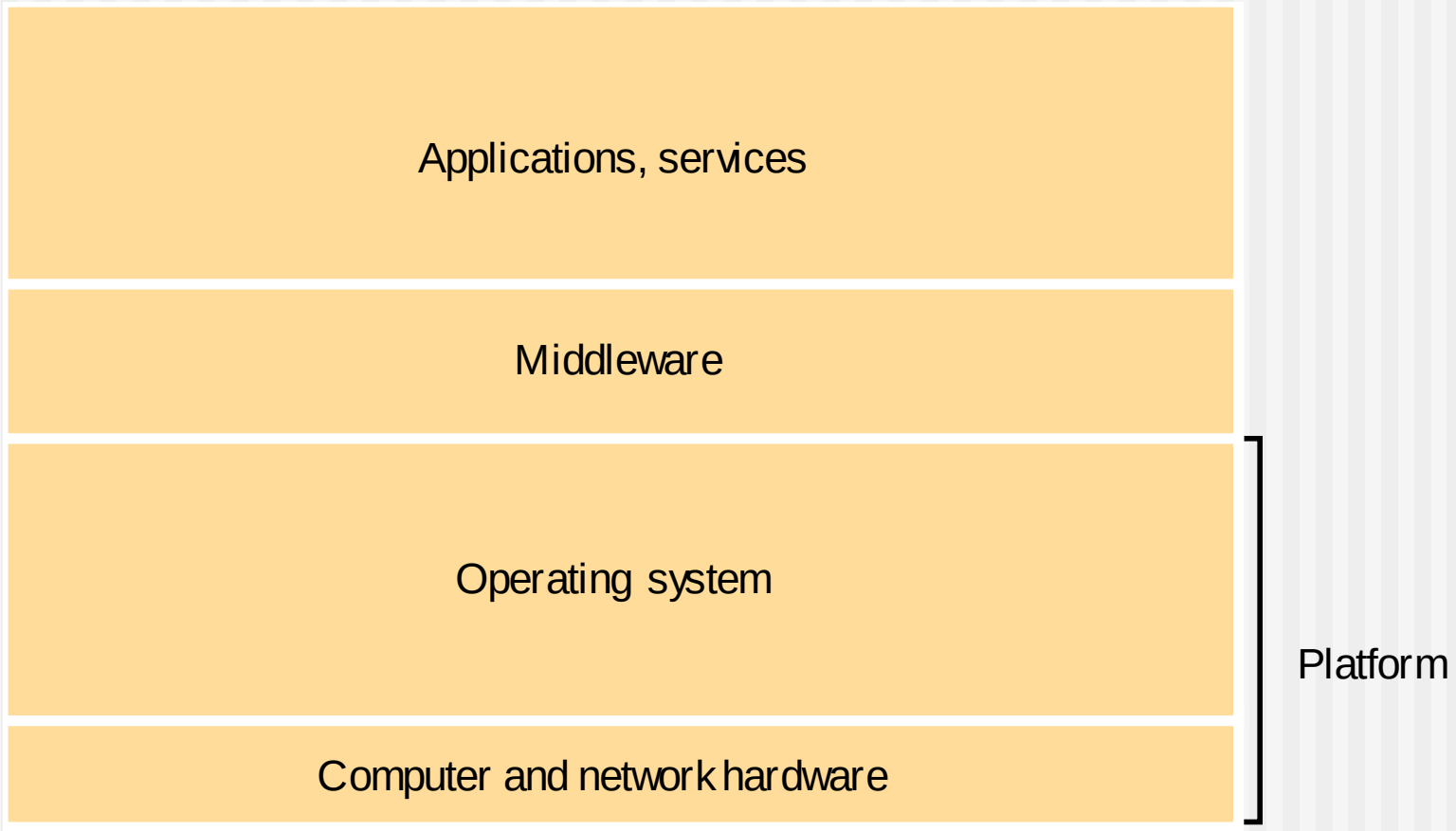
System Models

- **Architectural model** defines the way in which the components of the system are placed and how they interact with one another and the way in which they are mapped onto the underlying network of computers.
- **Fundamental models:**
  - **Interaction model** deals with communication details among the components and their timing and performance details.
  - **Failure model** gives specification of faults and defines reliable communication and correct processes.
  - **Security model** specifies possible threats and defines the concept of secure channels.
- We will discuss the various models at a high level in this discussion and will elaborate on each of these as we discuss other systems.

Architectural Model

- Concerned with placement of its parts and relationship among them.
- Example: client-server model, peer-to-peer model
- Abstracts the functions of the individual components.
- Defines patterns for distribution of data and workload.
- Defines patterns of communication among the components.
- Example: Definition of server process, client process and peer process and protocols for communication among processes; definition client/server model and its variations.

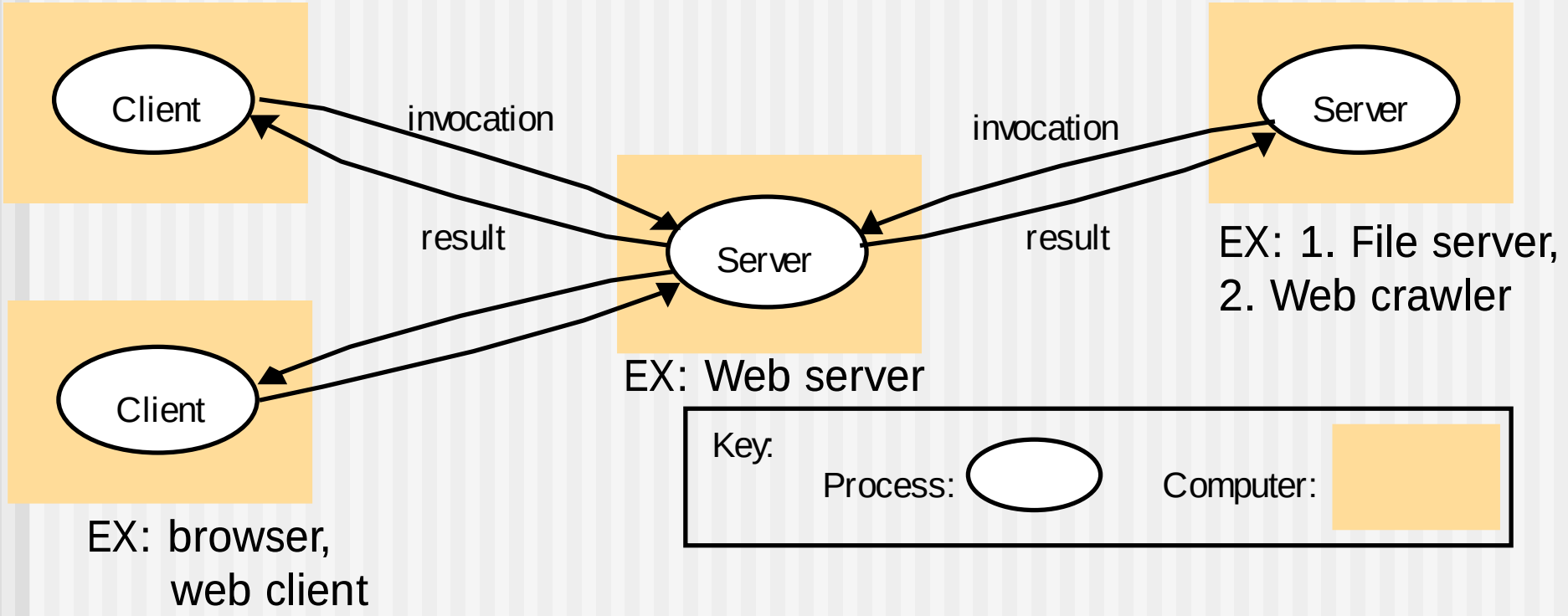
Software and hardware service layers in distributed systems



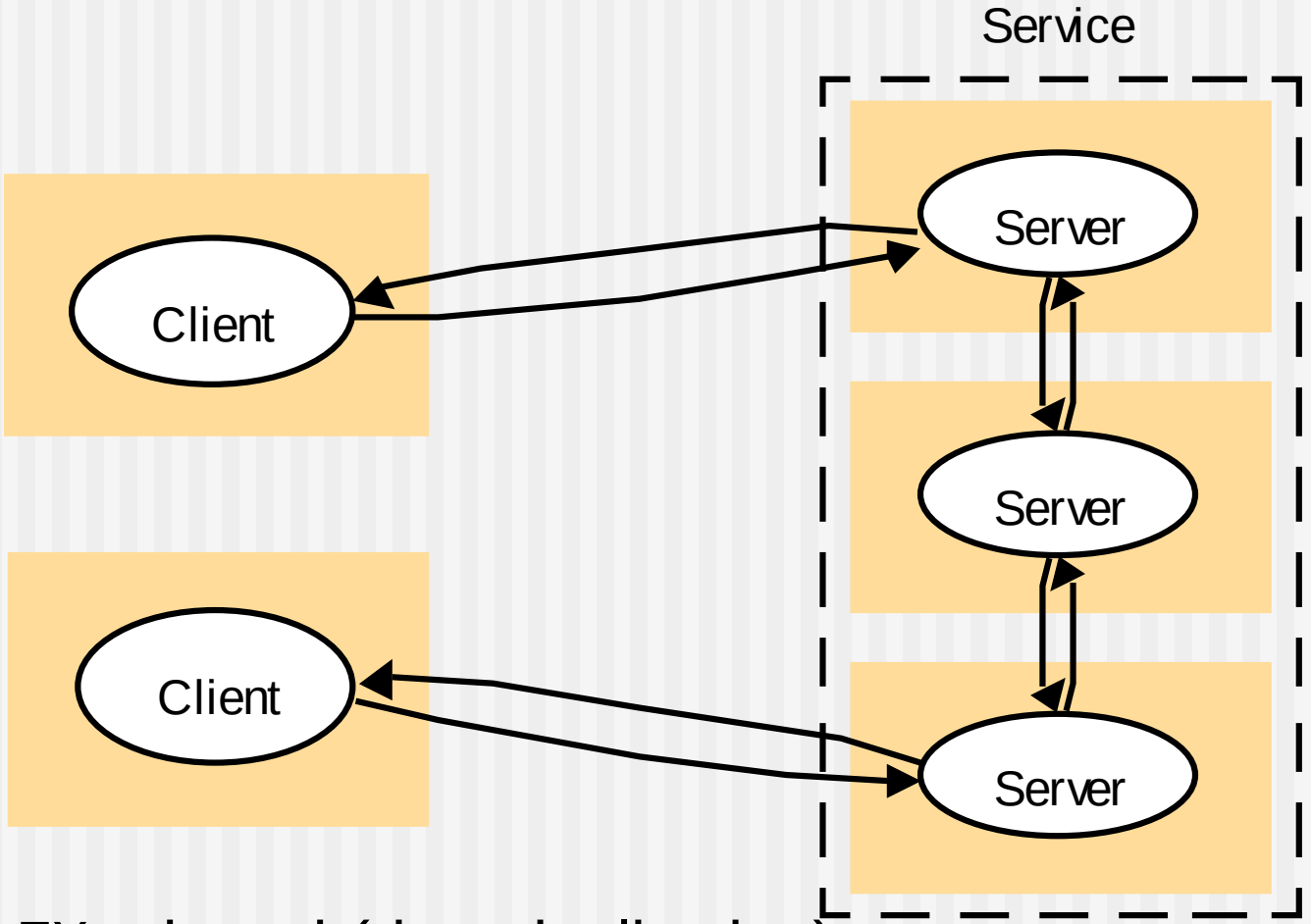
Middleware

- Layer of software whose purpose is to mask the heterogeneity and to provide a convenient programming model for application programmers.
- Middleware supports such abstractions as remote method invocation, group communications, event notification, replication of shared data, real-time data streaming.
- Examples: Java RMI, grid software (Globus, Open grid Services), Web services.

Clients invoke individual servers

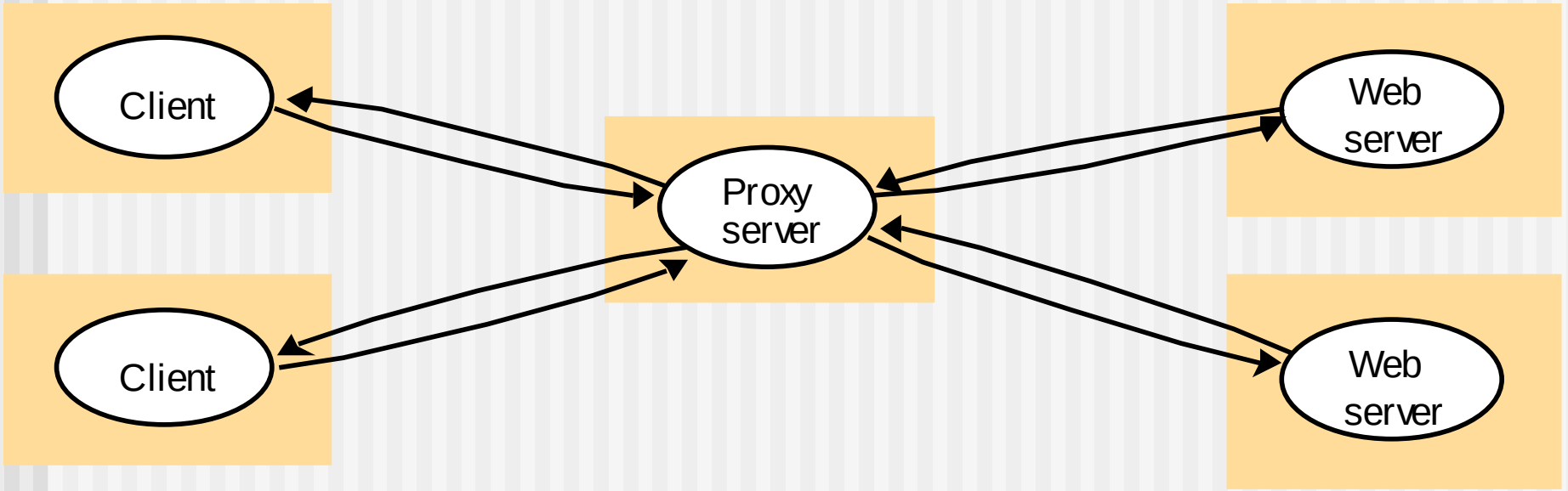


A service provided by multiple servers



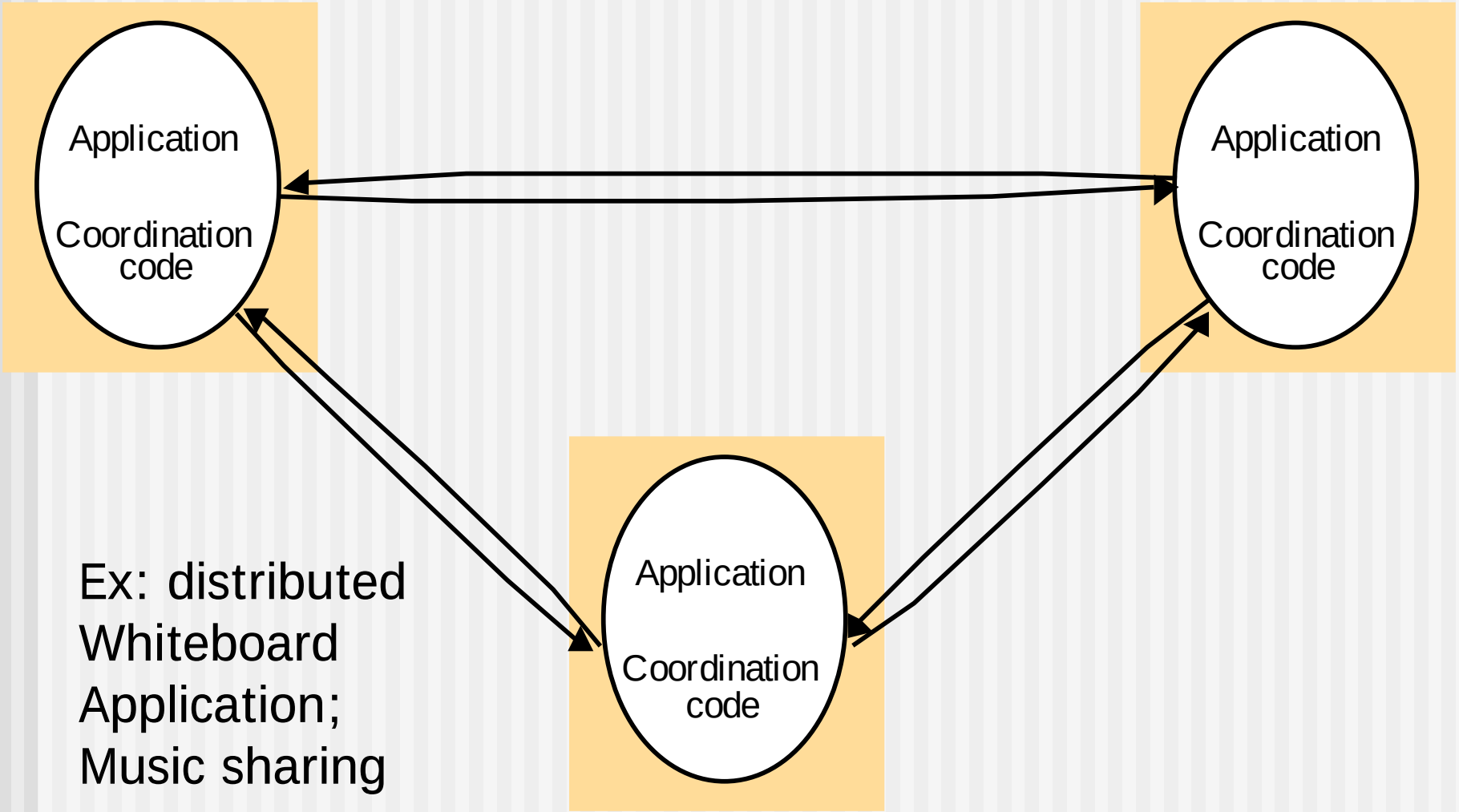
EX: akamai (data duplication), now amazon aws (zones)

Web proxy server and caches



Proxy servers + cache are used to provide increased Availability and performance. They also play a major role Firewall based security. http://www.interhack.net/pubs/fwfaq/

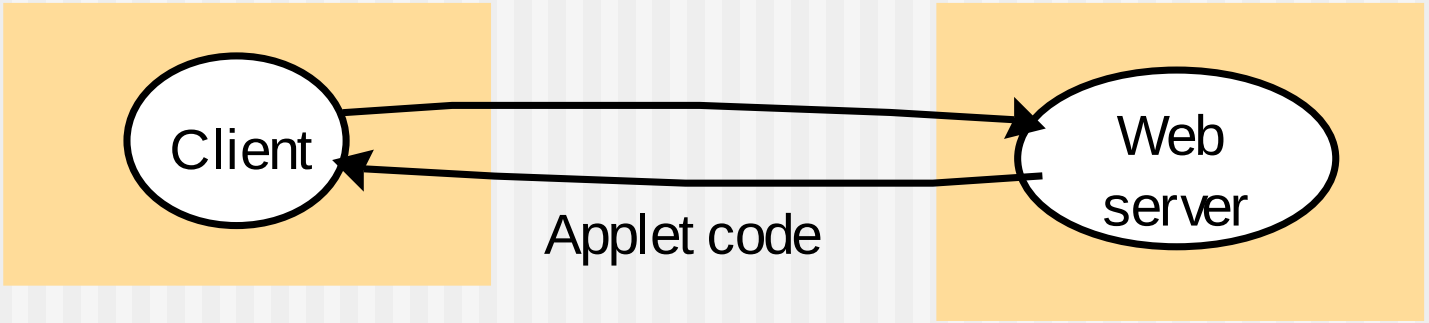
A distributed application based on peer processes



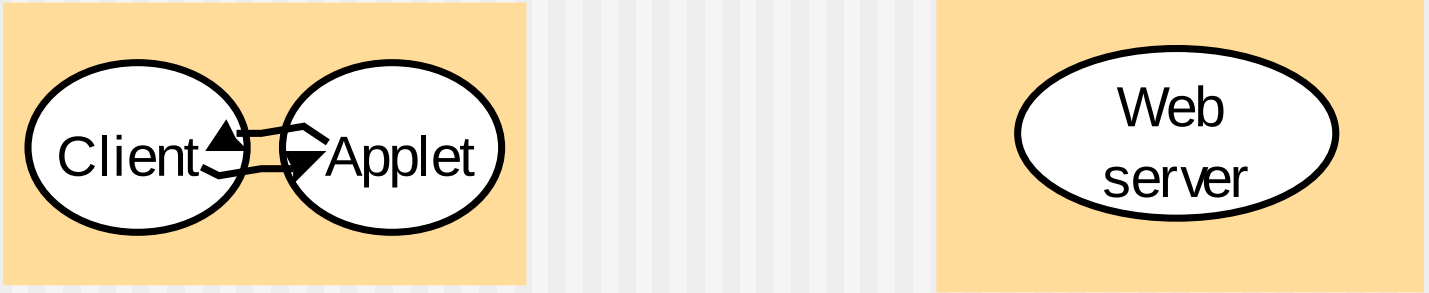
Ex: distributed Whiteboard Application; Music sharing

Web applets

a) client request results in the downloading of applet code



b) client interacts with the applet

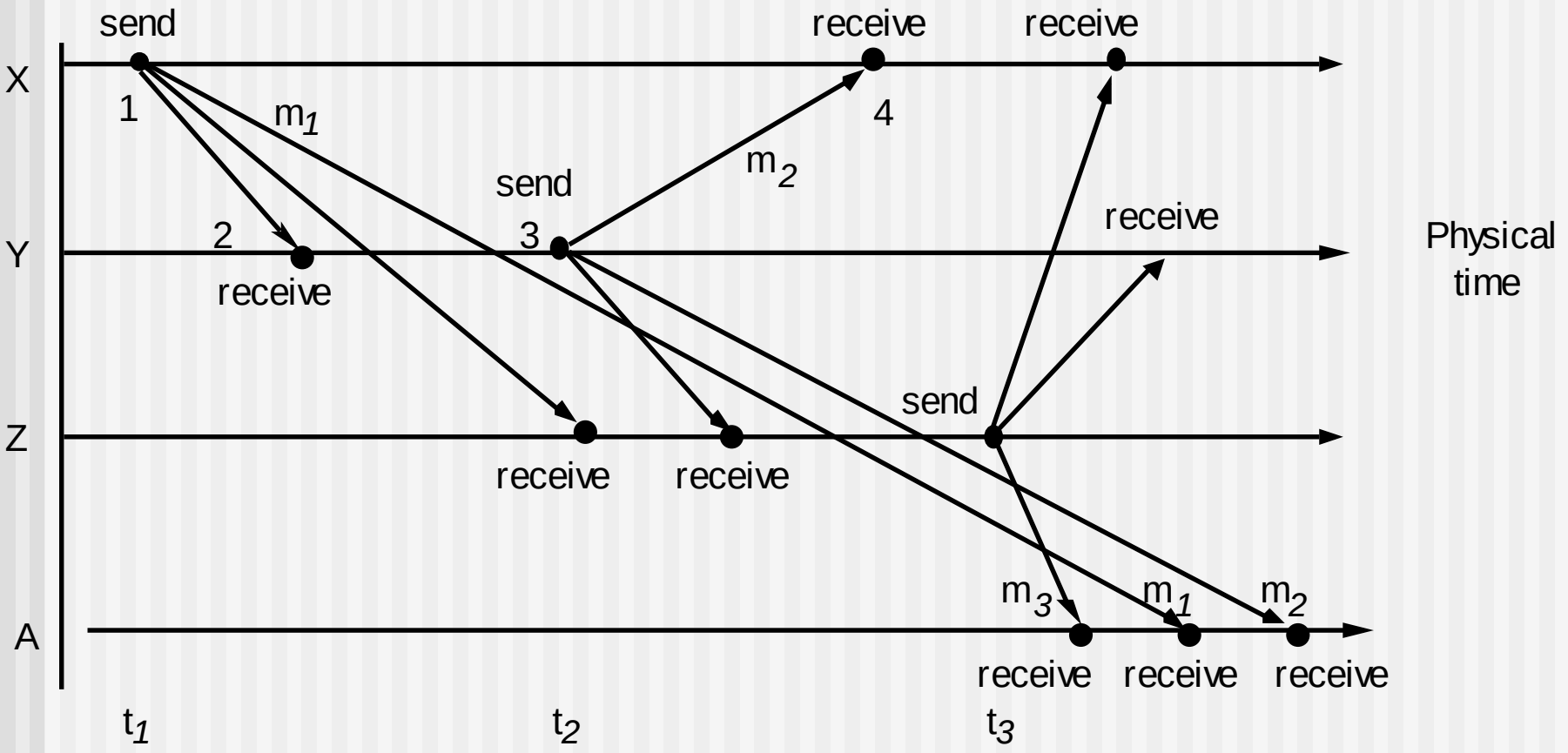


EX: Code streaming; mobile code

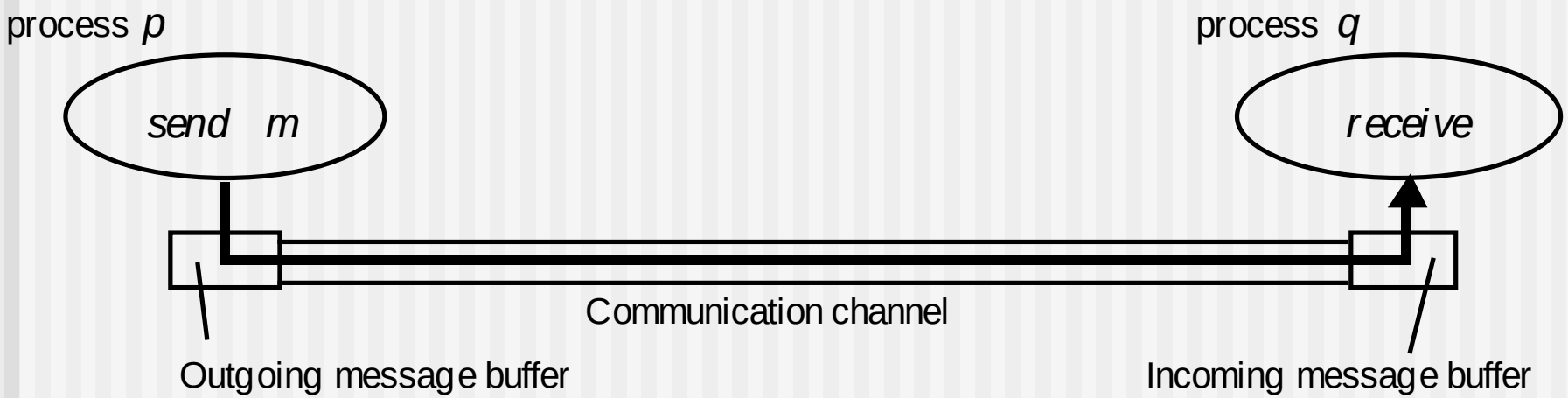
Interaction Models

- Within address space (using path as addresses)
- Socket based communication: connection-oriented, connection-less
  - Socket is an end-point of communication
  - Lets look at some code + details

Real-time ordering of events



Processes and channels



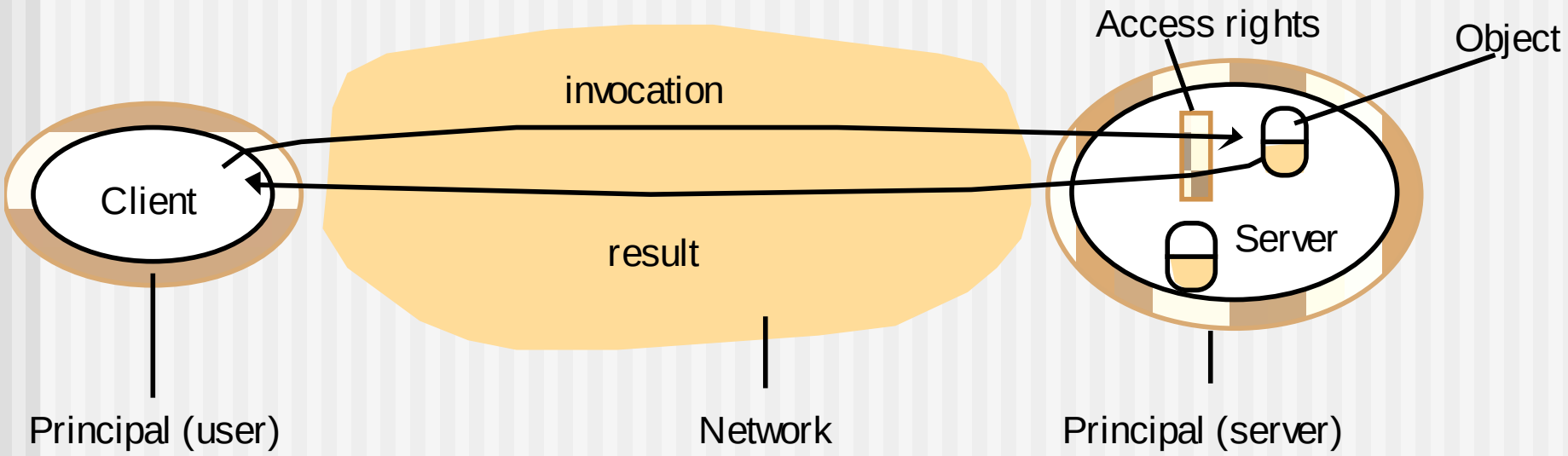
Omission and arbitrary failures

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.

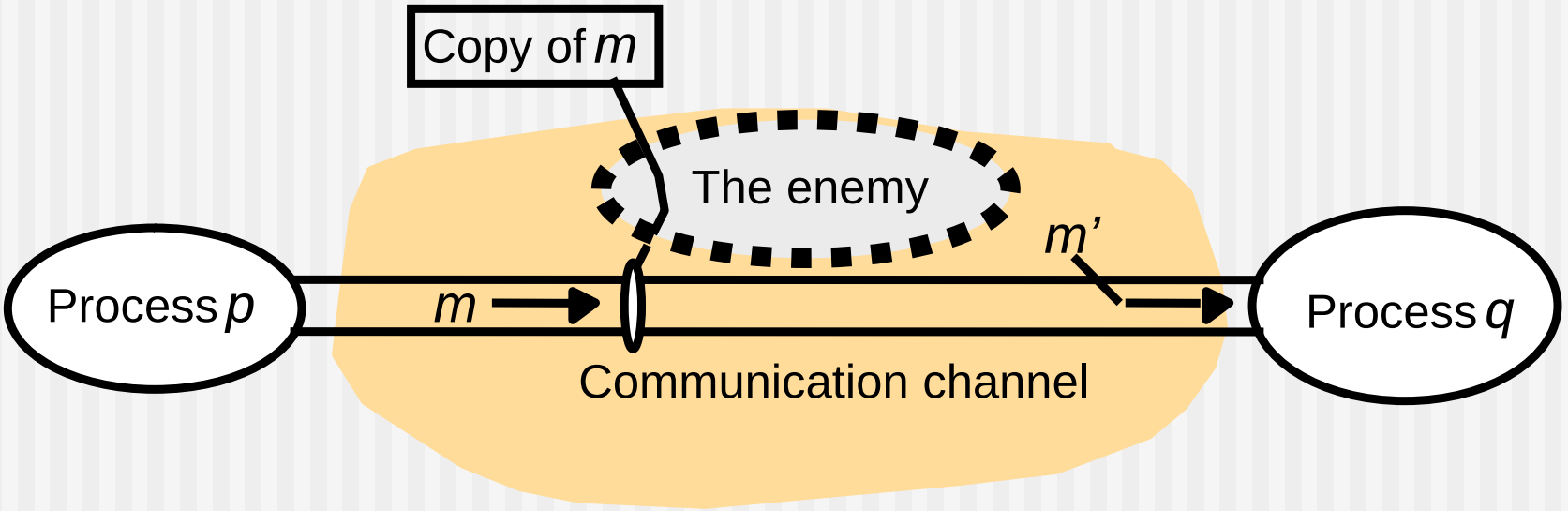
Timing failures

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.

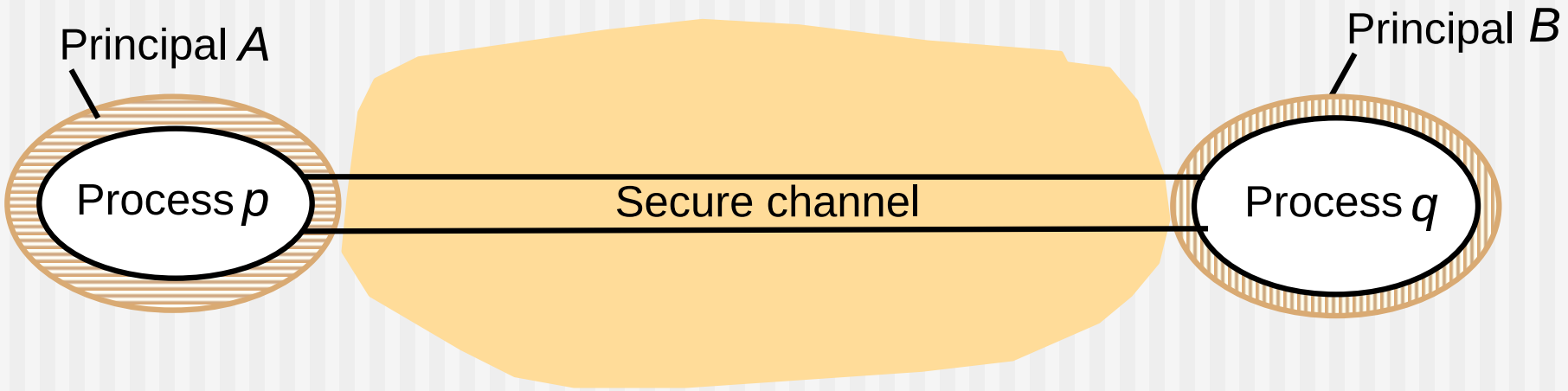
Objects and principals



The enemy


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

Secure channels



Summary of System Models

- When designing systems or analyzing systems, you want to examine at the high level the architectural model.
- Subsequent steps will explore fundamental models such as interaction model, security model, failure model, reliability model etc.