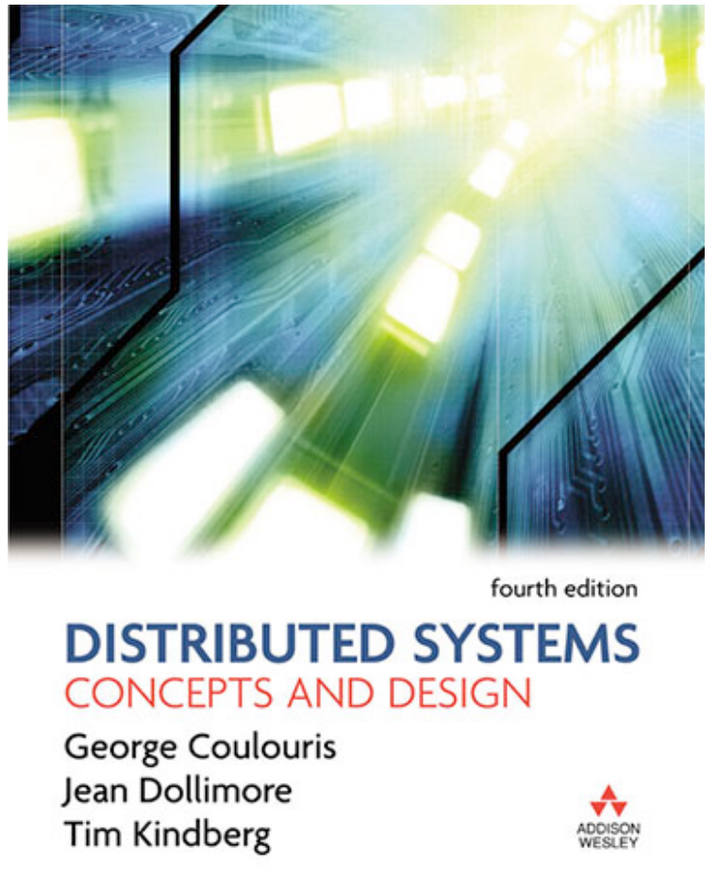
DTU Informatics
Department of Informatics and Mathematical Modelling

System Models

Nicola Dragoni
Embedded Systems Engineering
DTU Informatics



Architectural vs Fundamental Models

- Systems that are intended for use in real-world environments should be designed to function correctly in the widest possible range of circumstances and in the face of many possible difficulties and threats.

- An architectural model is concerned with the placement if its components and the relationships between them.

  - client-server systems

  - peer-to-peer systems

Architectural vs Fundamental Models

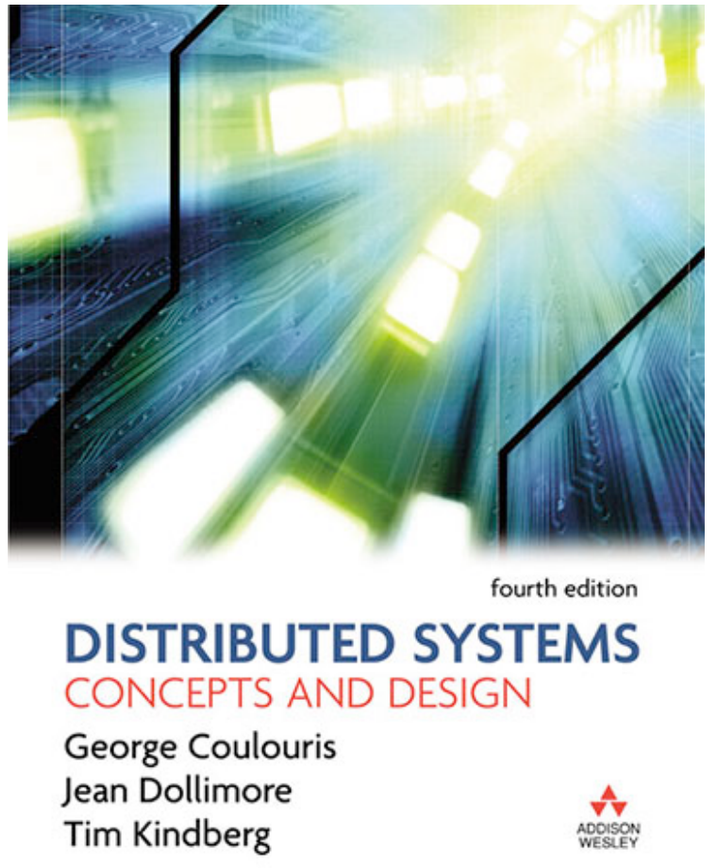
- Systems that are intended for use in real-world environments should be designed to function correctly in the widest possible range of circumstances and in the face of many possible difficulties and threats.

- An architectural model is concerned with the placement if its components and the relationships between them.
  - client-server systems
  - peer-to-peer systems

- Fundamental models are concerned with a more formal description of the properties that are common in all of the architectural models.

System Models

2.1 Introduction
2.2 Architectural Models
2.3 Fundamental Models



Architectural Models

- The architecture of a system is its structure in terms of separately specified components.

- An architectural model of a distributed system:

1. it simplifies and abstracts the functions of the individual components

2. it considers:

   - the placement of the components across a network of computers (seeking to define useful patterns for the distribution of data and workload)

   - the interrelationships between the components (i.e., their functional roles and the patterns of communication between them).

Initial Classification

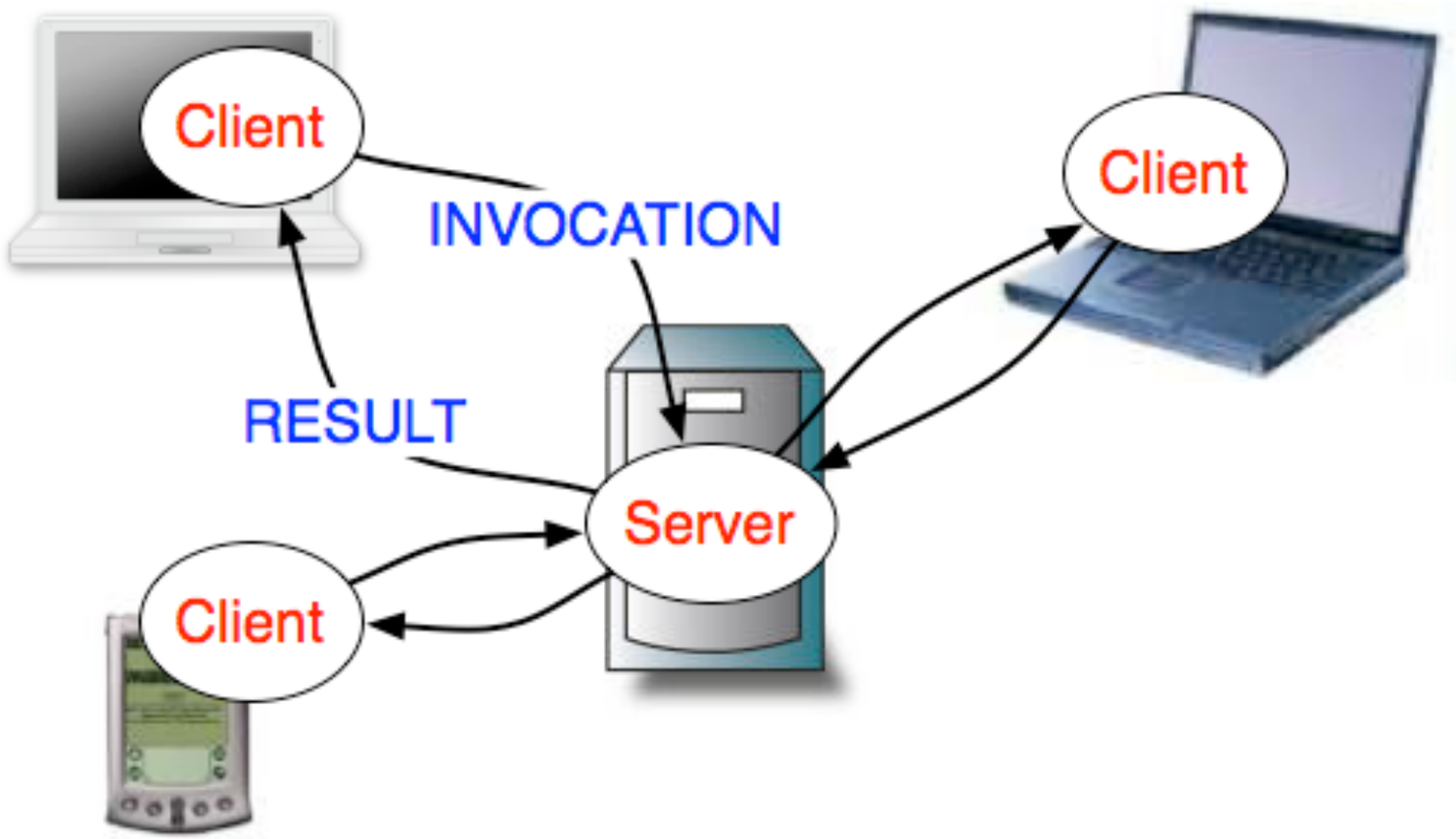
- Achieved by classifying processes as server processes, client processes and peer processes (processes that cooperate and communicate in a *symmetrical manner* to perform a task).

- This classification:
  - identifies the responsibilities of each component
  - helps to assess its workload
  - helps to determine the impact of failures in each component

- The results of this analysis can then be used to specify the placement of the processes in a manner that meets performance and reliability goals for the resulting system.

Architectural Model: Client-Server

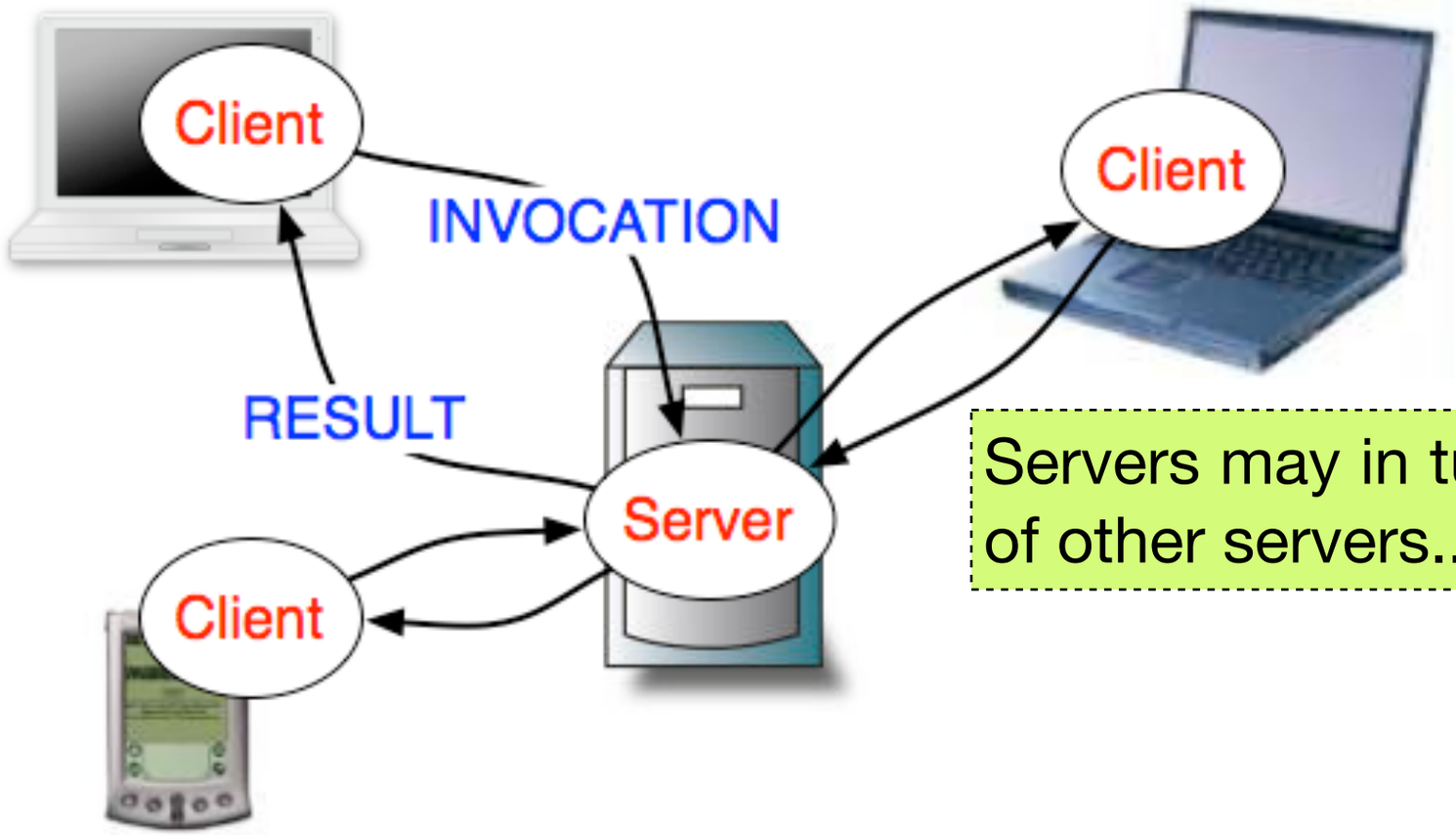
- Still the most widely employed architectural model.



Client processes interact with individual server processes in separate host computers in order to access the shared resources that they manage.

Architectural Model: Client-Server

- Still the most widely employed architectural model.



Client processes interact with individual server processes in separate host computers in order to access the shared resources that they manage.

On the Client-Server Role: Web Server Example

- Example 1: a Web server is often a client of a local file server that manages the files in which the web pages are stored.

On the Client-Server Role: Web Server Example

- Example 1: a Web server is often a client of a local file server that manages the files in which the web pages are stored.

- Example 2: Web servers and most Internet services are clients of the DNS service (which translates Internet Domain names to network addresses).

On the Client-Server Role: Web Server Example

- Example 1: a Web server is often a client of a local file server that manages the files in which the web pages are stored.

- Example 2: Web servers and most Internet services are clients of the DNS service (which translates Internet Domain names to network addresses).

- Example 3: search engine
  - Server: it responds to queries from browser clients
  - Client: it runs (in the background) programs called *web crawlers* that act as clients of other web servers

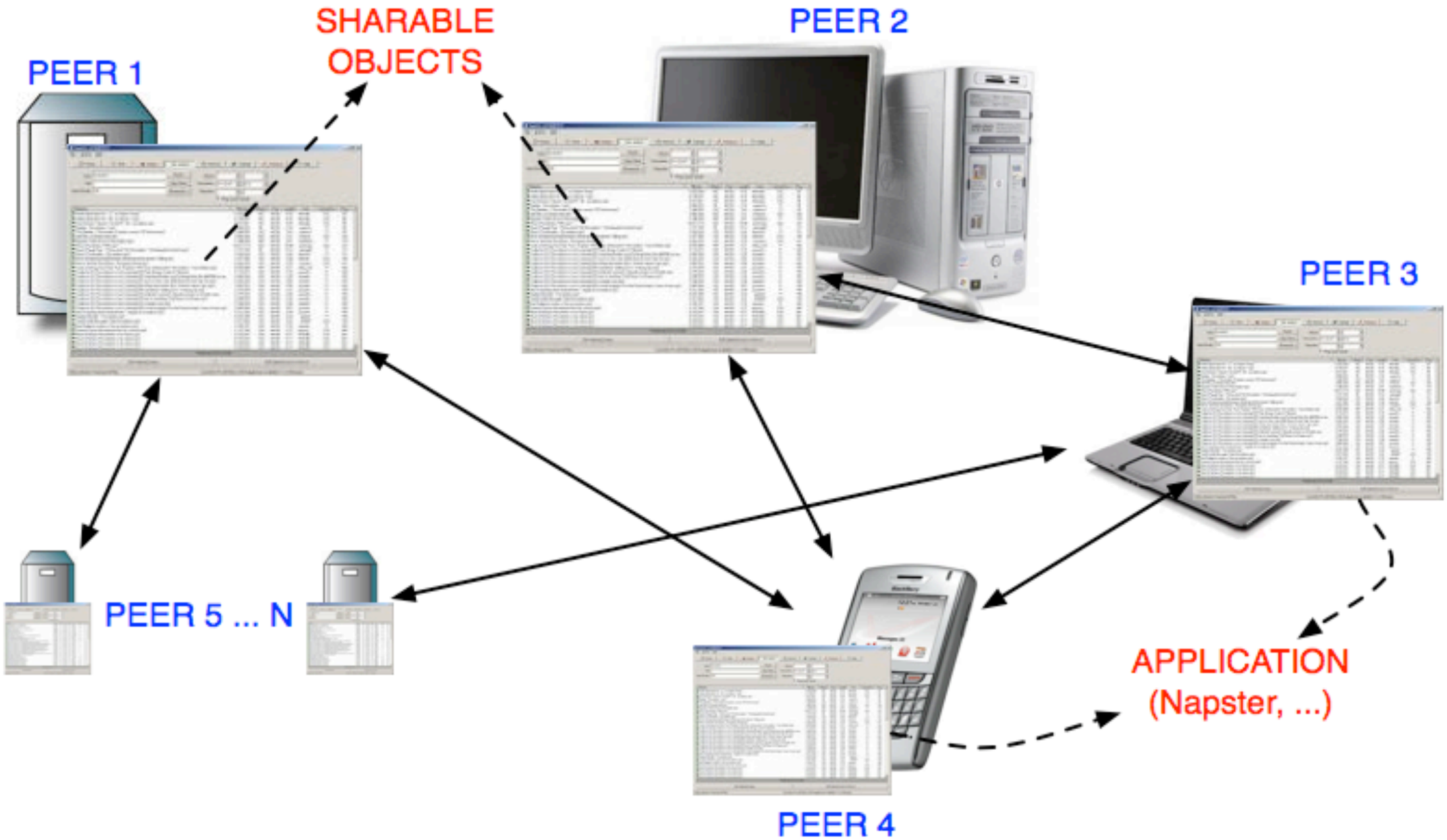
Architectural Model: Peer-to-Peer (P2P)

- All the processes involved in a task or activity play similar roles, interacting *cooperatively* as peers without any distinction between client and server processes or the computers that they run on.

- The hardware capacity and operating system functionality of today's desktop computers exceeds that of yesterday's servers.

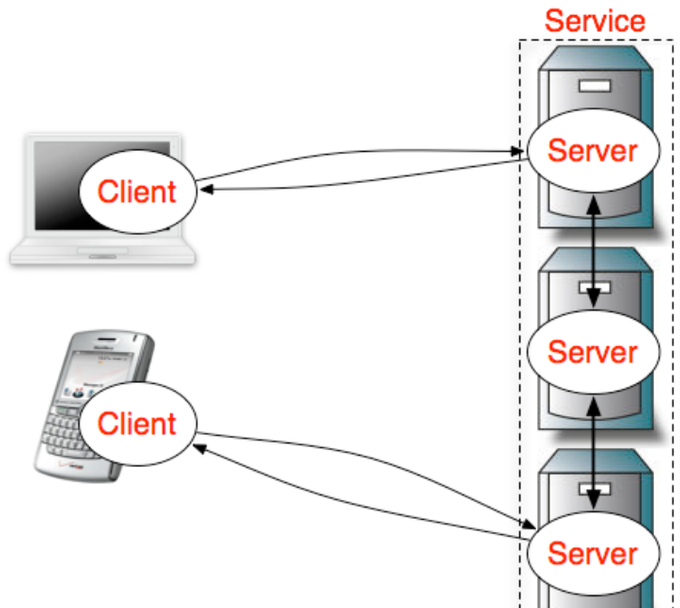
The aim of the P2P architecture is to exploit the resources (both data and hardware) in a large number of participating computers for the fulfillment of a given task or activity.

Distributed Application Based on a P2P Architecture

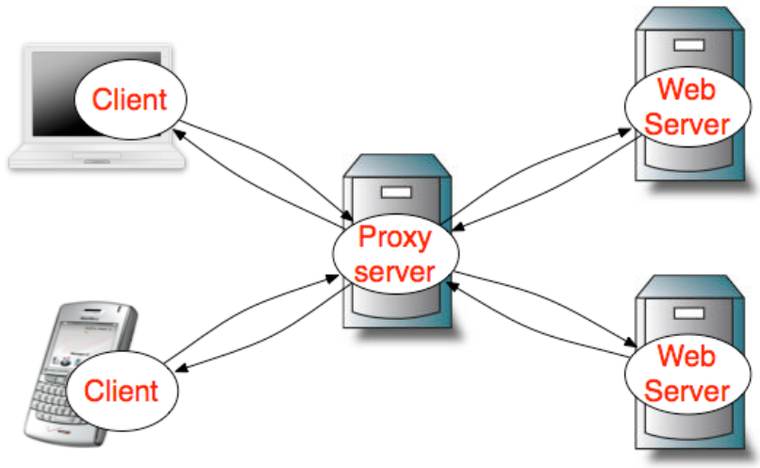


Variations of the Models

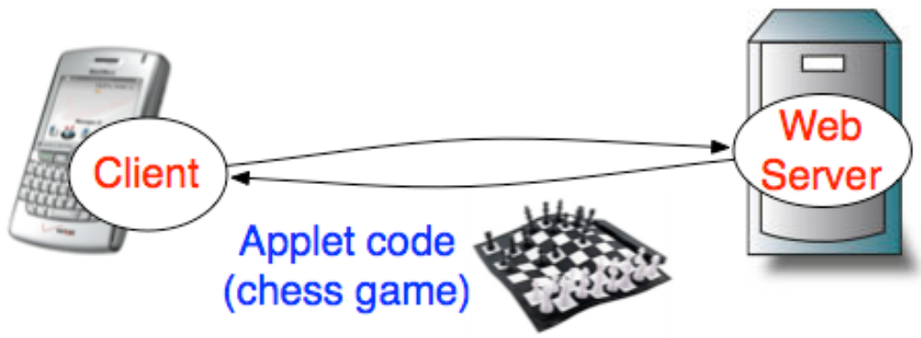
- Services provided by multiple servers



- Proxy server and caches



- Mobile code



- ...

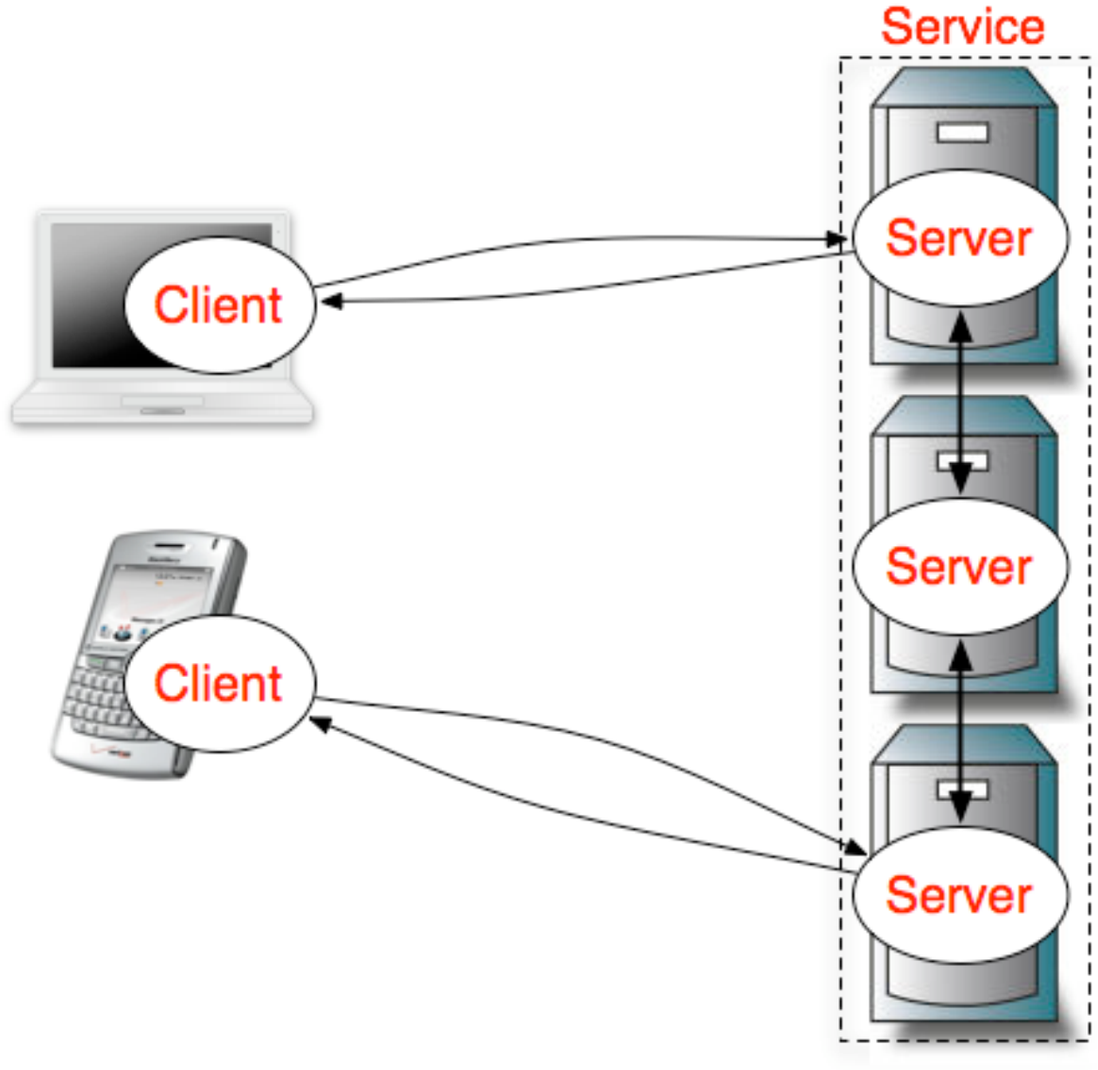
Variation: Services Provided by Multiple Servers

- Services may be implemented as several server processes in separate host computers interacting as necessary to provide a service to client processes.

- The servers may:

  1) partition the set of objects on which the service is based and distributed them between themselves (e.g. Web servers)

  2) they may maintain replicated copies of them on several hosts (e.g. SUN Network Information Service (NIS)).



Variation: Proxy Servers and Caches

- A cache is a store of recently used data objects that is closer than the objects themselves.
- Example 1: Web browsers maintain a cache of recently visited pages and other web resources in the client's local file system.

Variation: Proxy Servers and Caches

- A cache is a store of recently used data objects that is closer than the objects themselves.

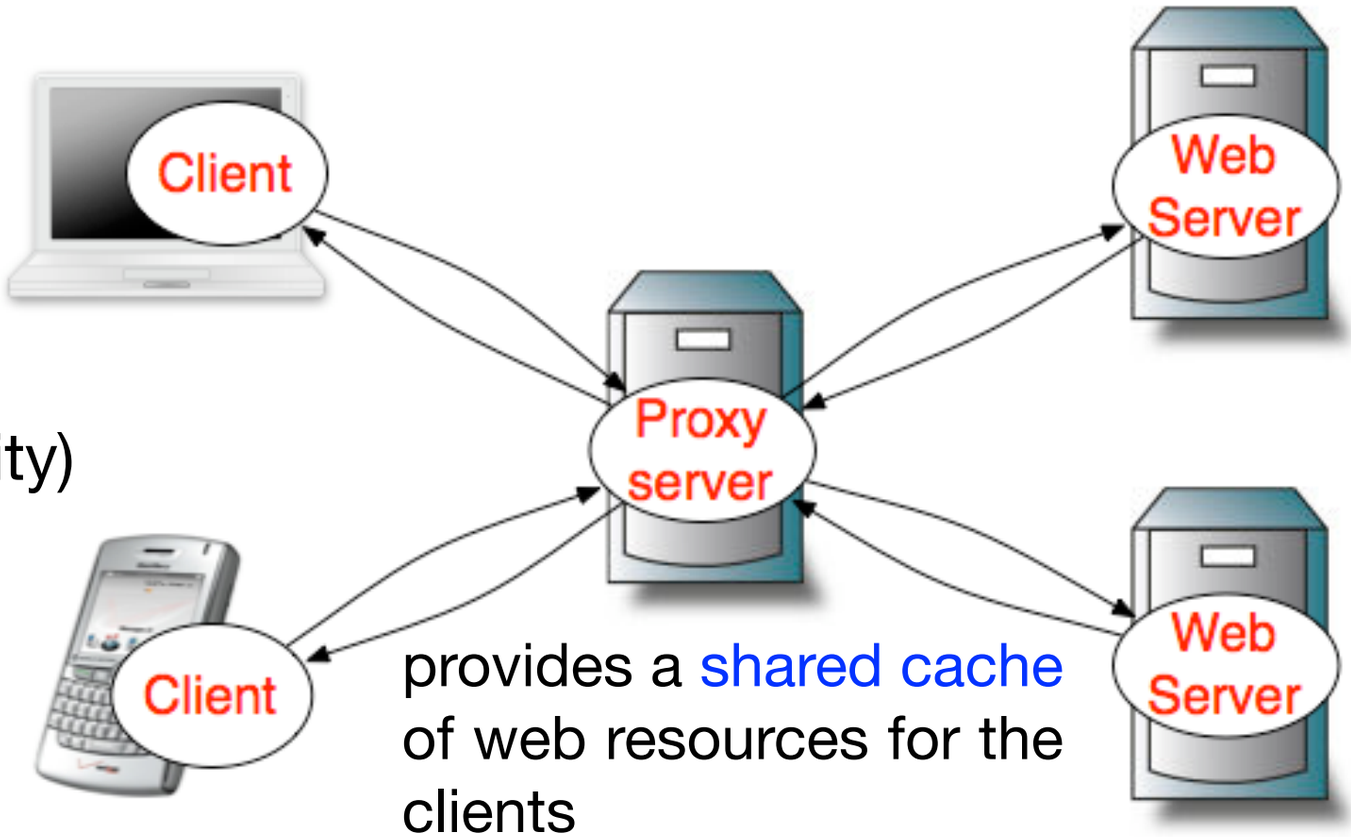
- Example 1: Web browsers maintain a cache of recently visited pages and other web resources in the client's local file system.

- Example 2: Web proxy server

  Purpose:

  1. To keep machines behind it anonymous (mainly for security)

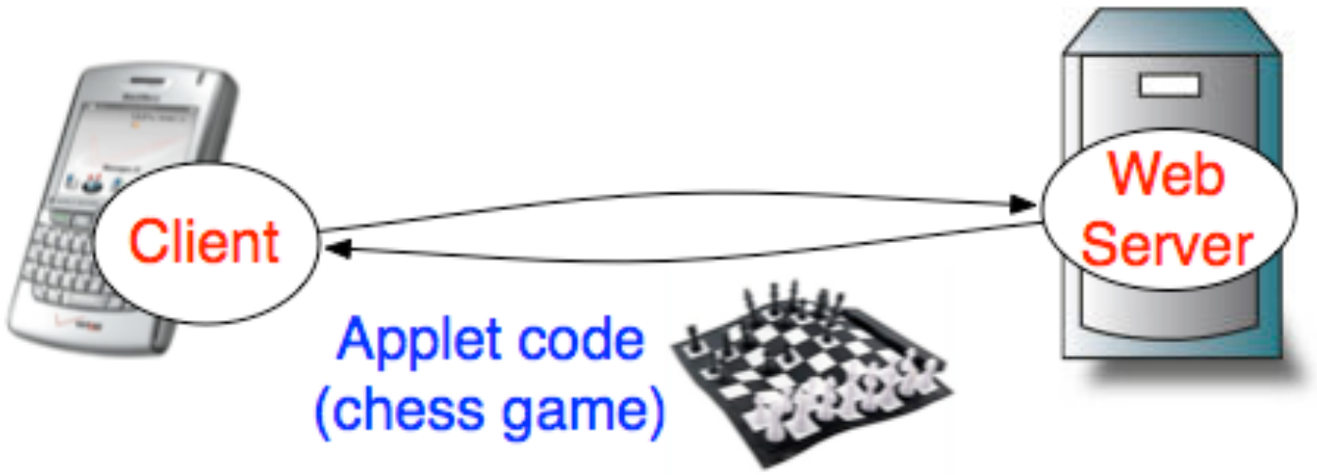
  2. To speed up access to a resource (via caching)



provides a shared cache of web resources for the clients

Variation: Mobile Code

A) Client request results in the downloading of applet code

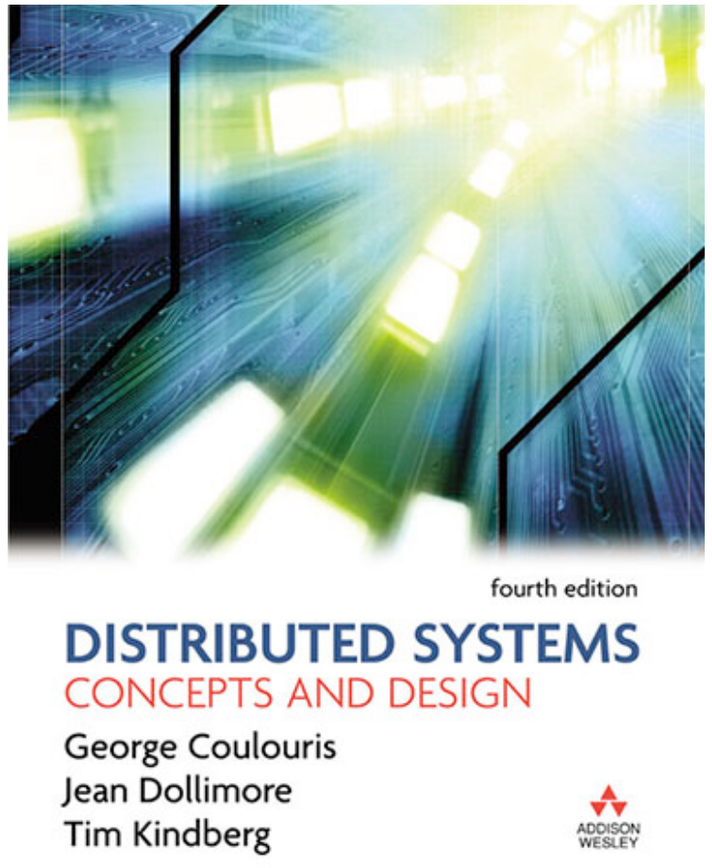


B) Client interacts with the applet

An advantage of running the downloaded code *locally* is that it can give good interactive response since it does not suffer from the delays or variability of bandwidth associated with network communication.

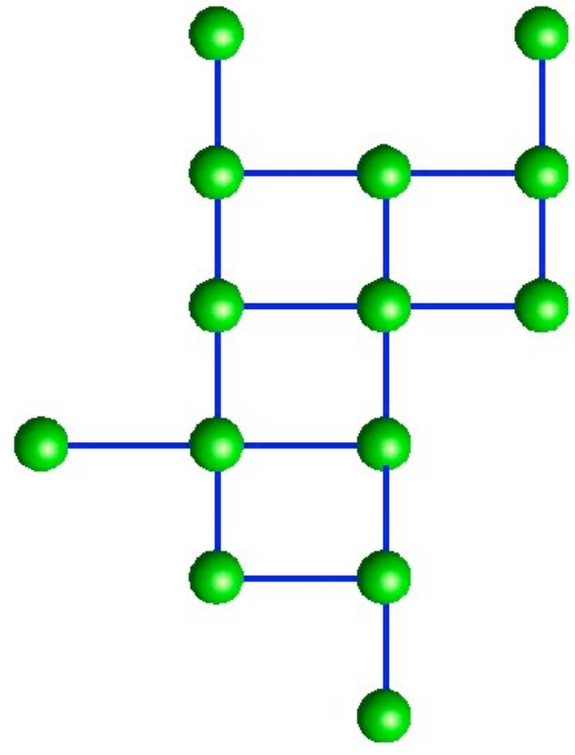
System Models

2.1 Introduction
2.2 Architectural Models
2.3 Fundamental Models



Fundamental (Abstract) Models

- The previous, quite different, models of systems share some fundamental properties.
  - For instance, all of them are composed of processes that communicate with one another by sending messages over a computer network.



- Fundamental models are concerned with a more formal description of the properties that are common in all the architectural models.

A model contains only the essential ingredients that we need to consider in order to understand and reason about some aspects of a system's behaviour.

Three Fundamental Models

- Interaction model: computation occurs within processes that interact by passing messages, resulting in communication (i.e., information flow) and coordination (synchronization and ordering of activities) between processes.

- Failure model: the correct operation of a distributed system is threatened whenever a fault occurs in any of the computers on which it runs or in the network that connects them.

- Security model: the openness of distributed systems exposes them to attack by both external and internal agents.

System Models

2.1 Introduction
2.2 Architectural Models
2.3 Fundamental Models
2.3.1 Interaction Model

Interaction Model

- Distributed systems are composed of many processes interacting in complex ways.
- For example:
  - Multiple server processes may cooperate with one another to provide a service
    - ➡ Domain Name Service, which partitions and replicates its data at servers throughout the Internet
  - A set of peer processes may cooperate with one another to achieve a common goal
    - ➡ A voice conference system that distributes streams of audio data in a similar manner, but with strict real-time constraints.

Distributed Algorithm

- Algorithm: a sequence of steps to be taken in order to perform a desired computation.

- Distributed systems composed of multiple processes can be described by a distributed algorithm: a definition of the steps to be taken by each of the processes of which the system is composed, *including the transmission of messages between them*.

- Messages are transmitted between processes to transfer information between them and to coordinate their activity.

Some Assumptions

- The rate at which each process proceeds **cannot** in general be predicted.
- The timing of the transmission of messages **cannot** in general be predicted.
- Each process has its own state, consisting of the set of data that it can access and update.
- The state belonging to each process is completely private (that is, it cannot be accessed or updated by any other processes).

Factors Affecting Interacting Processes

- Communication performance is often a limiting characteristic.

- It is impossible to maintain a single global notion of time.

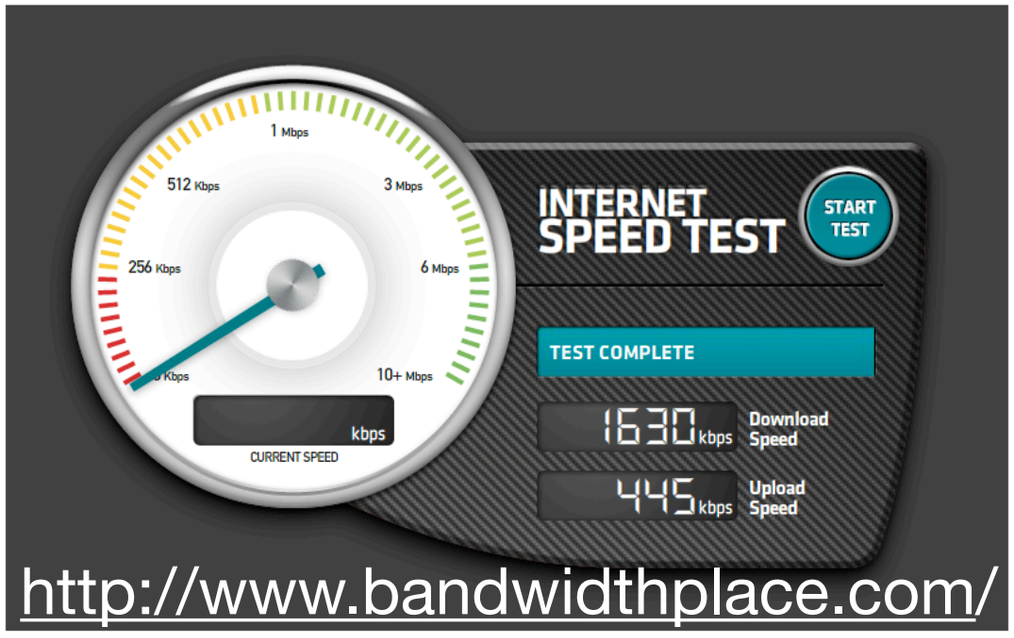
Performance of Communication Channels: Latency

- Latency: the delay between the start of a message's transmission from one process and the beginning of its receipt by another.

- The latency includes:
  - The time taken for the first of a string of bits transmitted through the network to reach its destination.
  - The delay in accessing the network, which increases significantly when the network is heavily loaded.
  - The time taken by the operating system communication services at both the sending and receiving processes, which varies according to the current load of the operating systems.

Performance of Communication Channels: Bandwidth

- The bandwidth of a computer network is the total amount of information that can be transmitted over it in a given time.
- Usually expressed in bit/s or multiples of it (kbit/s, Mbit/s, etc)
- When a large number of communication channels are using the same network, they have to share the available bandwidth.



http://www.bandwidthplace.com/

56 kbit/s	Modem / Dialup
1.5 Mbit/s	ADSL Lite
1.544 Mbit/s	T1
10 Mbit/s	Ethernet
11 Mbit/s	Wireless 802.11b
44.736 Mbit/s	T3
54 Mbit/s	Wireless-G 802.11g
100 Mbit/s	Fast Ethernet
155 Mbit/s	OC3
300 Mbit/s	Wireless-N 802.11n
622 Mbit/s	OC12
1000 Mbit/s	Gigabit Ethernet
2.5 Gbit/s	OC48
9.6 Gbit/s	OC192
10 Gbit/s	10 Gigabit Ethernet

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 a value of the current time.

- Therefore, two processes running on different computers can associate timestamps with their events.

- However, event if two processes read their clocks at the same time, their local clocks may supply different time values.

- This is because computer clocks drift from prefect time and, more importantly, their drift rates differ from one another.

- Clock drift rate: relative amount that a computer clock differs from a perfect reference clock.

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 opposite extreme positions provide a pair of simple models:
  - Synchronous distributed systems: strong assumption of time
  - Asynchronous distributed systems: no assumptions about time

Synchronous Distributed System

- A distributed system in which the following bounds are defined:
  - 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

- A distributed system in which there are no bounds on:
  - process execution speeds: each step may take an arbitrarily long time
  - message transmission delays: a message may be received after an arbitrarily long time
  - clock drift rates: the drift rate of a clock is arbitrary

- This exactly models the Internet, in which there is no intrinsic bound on server or network load and therefore on how long it takes, for example, to transfer a file using ftp, or to receive an email message.

- *Any solution that is valid for an asynchronous distributed system is also valid for a synchronous one*. **Why? What about the contrary?**

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.

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.

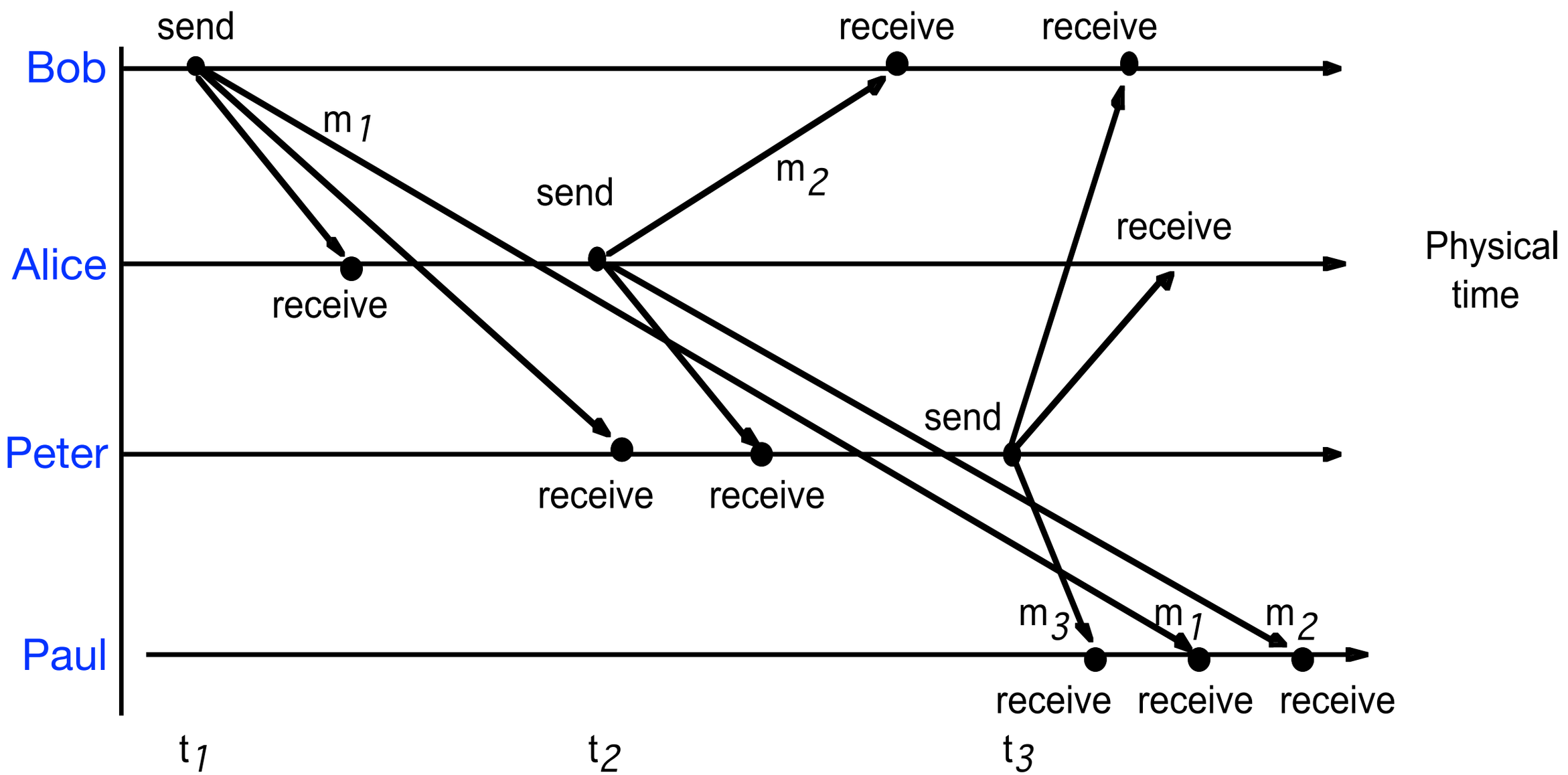
- The execution of a system can be described in terms of events and their ordering despite the lack of accurate clocks.

- Example [Real-Time Ordering of Events]: consider the following set of exchanges between a group of email users Bob, Alice, Peter, and Paul on a mailing list:

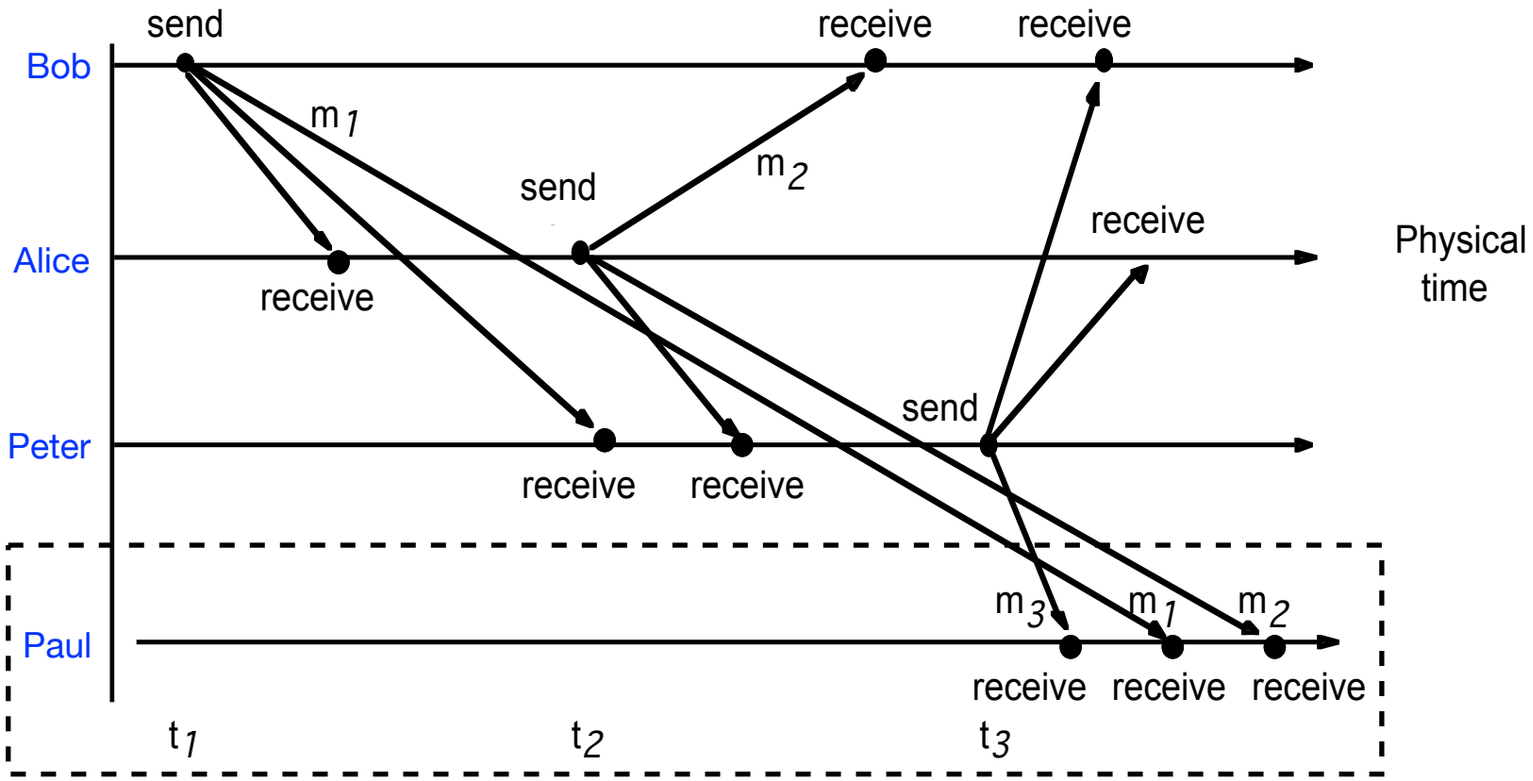
  1. Bob sends a message with the subject *Meeting*

  2. Alice and Peter reply by sending a message with the subject *Re: Meeting*

Example: Real-Time Ordering of Events



Example: Real-Time Ordering of Events (cont.)



Inbox	
From	**Subject**
Peter	Re:meeting
Bob	Meeting
Alice	Re: Meeting

System Models

2.1 Introduction
2.2 Architectural Models
2.3 Fundamental Models
2.3.2 Failure Model

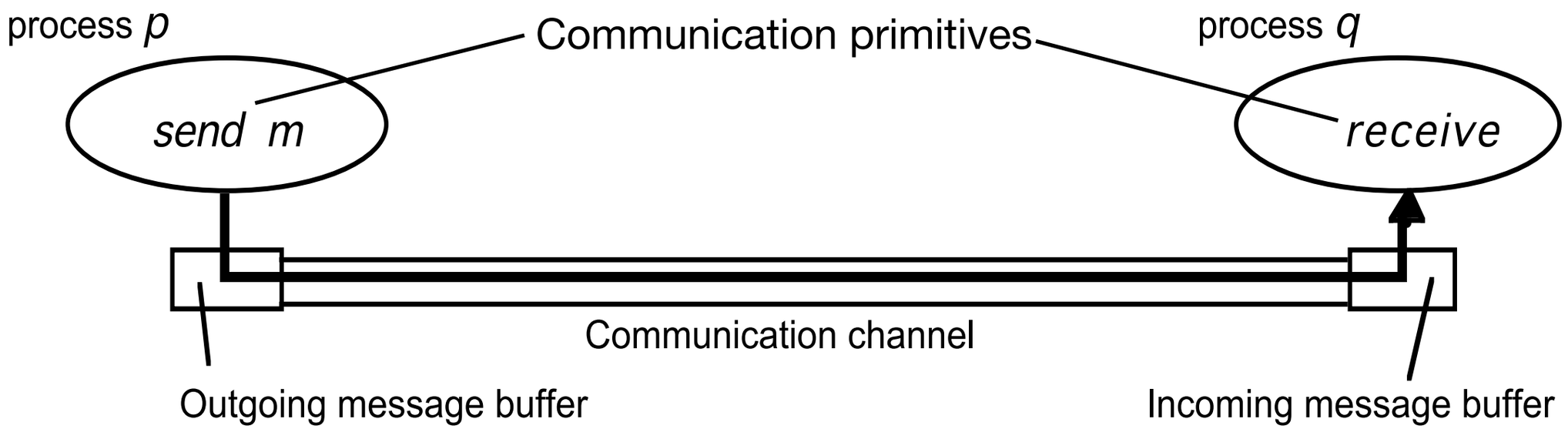
Failure Model

- 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).

- The failure model defines the ways in which failures may occur in order to provide an understanding of the effects of failures.

- Example of taxonomy of failures [Hadzilacos and Toueg, 1994]:
  - Omission failures: a process or communication channel fails to perform actions that it is supposed to do
  - Arbitrary failures: any type of error may occur
  - Timing failures: applicable in synchronous distributed systems

[Failure Model] Omission Failures



Class of failure	Affects	Description
Crash	Process	Process halts prematurely and remain halted.
Omission	Channel	A msg inserted in an outgoing msg 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.

[Failure Model] 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.

- Arbitrary failure of a process: the process arbitrarily omits intended processing steps or takes unintended processing steps.

- Communication channel arbitrary failures: message contents may be corrupted or non-existent messages may be delivered or real messages may be delivered more than once.

Class of failure	Affects	Description
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.

[Failure Model] 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.

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.

[Failure Model] 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.

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.

- In an asynchronous distributed systems, an overloaded server may respond too slowly, but *we cannot say that it has a timing failure since no guarantee has been offered*.

System Models

2.1 Introduction
2.2 Architectural Models
2.3 Fundamental Models
2.3.3 Security Model

Security Model

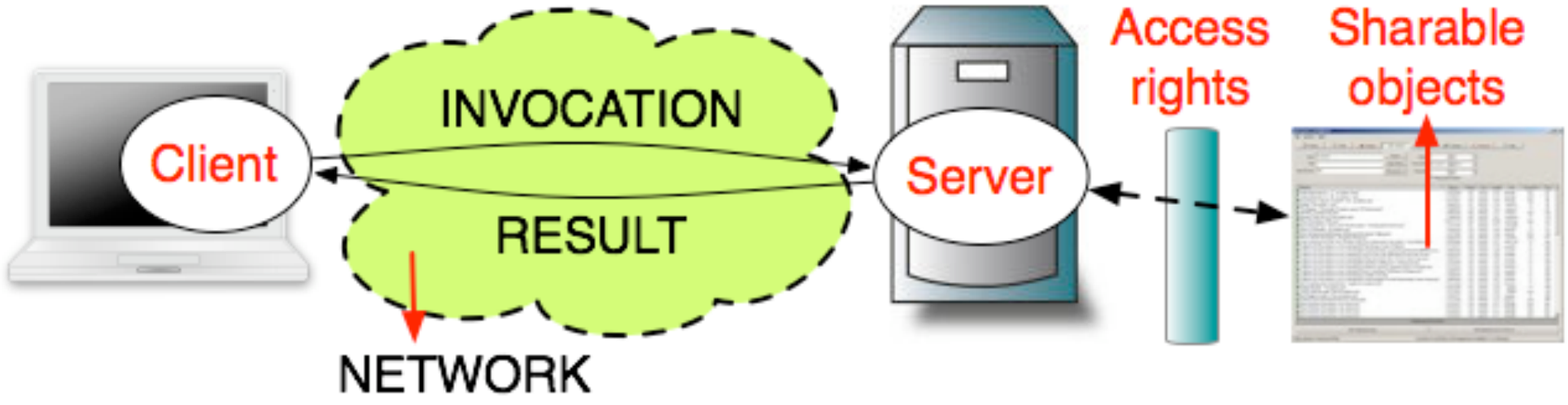
- Motivating factor for distributed systems: sharing of resources.

- Architectural model: distributed system described in terms of processes encapsulating (sharable) objects and providing access to them through interactions with other processes.

- This model provides the basis for the 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.

[Security Model] Protecting Objects

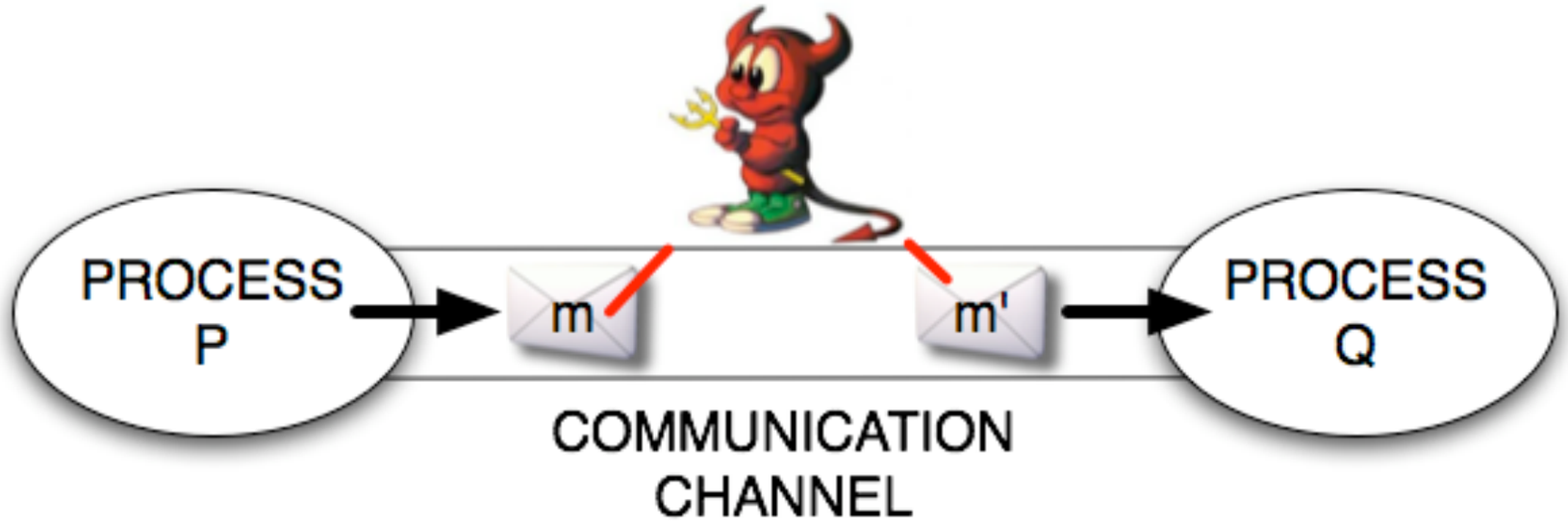


- Objects are intended to be used in different ways by different users:
  - some objects may hold a user's private data (such as the mailbox)
  - other objects may hold shared data such as web pages.
- Access rights specify who is allowed to perform the operations on an object (for instance, who is allowed to read and write its state).

[Security Model] Security Threats

- A security threat is a potential violation of security.

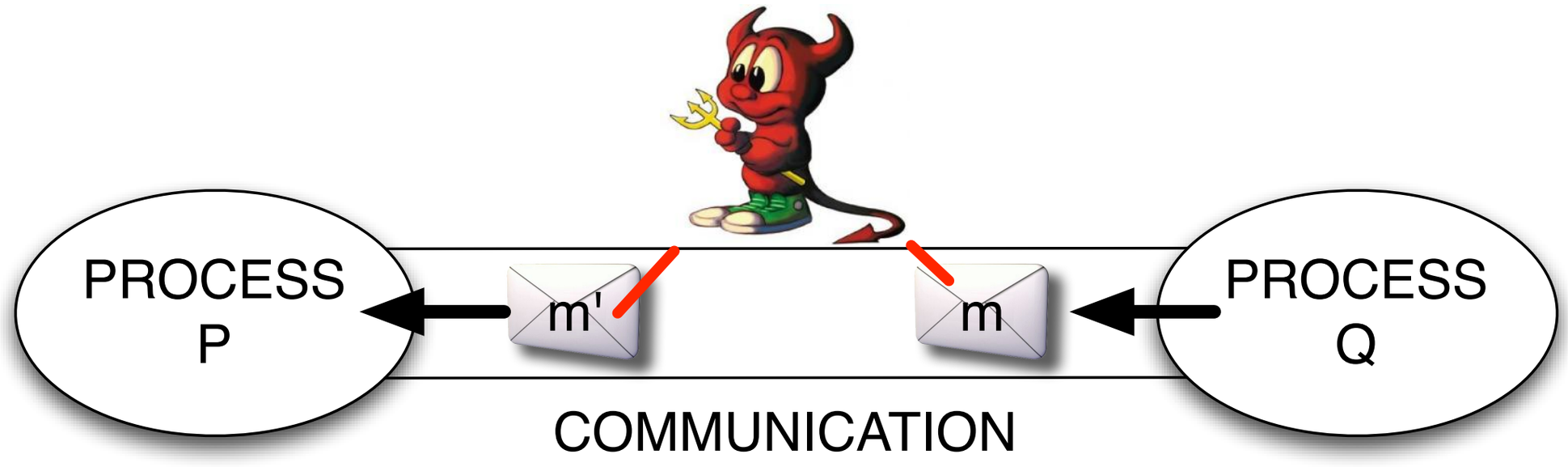
- To model security threats, we postulate an enemy (or adversary) that is *capable of sending any message to any process and reading or copying any message between a pair of processes*.



- The threats from potential enemy are discussed under the headings threats to processes, threats to communication channels and denial of service.

[Security Model] Threats to Processes

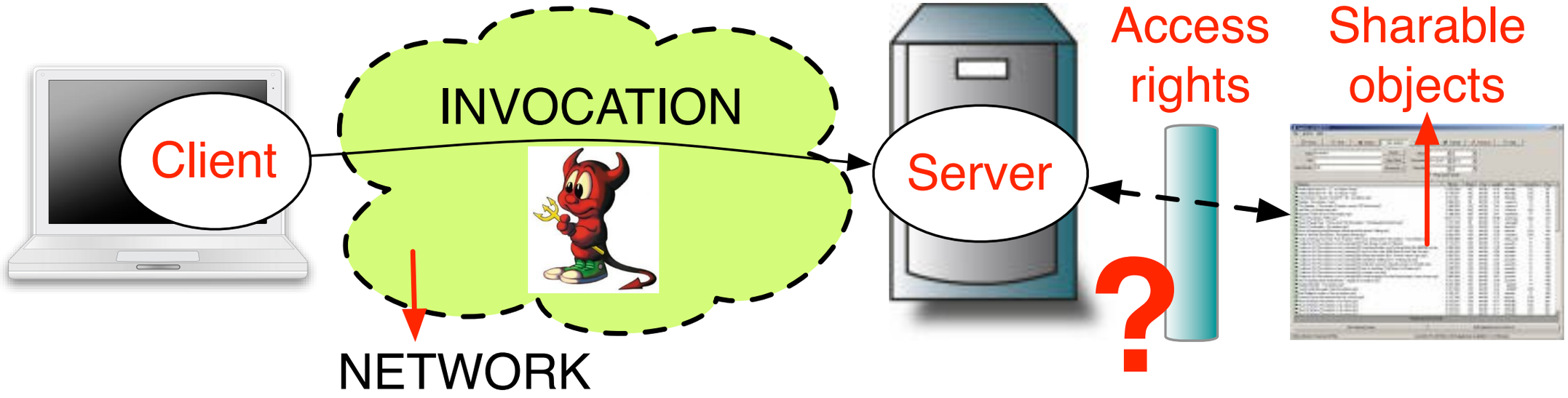
- A process receives a message from another process in the distributed system, and it is not able to determine the identity of the sender.



m’ = message with a forged source address

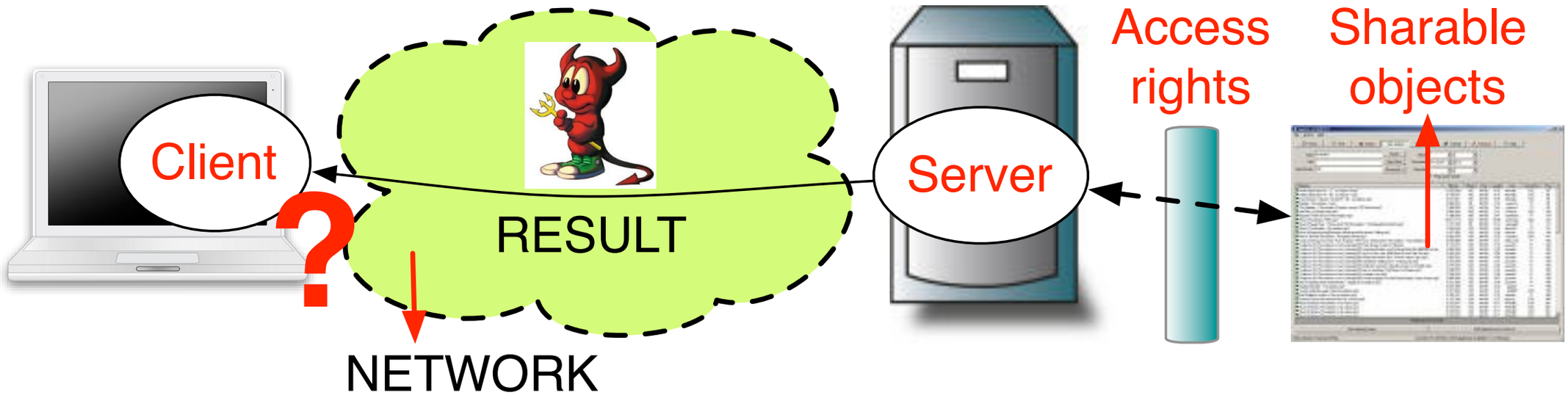
[Security Model] Threats to Processes

- This lack of reliable knowledge is a threat to the correct functioning of both servers and clients:
  - Server: without reliable knowledge of the sender's identity, a server cannot tell whether to perform the operation or reject it.



[Security Model] Threats to Processes

- This lack of reliable knowledge is a threat to the correct functioning of both servers and clients:
  - Client: when a client receives the result of an invocation from a server, it cannot necessarily tell whether the source of the result message is from the intended server or from an enemy, perhaps “spoofing” the mail server.



[Security Model] Threats to Communication Channels

- An enemy can copy, alter or inject messages as they travel across the network.
- Such attacks present a threat to the privacy and integrity of information as it travels over the network and to the integrity of the system.
  - Example: a result message containing a user's mail item might be revealed to another user or it might be altered to say something quite different.

[Security Model] Threats to Communication Channels

- An enemy can copy, alter or inject messages as they travel across the network.

- Such attacks present a threat to the privacy and integrity of information as it travels over the network and to the integrity of the system.
  - Example: a result message containing a user's mail item might be revealed to another user or it might be altered to say something quite different.

- Another form of attack is the attempt to save copies of messages and to reply them at a later time, making it possible to reuse the same message over and over again.
  - Example: someone could benefit by resending an invocation message requesting a transfer of a sum of money from one bank account to another.

[Security Model] 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 or message transmissions in a network, resulting in overloading of physical resources (network bandwidth, server processing capacity, ...).

- Intentions: delaying or preventing actions by other users.

[Security Model] 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 or message transmissions in a network, resulting in overloading of physical resources (network bandwidth, server processing capacity, ...).

- Intentions: delaying or preventing actions by other users.

Example: on August 6, 2009, Twitter was shut down for hours due to a DoS attack:

THU AUG 6TH

Ongoing denial-of-service attack 21 hours ago

We are defending against a denial-of-service attack, and will update status again shortly.

Update: the site is back up, but we are continuing to defend against and recover from this attack.

Update (9:46a): As we recover, users will experience some longer load times and slowness. This includes timeouts to API clients. We're working to get back to 100% as quickly as we can.

Update (4:14p): Site latency has continued to improve, however some web requests continue to fail. This means that some people may be unable to post or follow from the website.