Distributed Systems

a.y. 2023/2024

Distributed Systems: lecture 4,5,6

Models for Distributed Systems

Models ...

- Shared properties and common design problems can be represented in the form of descriptive models
- Each model is intended to provide an abstract, simplified but consistent description of a relevant aspect of the design

…two types of models…

- Architectural models

- Fundamentals models:
  - – interaction
  - – failure
  - – security

Remember ... No global time

message-based communication

Distributed Systems:

Models for Distributed Systems:
Architectures

Architecture

Concerned with:

- the placement of parts
- the relationships between them

(software architecture)

- The mapping onto the underlying network of computers

(system architecture)

- Processes and objects

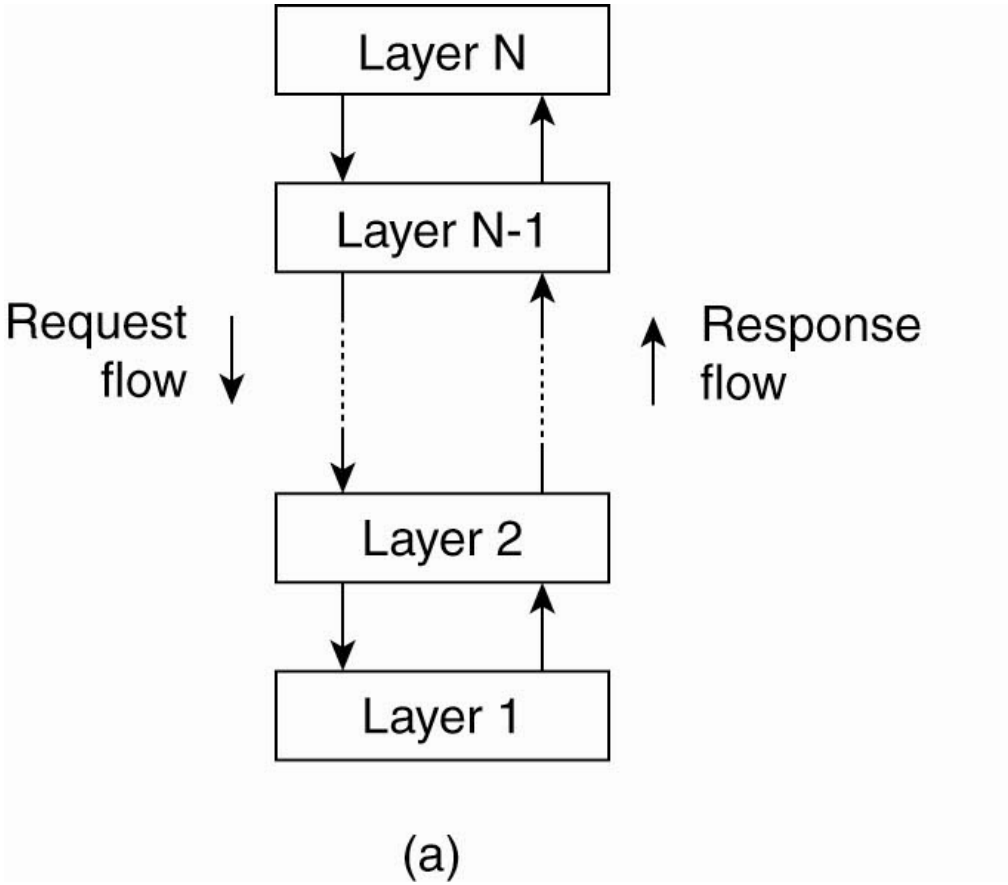
Architectural Styles

- A component is a modular unit with well-defined required and provided interface that is replaceable within its environment
- A connector is a mechanism that mediates communication, coordination, cooperation among components

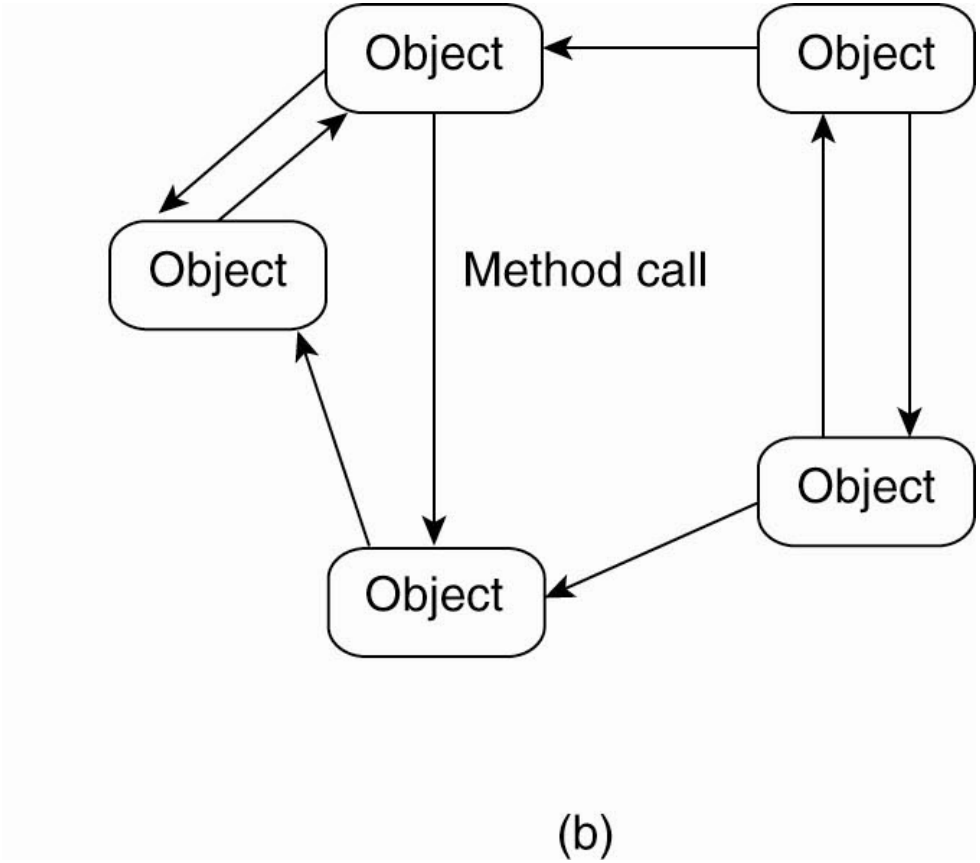
Architectural Styles

- Layered architectures
- Object-based architectures
- Data-centered architectures
- Event-based architectures

…layered…



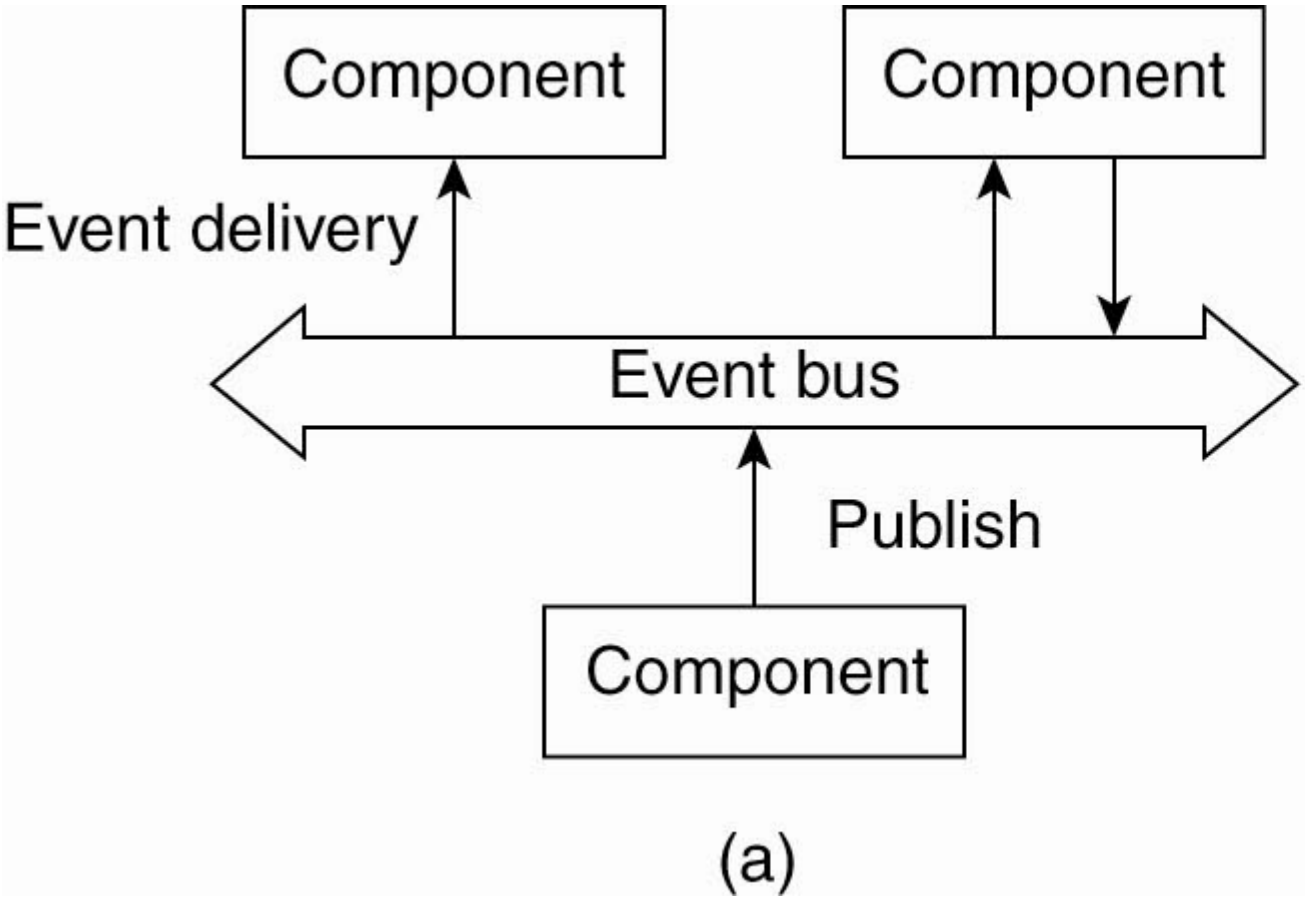
…object-based…



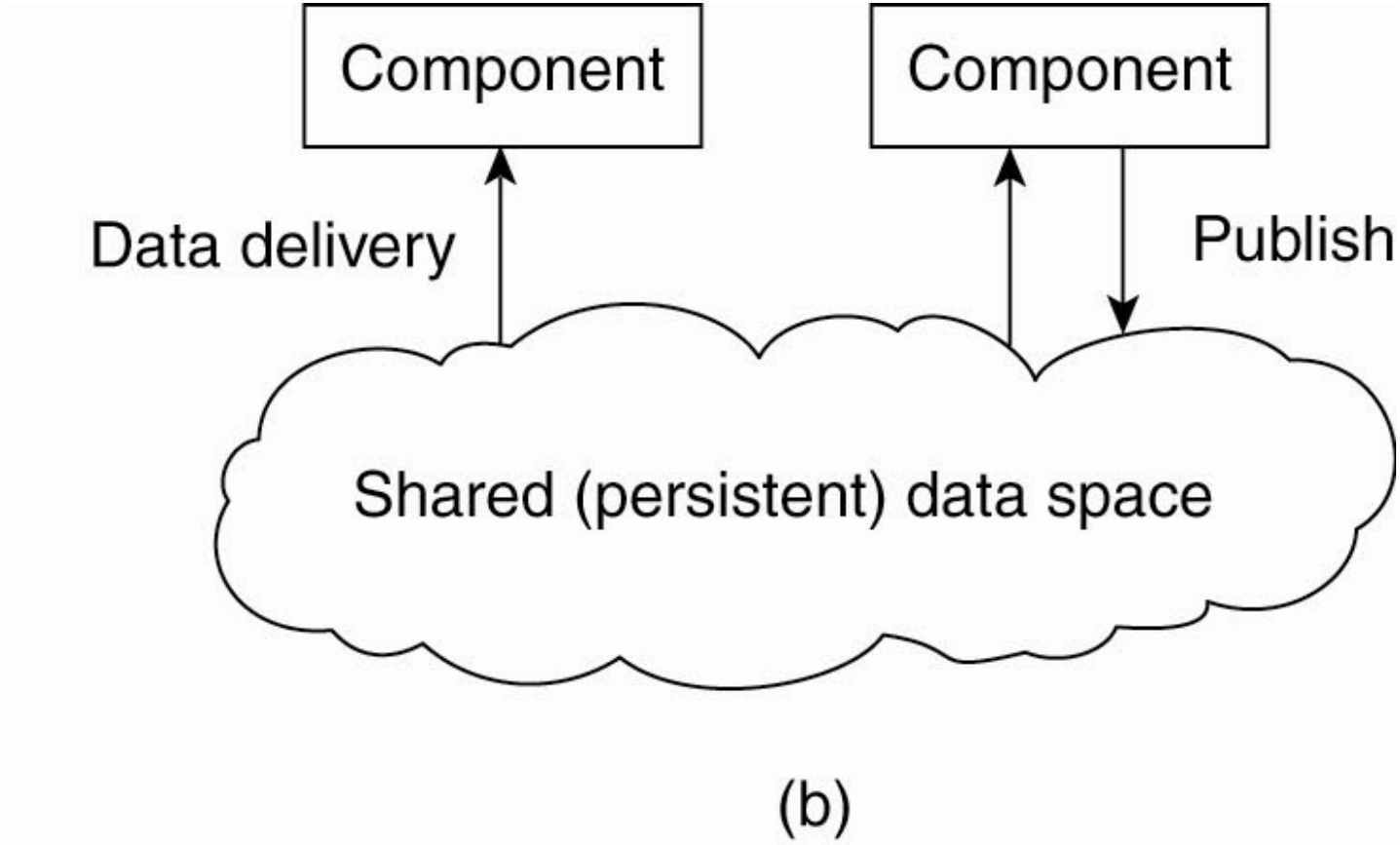
...data centered...

- A common repository
- Communication via a shared space
- Blackboard
- Shared file system/Shared data center

…event-based…


Component
Component
Event delivery
Event bus
Publish
Component
(a)

…shared data-space



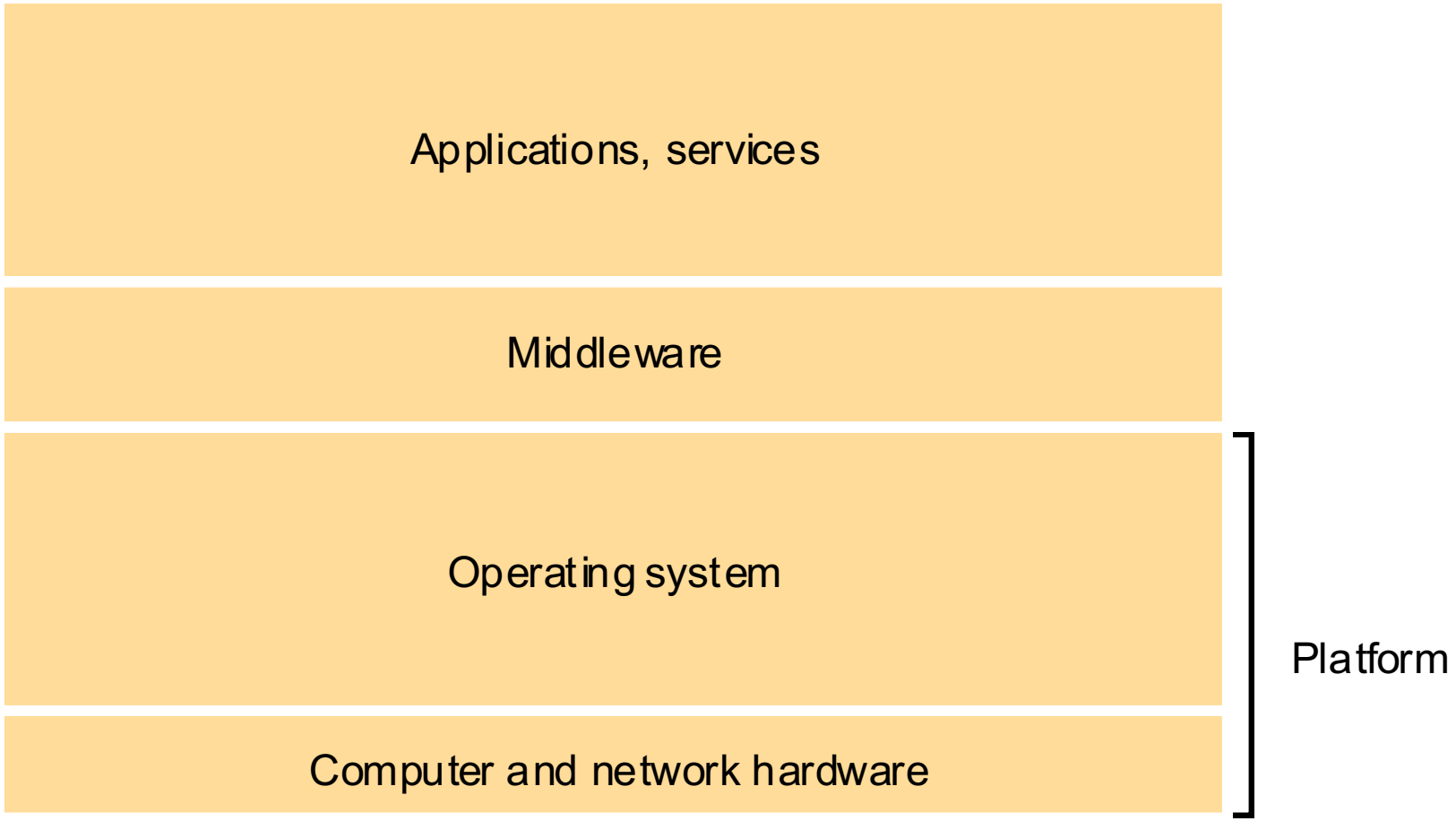
Process taxonomy

- Server process ...
- Client process ...
- Peer process ...
- Moving process ...

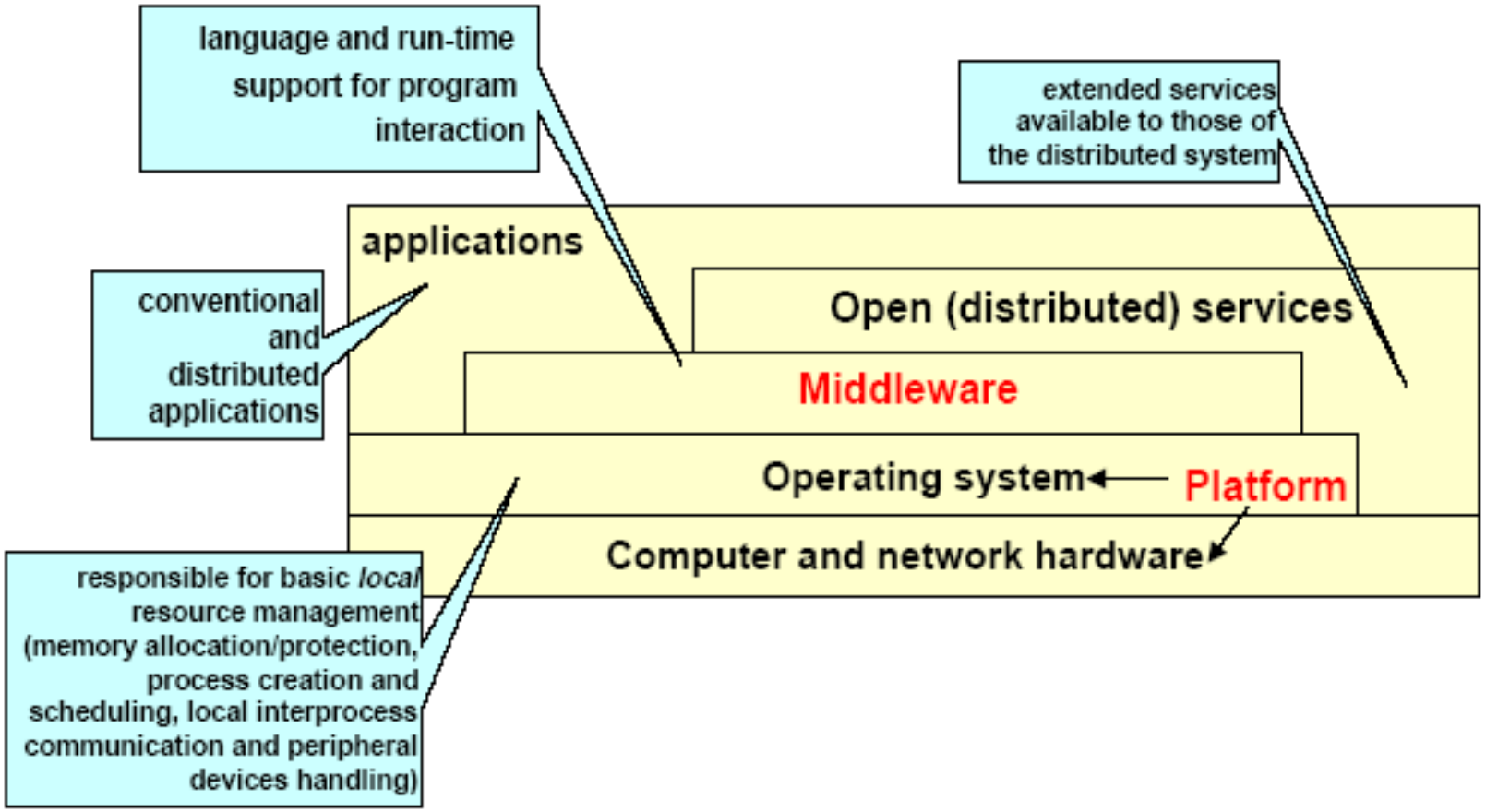
System architectures

- Software layers
- Architectural models
  - client-server, peer processes,…
  - mobile code, agents,...

Software Layers



Software Layers



Software layers

- Service layers
- Higher-level access services at lower layers
- Services can be located on different computers

Important layers

- Platform
  - lowest-level hardware+software
  - common programming interface, yet
  - different implementations of operating system facilities for co-ordination & communication
- Middleware
  - programming support for distributed computing

...middleware provides...

support for distributed processes/objects:

- suitable for applications programming
- communication via
  - remote method invocation (Java RMI), or
  - remote procedure call (Sun RPC)

services infrastructure for application programs

- naming, security, transactions, event notification,

Support a higher level of abstraction for:

- Communication between group of processes
- Notification of events
- Replication of shared data
- Multimedia data transmission in real time

- Object Management Group's Common Object Request Broker Architecture (CORBA)
- Microsoft Distributed Component Object Model (DCOM)
- Java RMI

The layered view...

- though appropriate for simple types of resource data sharing:
  - e.g. databases of names/addresses/exam grades
- too restrictive for more complex functions?
  - reliability, security, fault-tolerance, etc, need access to application's data
- see end-to-end argument [Saltzer, Reed & Clarke]

- “Some communiction-related functions can be completely and reliably implemented only with the knowledge and help of the application standing at the end points of the communication systems”

Saltzer, J.H., Read, D.P. and Clarke, D. (1984). End-to-End Arguements in System Deisgn. *ACM Transactions on Computer Systems* Vol. 2, No 4, pp. 277-288

- Checks, error-correction mechanism and security are at many levels ...
- Checking the correctness within the communication systems could not be enough

Architectural models

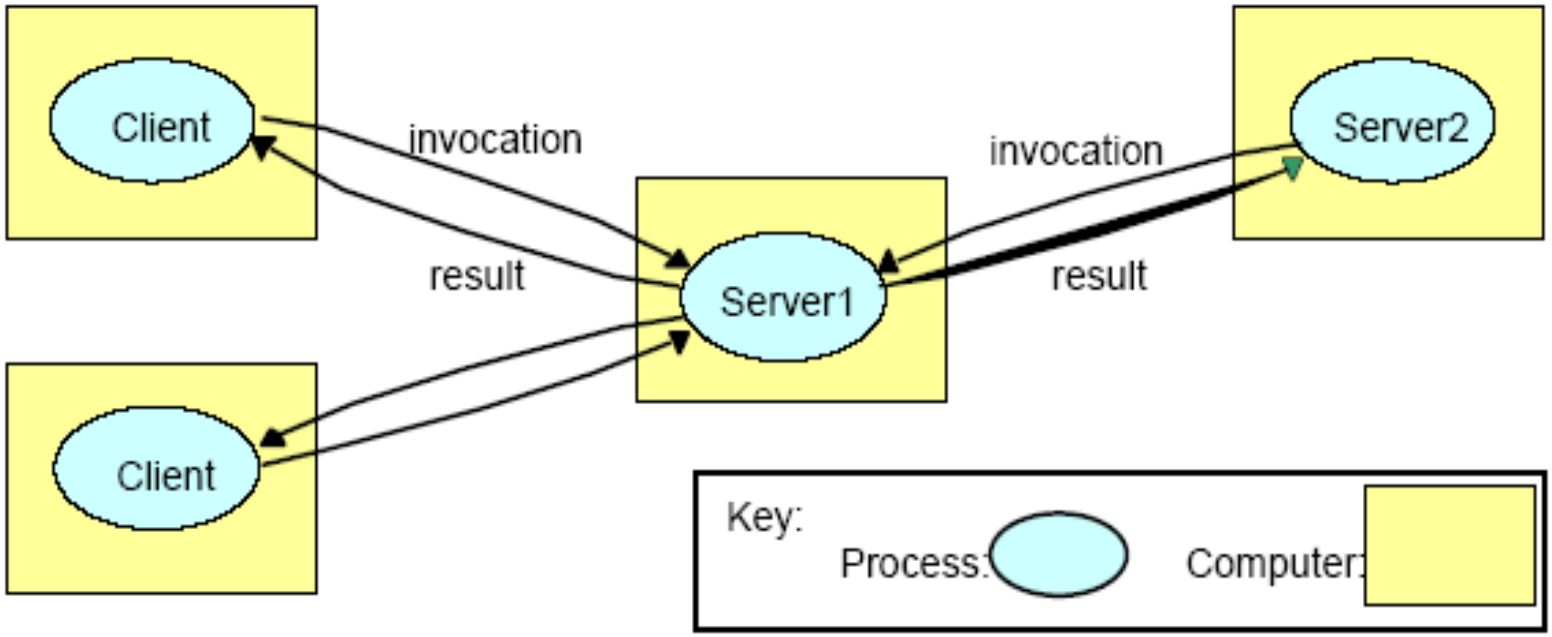
- Define
  - software components (processes, objects)
  - ways in which components interact
  - mapping of components onto the underlying network
- Why needed?
  - to handle varying environments and usage
  - to guarantee performance

…Main types of models…

- Client-server
  – first and most commonly used
- Multiple servers
  – to improve performance and reliability

- Proxy servers
  – to reduce load on network, provide access through firewall
- Peer processes
  – faster interactive response

Client server



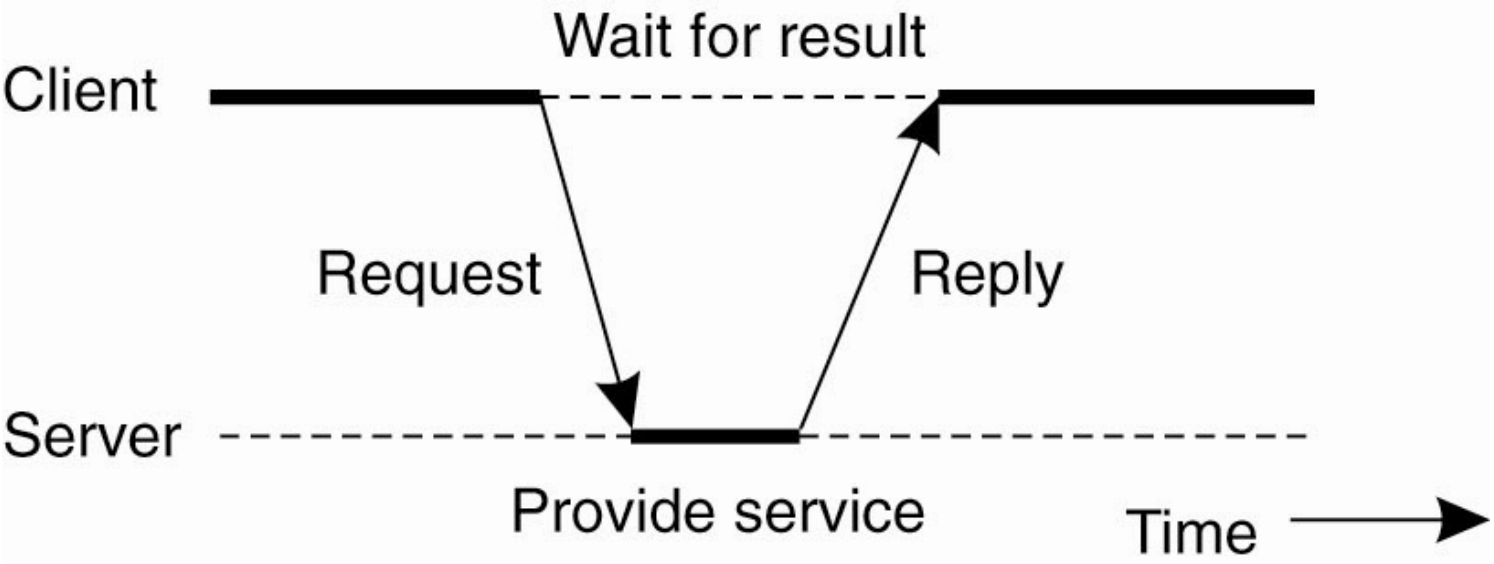
Server1 acts as **client** for Server2

…Main types of models…

- Client-server
  – first and most commonly used
- Multiple servers
  – to improve performance and reliability

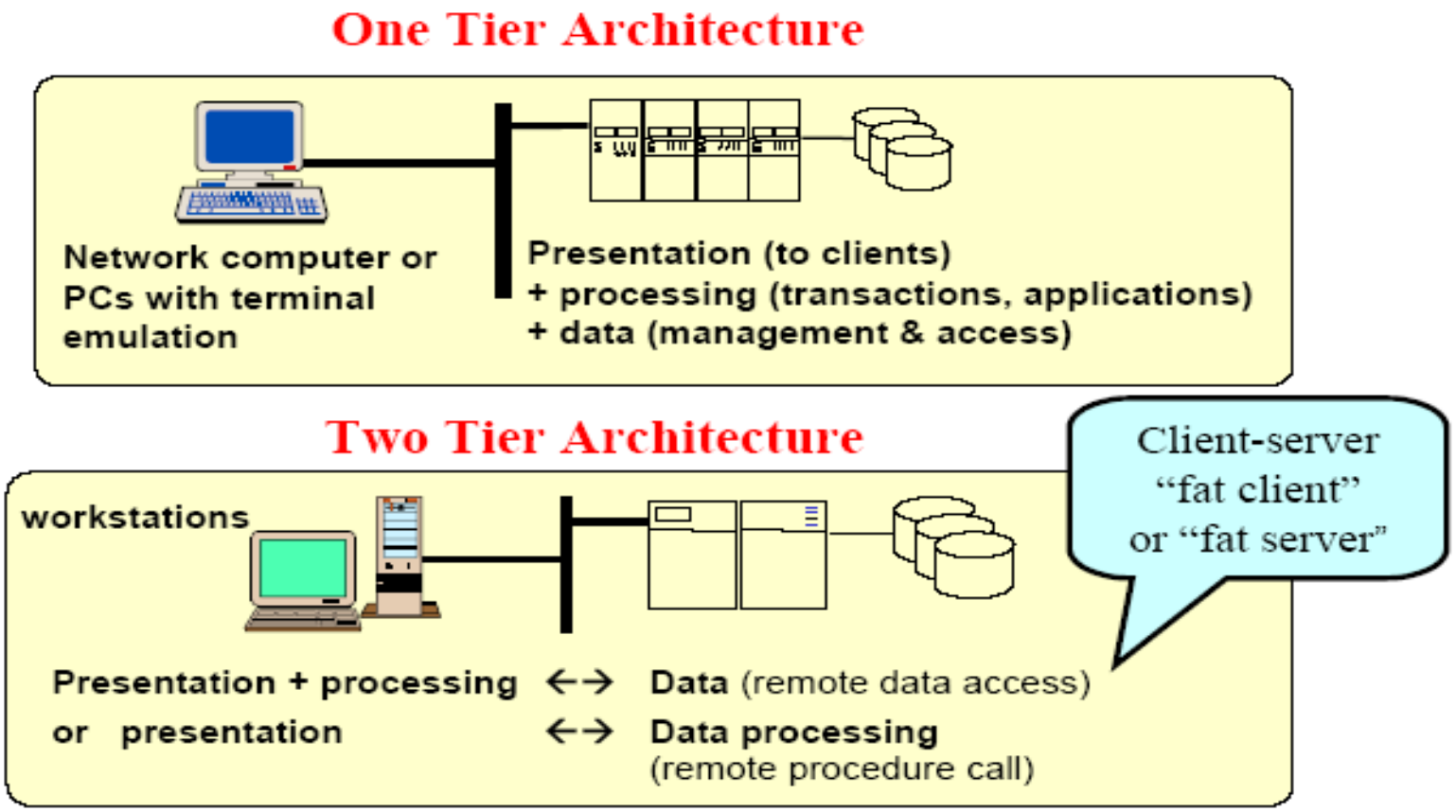
- Proxy servers
  – to reduce load on network, provide access through firewall
- Peer processes
  – faster interactive response

Centralized Architectures

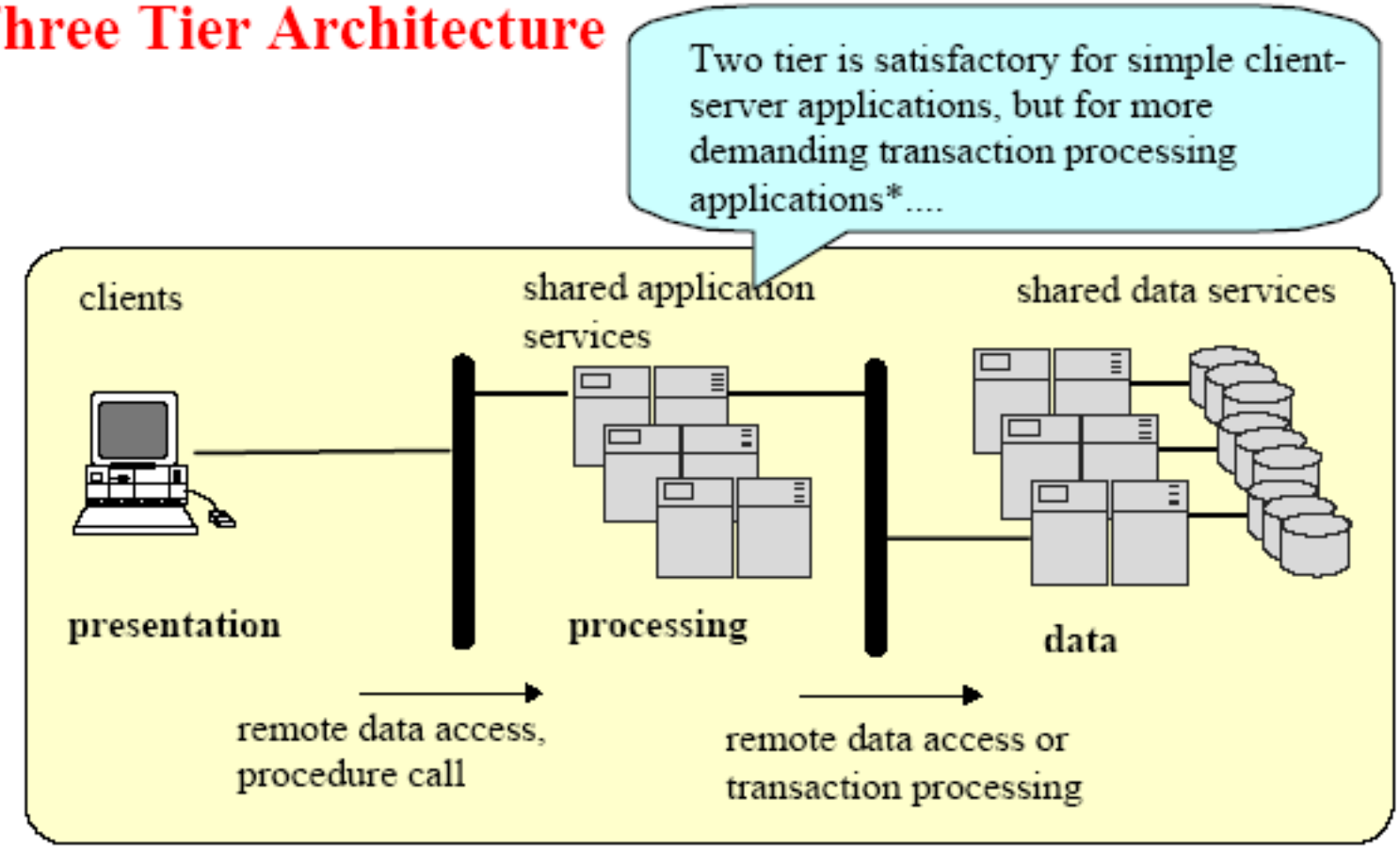


- Request-Reply Behaviour

Client-Server Systems



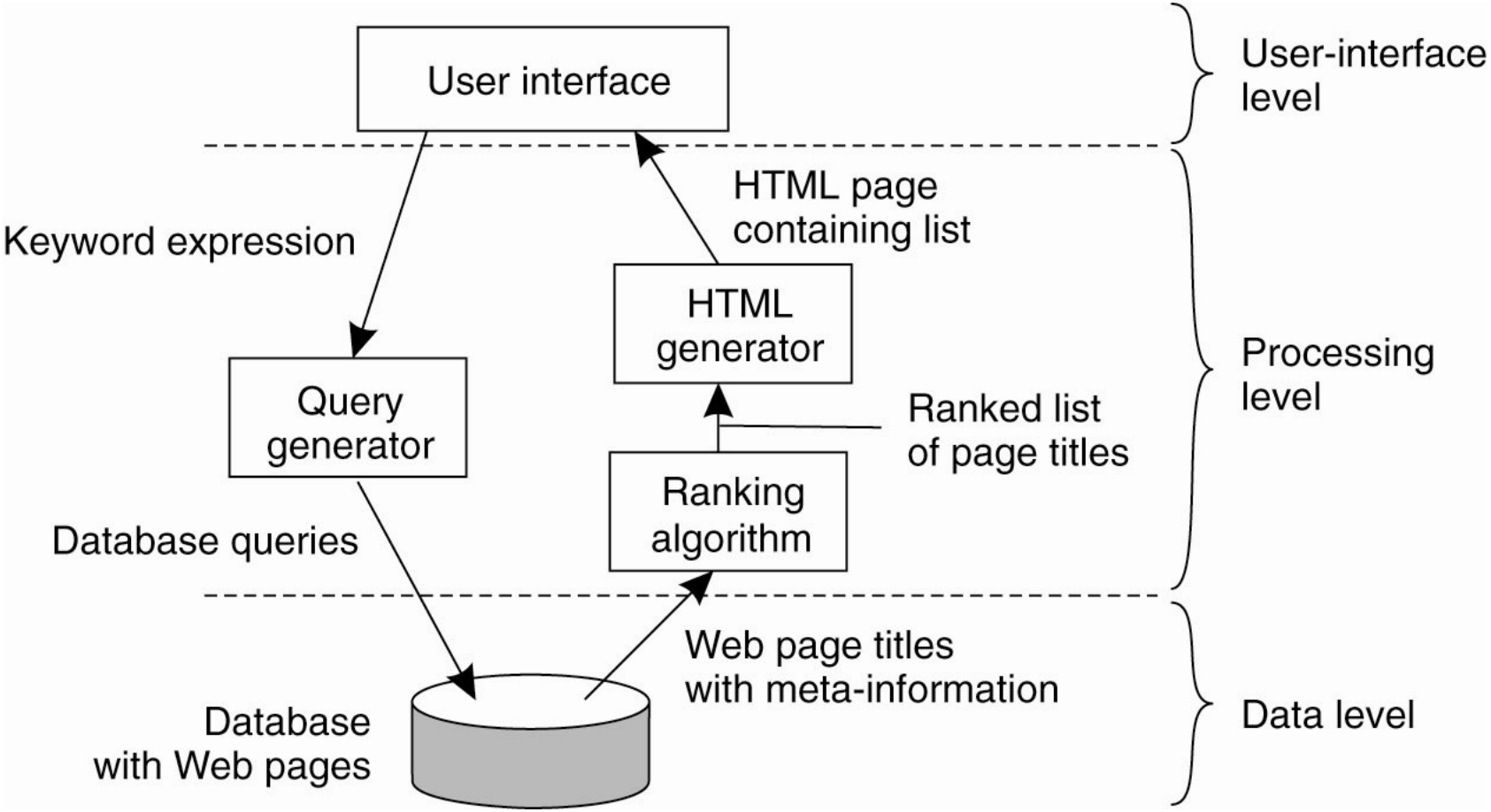
Three Tier Architecture



…Application Layering…

- The user-interface level
- The processing level
- The data level

…Application Layering…

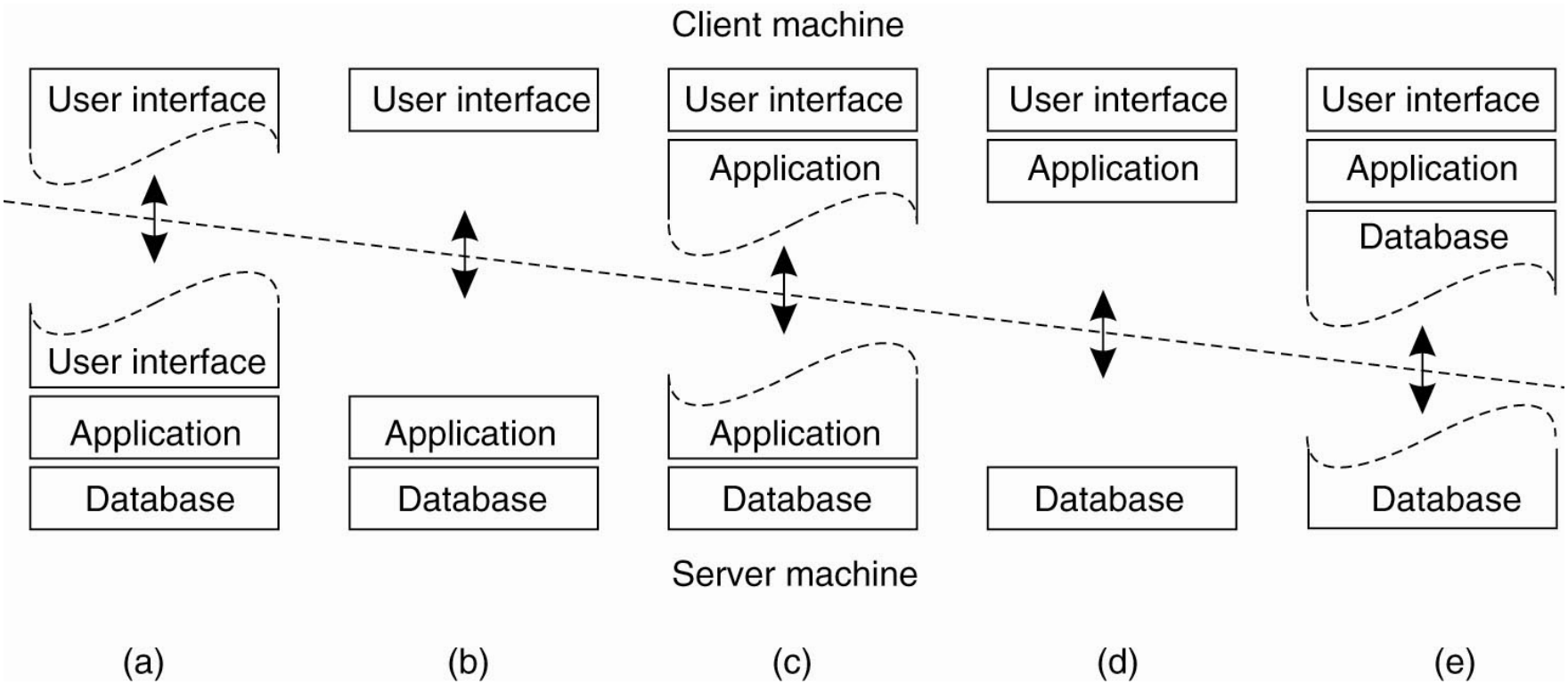


Multitiered Architectures

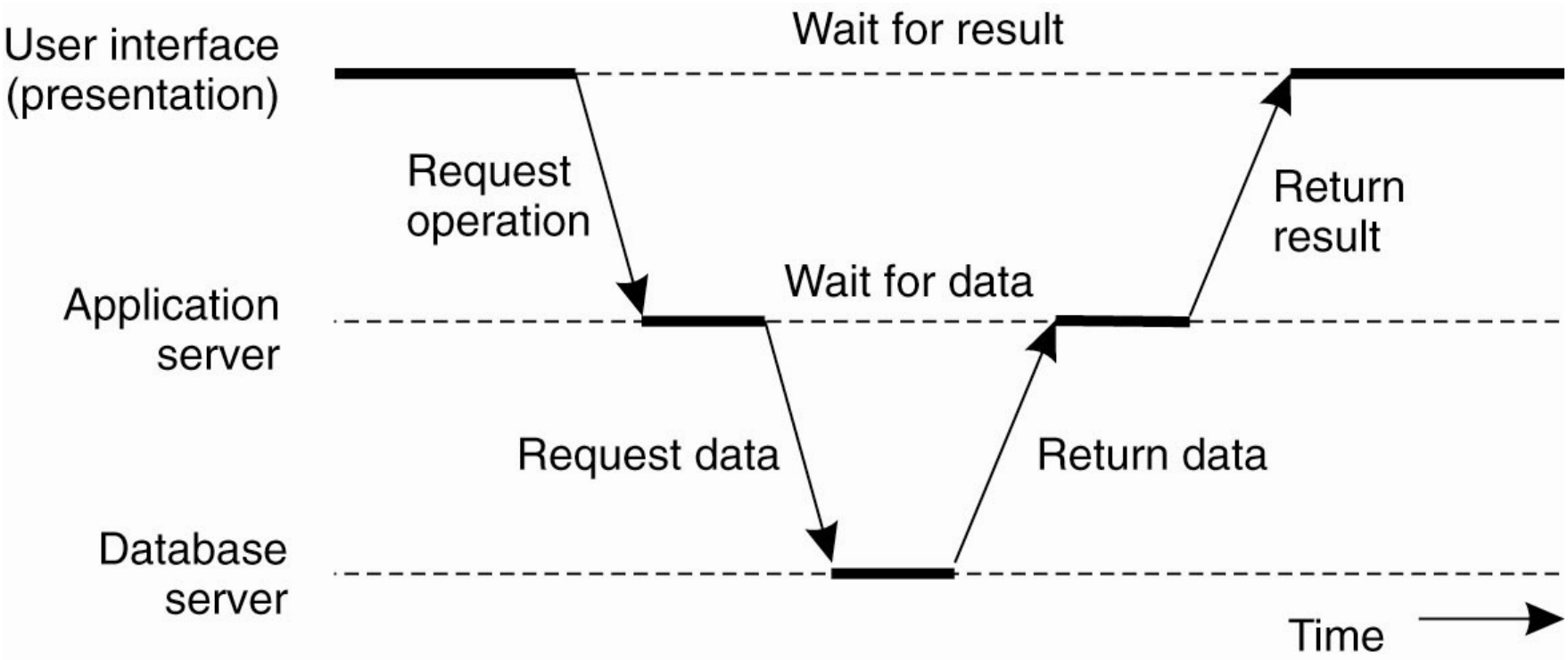
The simplest organization is to have only two types of machines:

- A client machine containing only the programs implementing (part of) the user-interface level
- A server machine containing the rest, i.e., the programs implementing the processing and data level

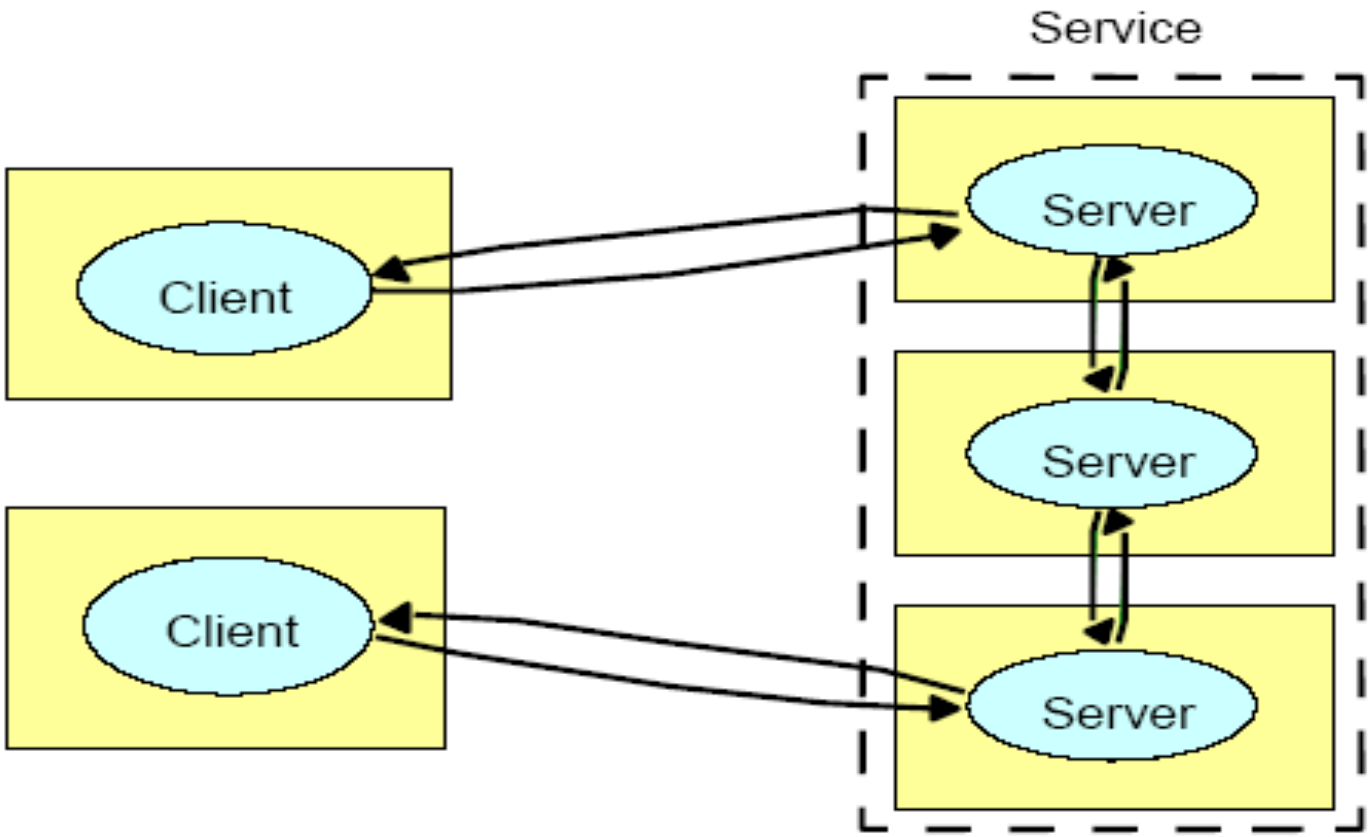
. . . Multitiered Architectures . . .



Multitiered Architectures (3)

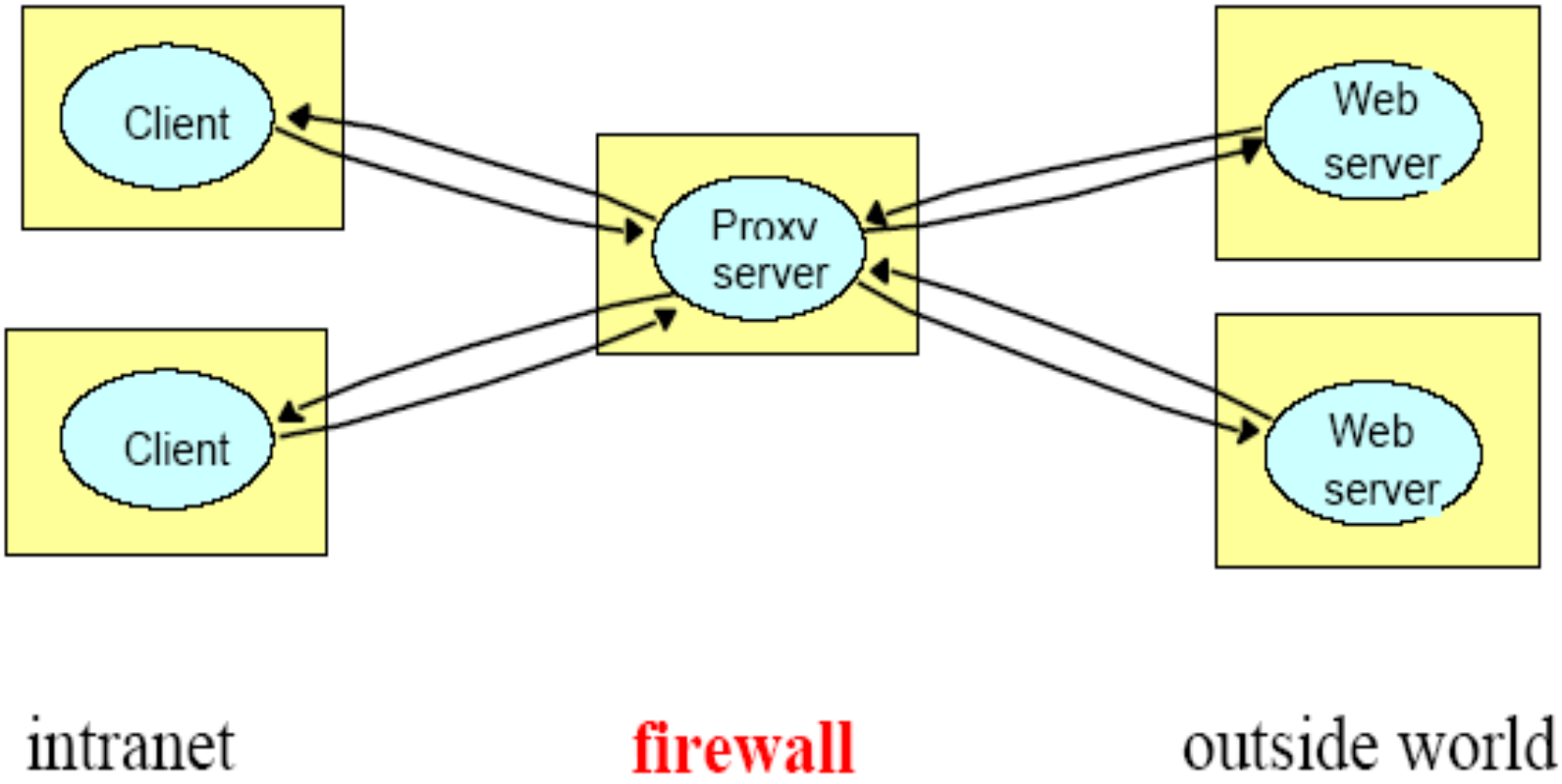


Multiple servers

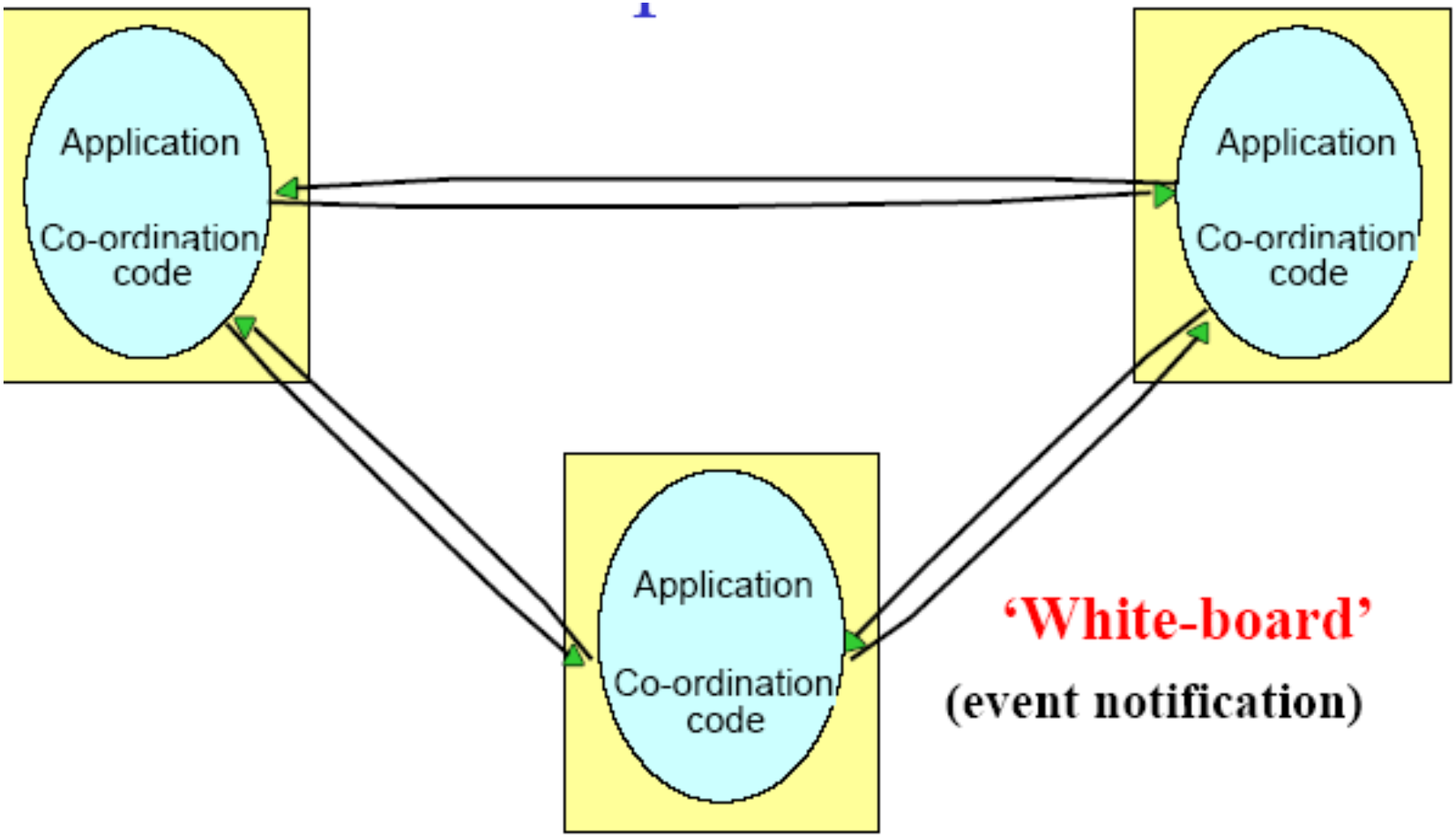


Servers may **interact**

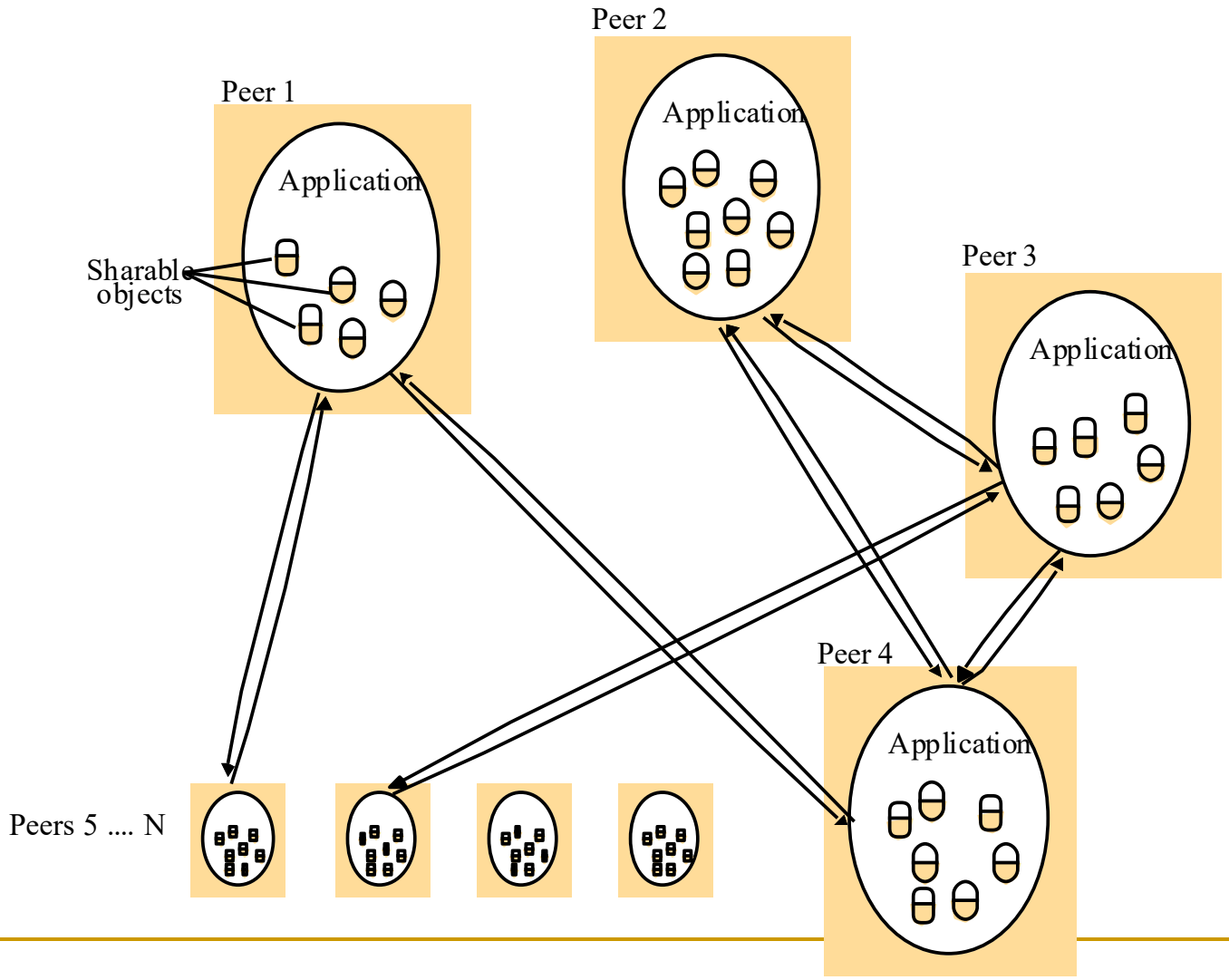
Proxy servers



Peer processes



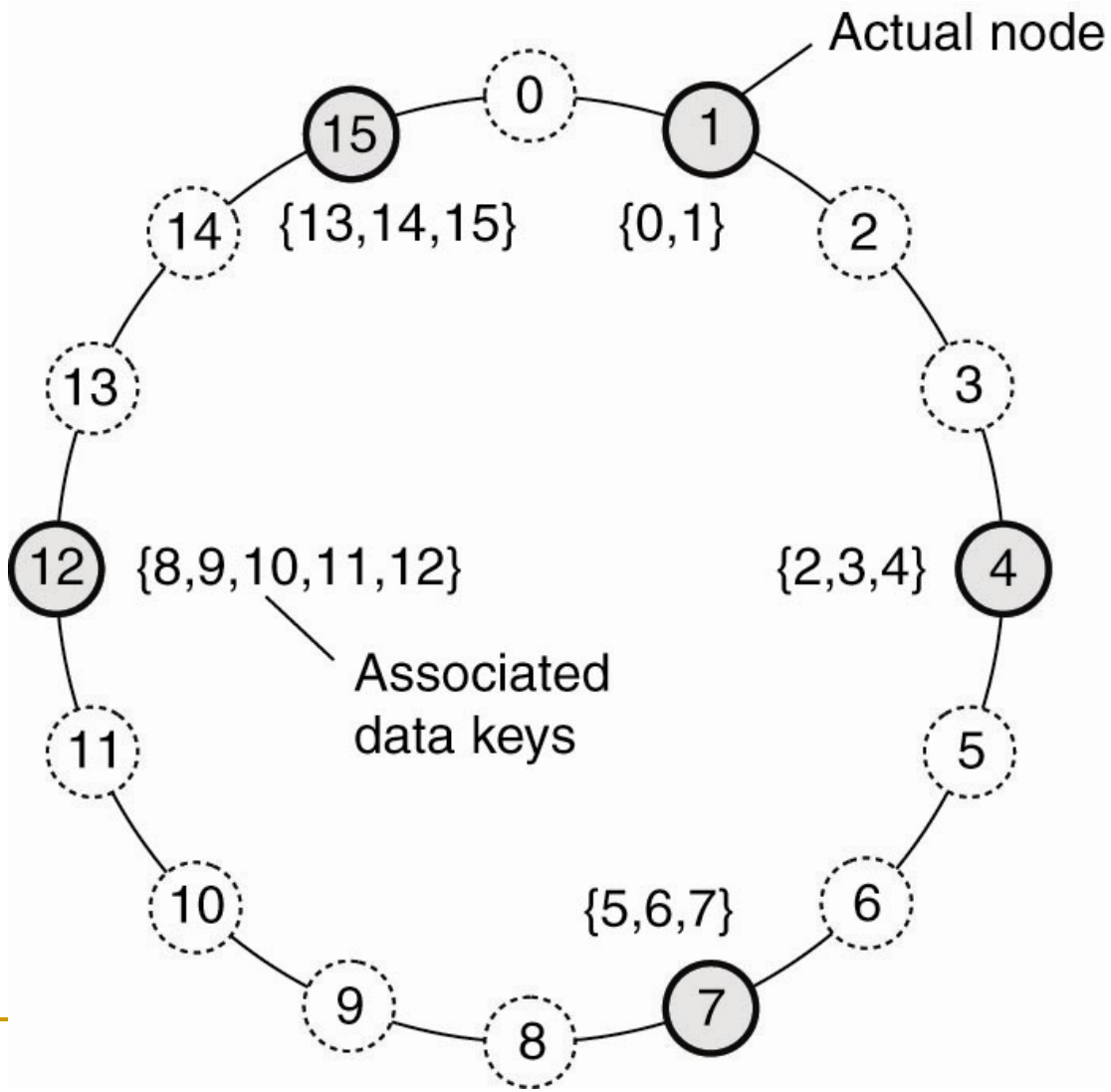
A distributed application based on peer processes



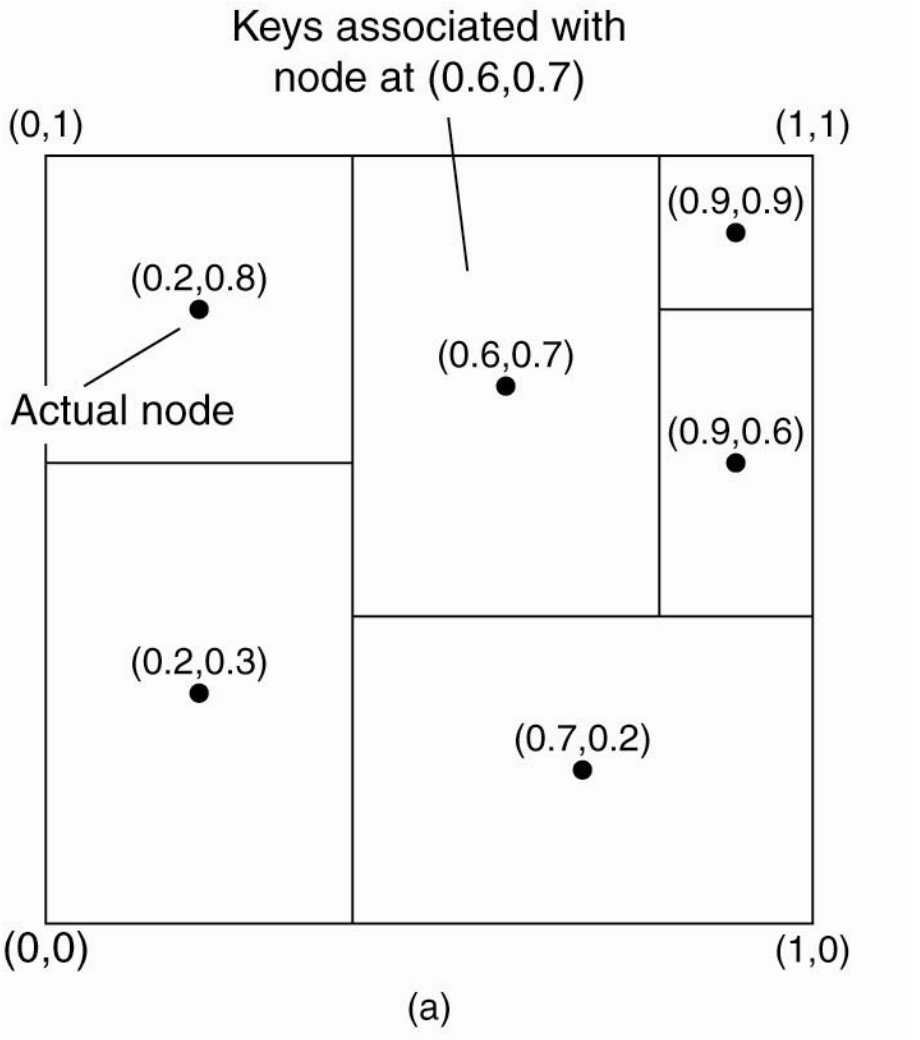
- Vertical distribution
- Horizontal distribution
- Peer-to-peer systems

- Overlay networks
- DHT…Distributed Hash Table
- Membership management

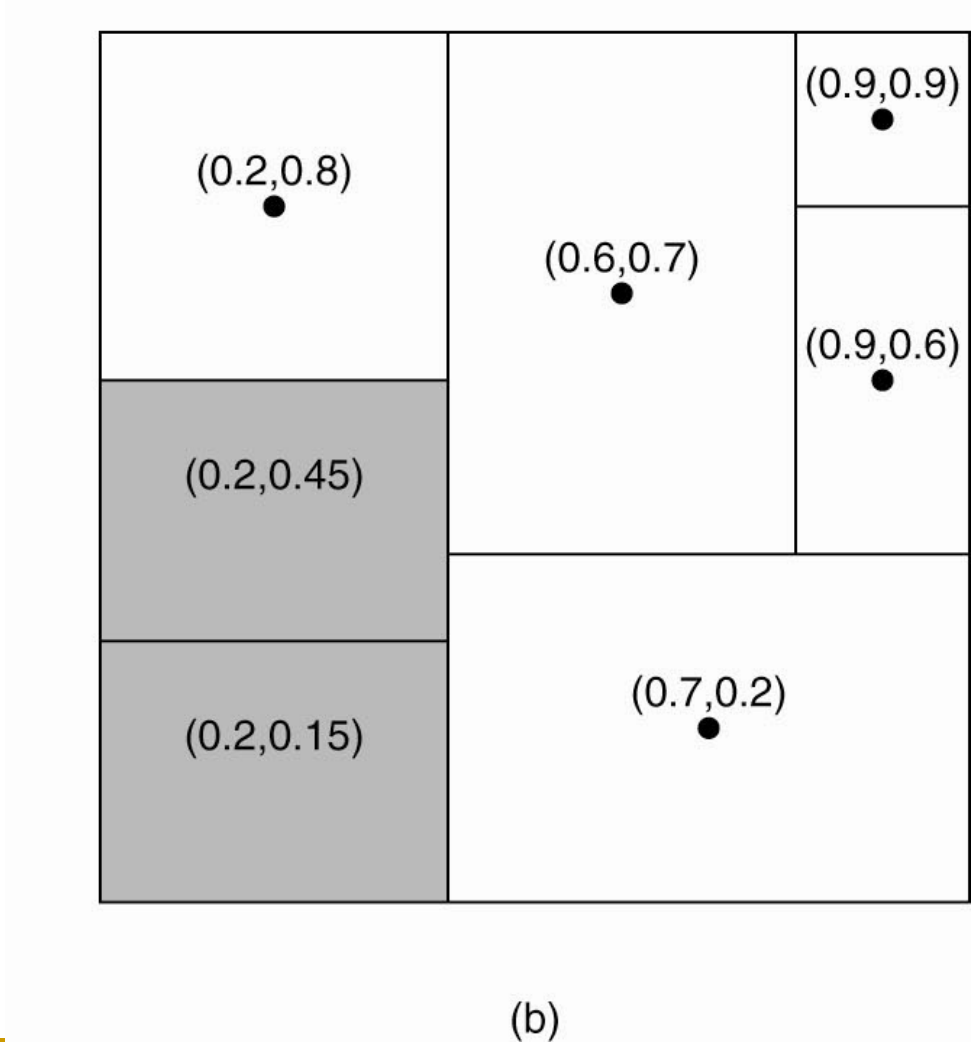
Structured Peer-to-Peer Architectures



Structured Peer-to-Peer Architectures



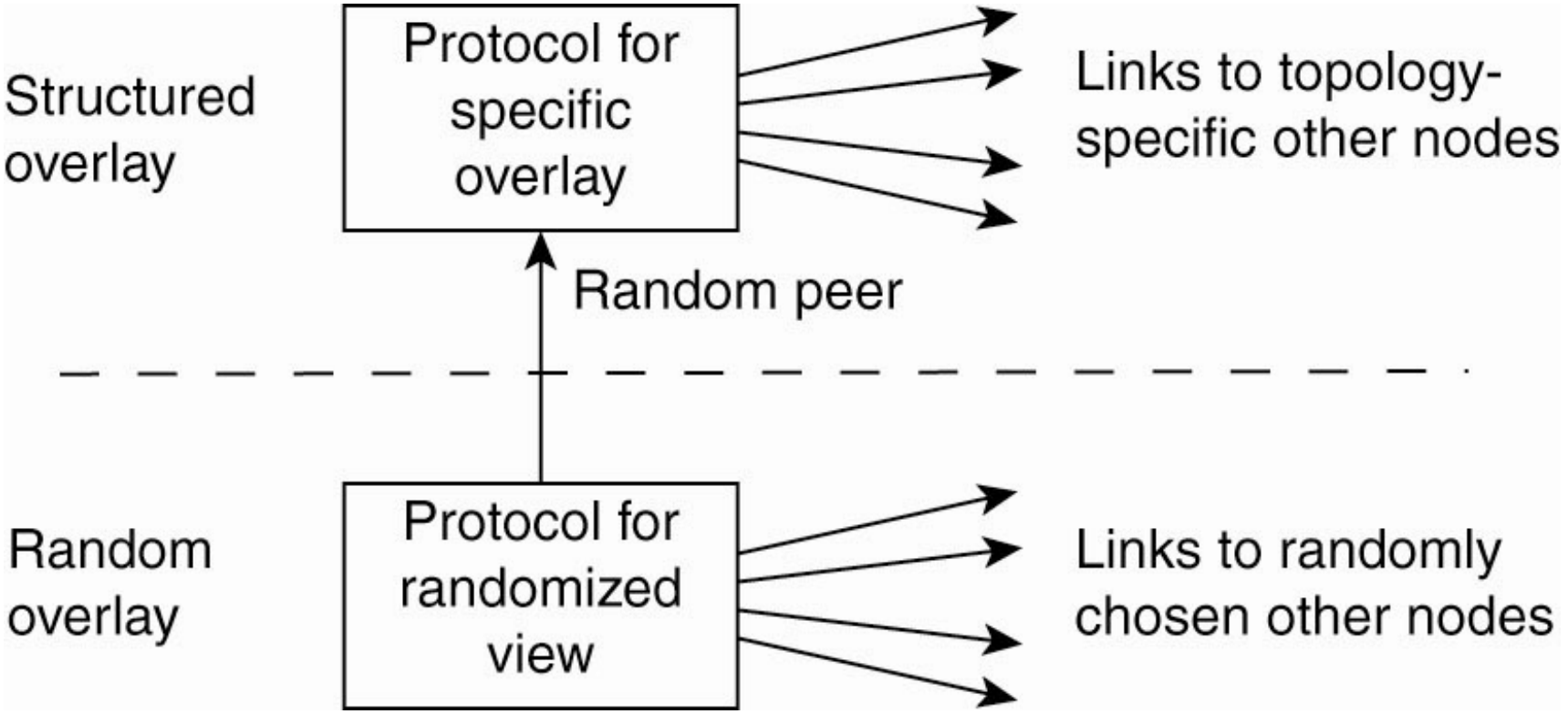
Structured Peer-to-Peer Architectures



Unstructured Peer-to-Peer Architectures

- Random graph
- Flooding

Topology Management of Overlay Networks



Unstructured Peer-to-Peer Architectures (1)

Actions by active thread (periodically repeated):

```
select a peer P from the current partial view;
if PUSH_MODE {
    mybuffer = [(MyAddress, 0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
} else {
    send trigger to P;
}
if PULL_MODE {
    receive P's buffer;
}
construct a new partial view from the current one and P's buffer;
increment the age of every entry in the new partial view;
```

(a)

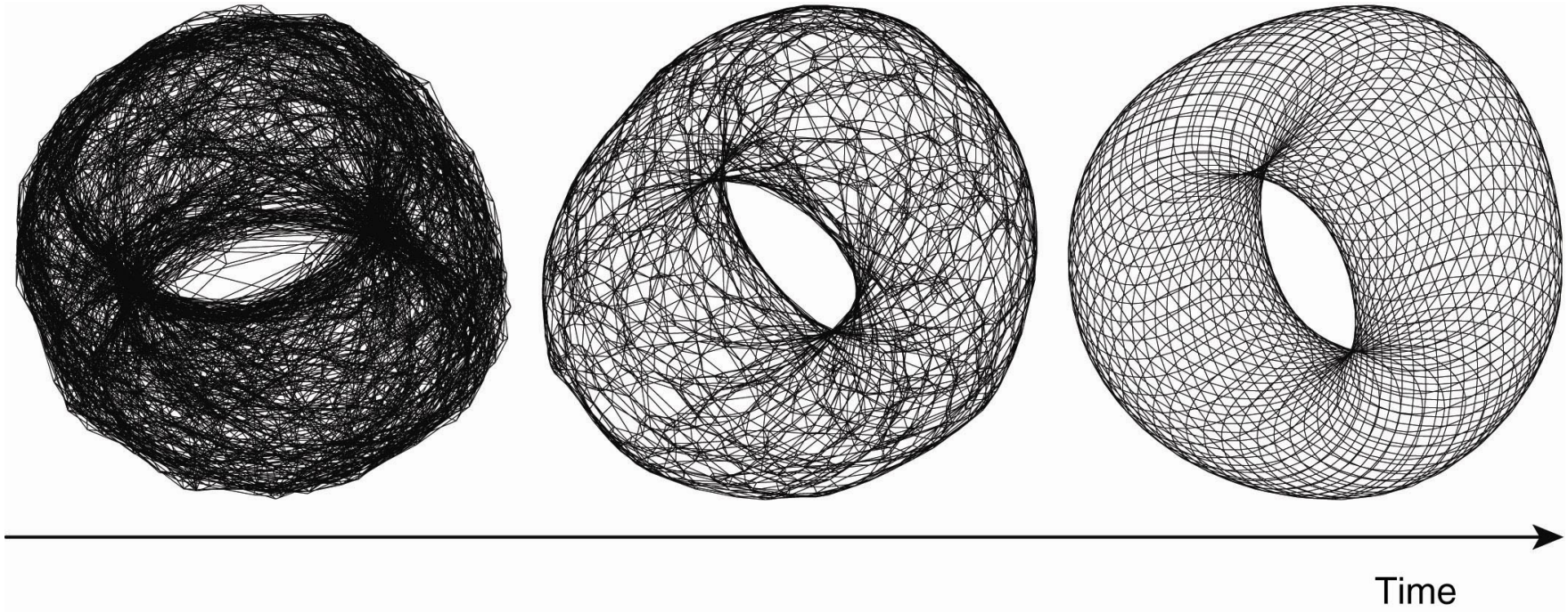
Unstructured Peer-to-Peer Architectures (2)

Actions by passive thread:

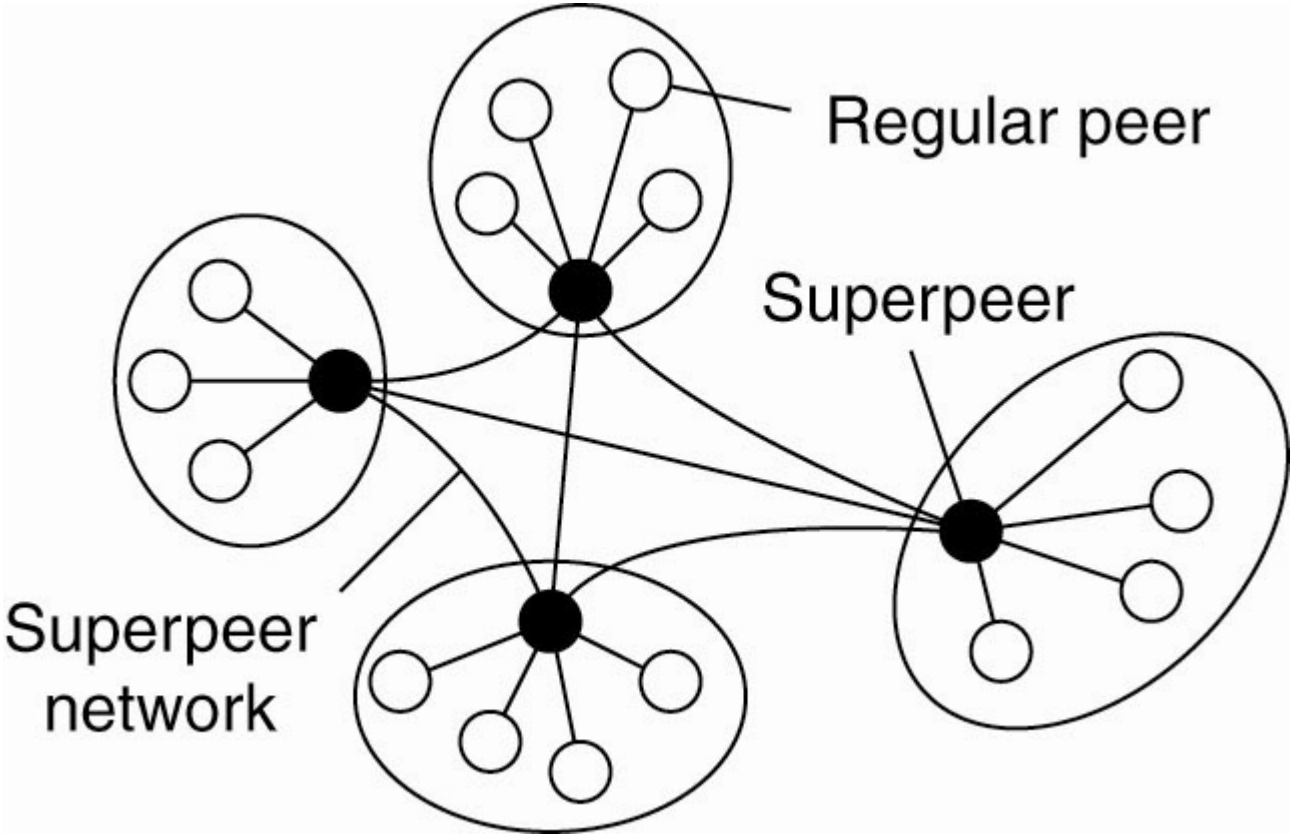
```
receive buffer from any process Q;
if PULL_MODE {
    mybuffer = [(MyAddress, 0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
}
construct a new partial view from the current one and P's buffer;
increment the age of every entry in the new partial view;
```

(b)

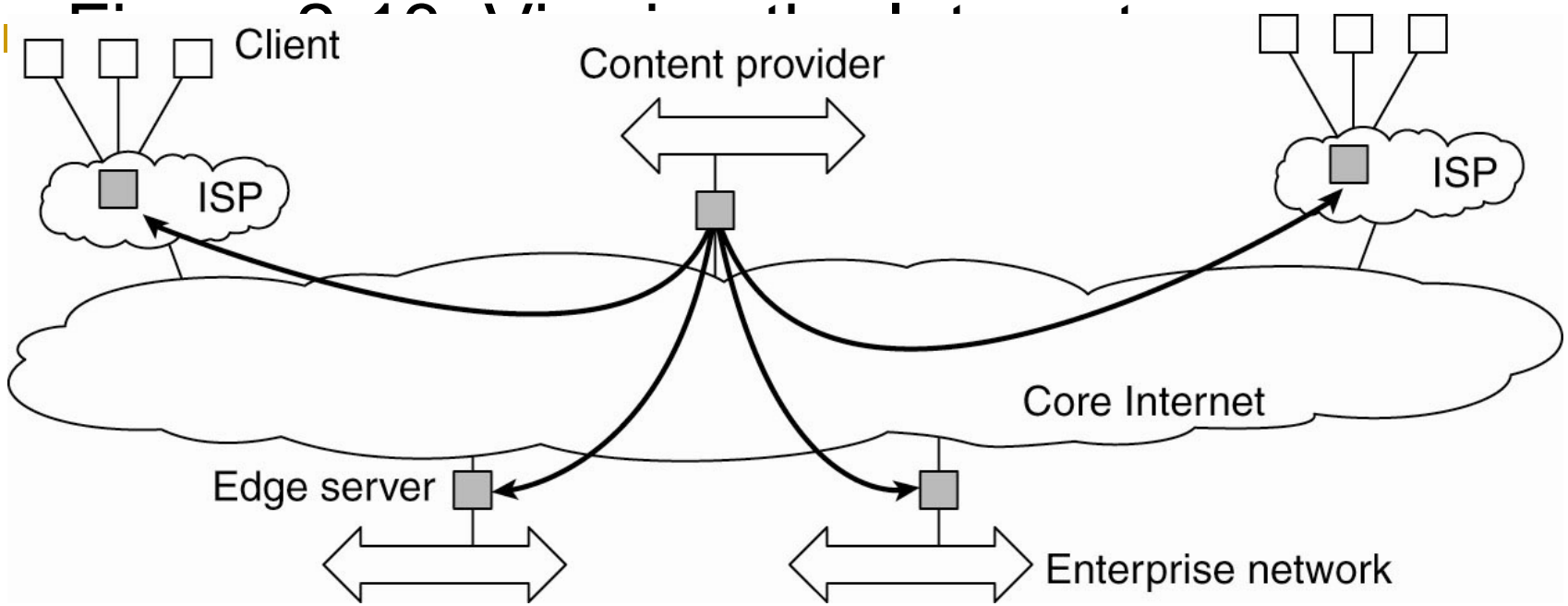
Topology Management of Overlay Networks



Superpeers



Edge-Server Systems

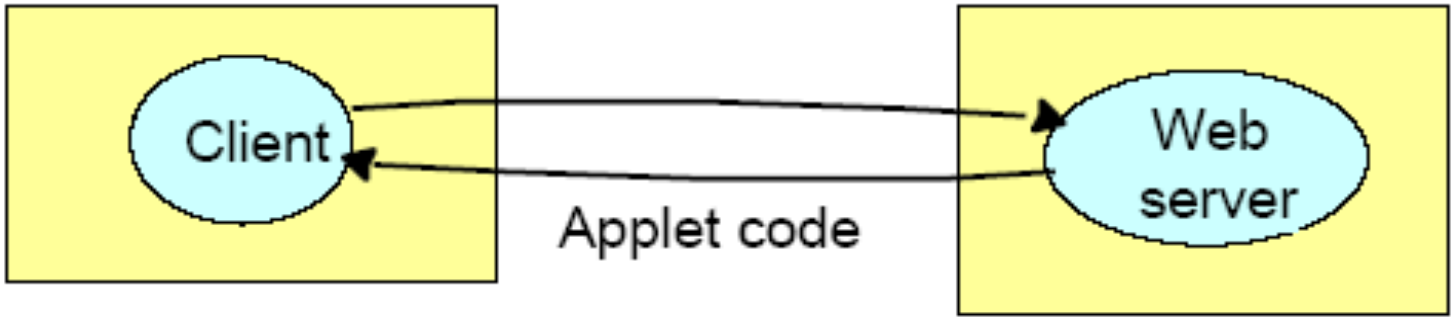


Client server and mobility

- Mobile code
  - downloaded from server, runs on locally
  - e.g. web applets

Web applets

Client requests results, applet code is downloaded:



Client interacts with the applet:

...mobility...

- Mobile agent (code + data + state)
  - travels from computer to another
  - collects information, returning to origin.

- Mobile devices forming “spontaneous networks”

Spontaneous networks ...

- Easy connection to a LAN
- Easy integration with local services
- Limited connectivity
- Security and privacy
- Discovery service=registration+lookup

Distributed Systems:

Models for Distributed Systems:
Fundamentals Models

Fundamental models

- Interaction model ... it reflects communication delays and limited accuracy due to local timing
- Failure model ...at processing and communication level
- Security model ...

Interaction model

It deals with communication and coordination

Distributed Algorithms ...

No matter how communication channels are realized ....

- Latency
- Bandwidth
- Jitter

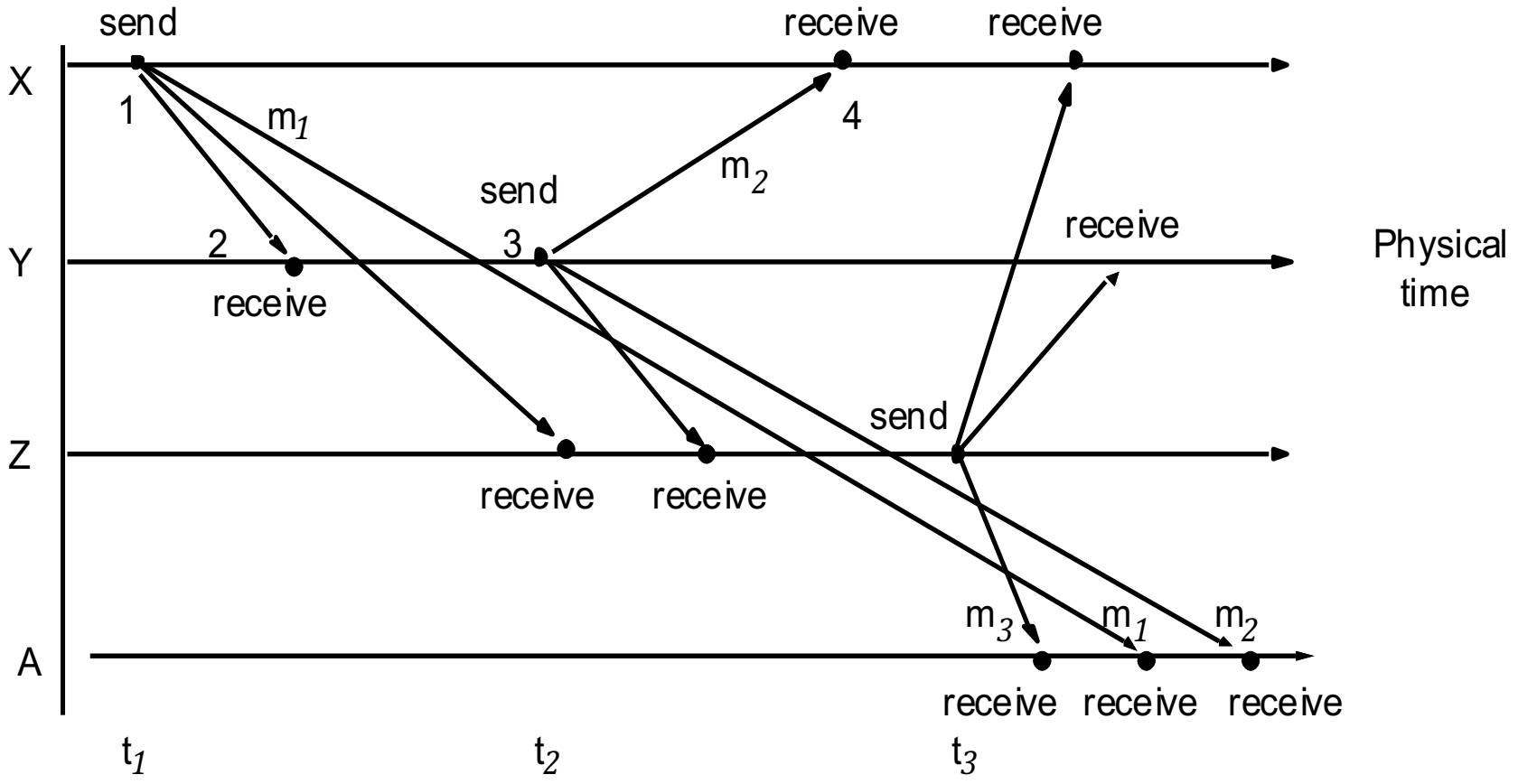
Clock drift

- Synchronous distributed systems
  - Clock drift
  - Execution step
  - Message transmission time
- Asynchronous distributed systems
- Agreement problem

…Event ordering…

- Occurred before
- Occurred after
- Concurrent
- A message is received after it is sent and replies are sent after receiving messages

…real-time ordering of events…



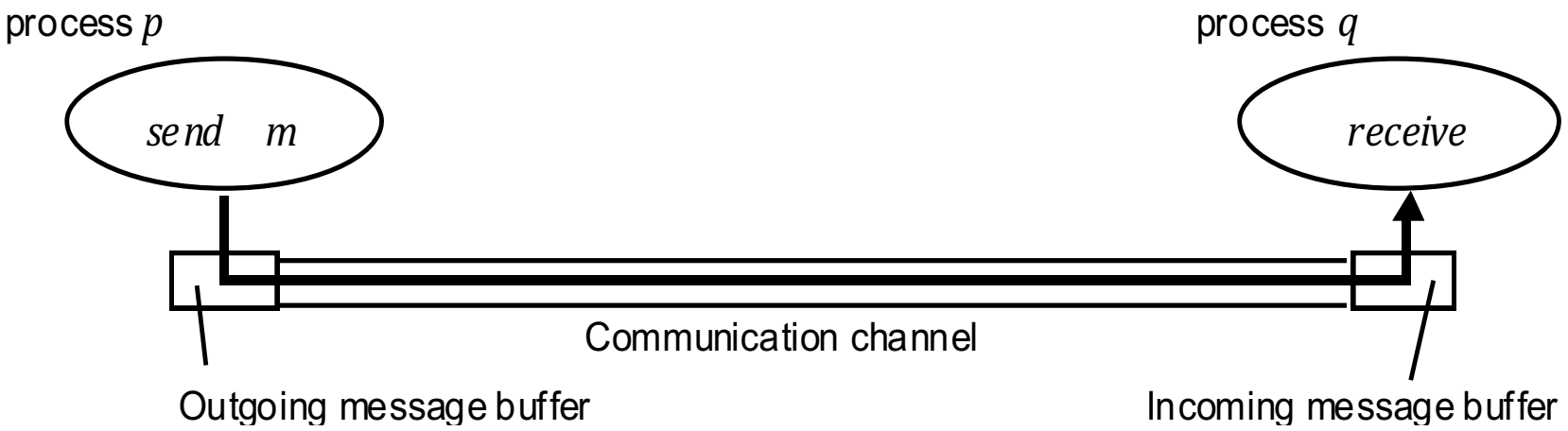
Failure model

- To provide an understanding of the effects of failures .... DSs expected to continue if failure has occurred:

- Omission failures…fail to perform actions
  - Process omission failures ... A stop, A crash ... A clean crash ...

- Fail-stop
- Crash
- Arbitrary

…processes and channels…



- Communication omission failures
  - Send-omission
  - Receive-omission
  - Channel-omission: the communication channel does not transport a message from the out buffer to the in buffer

Failure issues: …types of failures…

- benign failures (omission, stopping, timing/performance)
- arbitrary (called Byzantine)
  - corrupt message, wrong method called, wrong result

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.

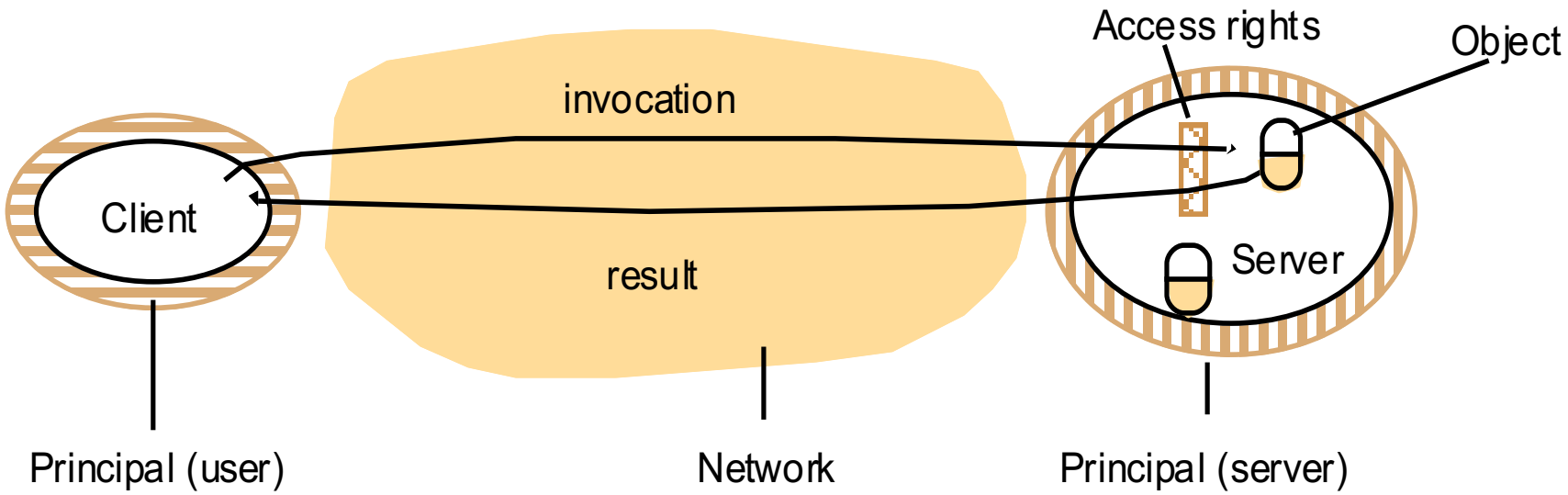
…timing failures…

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.

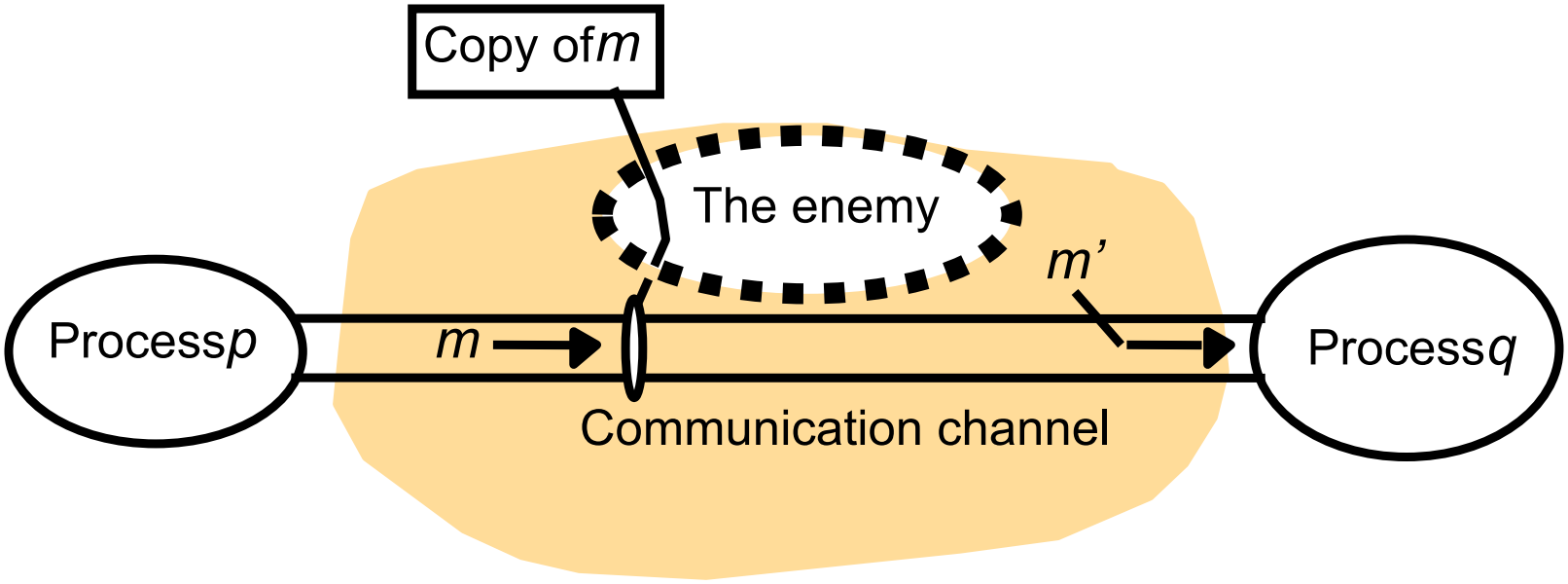
Security models

- Protecting objects ... Access rights
- Securing processes and their interaction
  - Enemy (adversary) modeling

Objects and principals

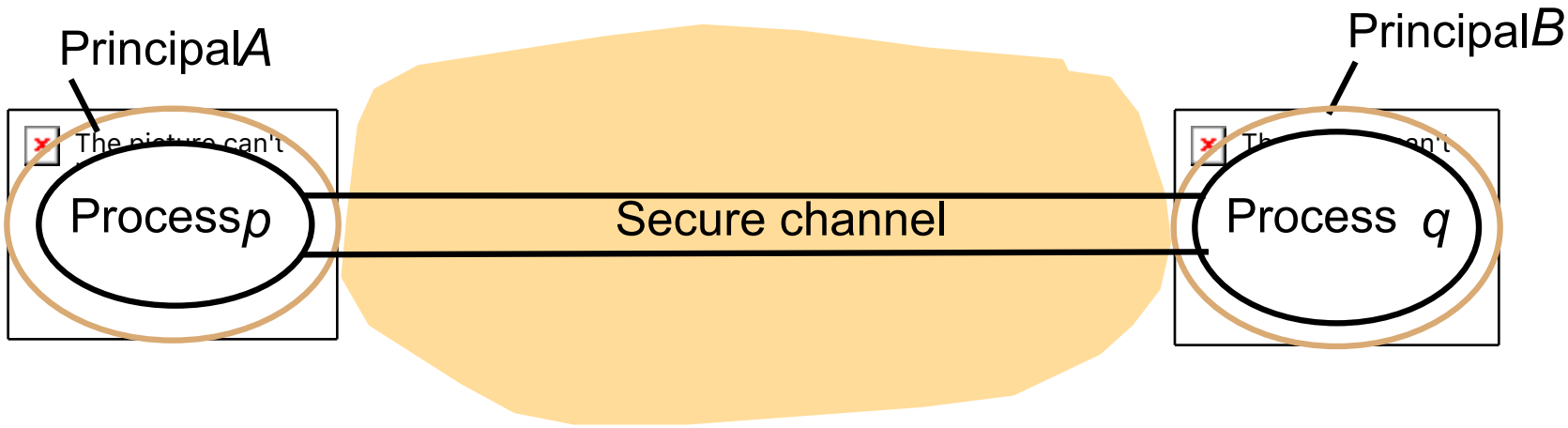


…the enemy…


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

- Cryptography and shared secrets
- Authentication
- Secure channels

…secure channels…



…Distributed Systems…

end of lectures

References:

- A.S. Tanenbaum, M. Van Steen, “Distributed Systems: Principles and Paradigm”, Prentice- Hall, II edition, 2007, Chap. 2 “Architectures”
- George Coulouris, Jean Dollimore, Tim Kindberg, “Distributed Systems: concepts and design”, fourth edition, Addison-Wesley, 2005, Chap. 2 “System models”