

06-06798 Distributed Systems

Lecture 2: Architectural Models

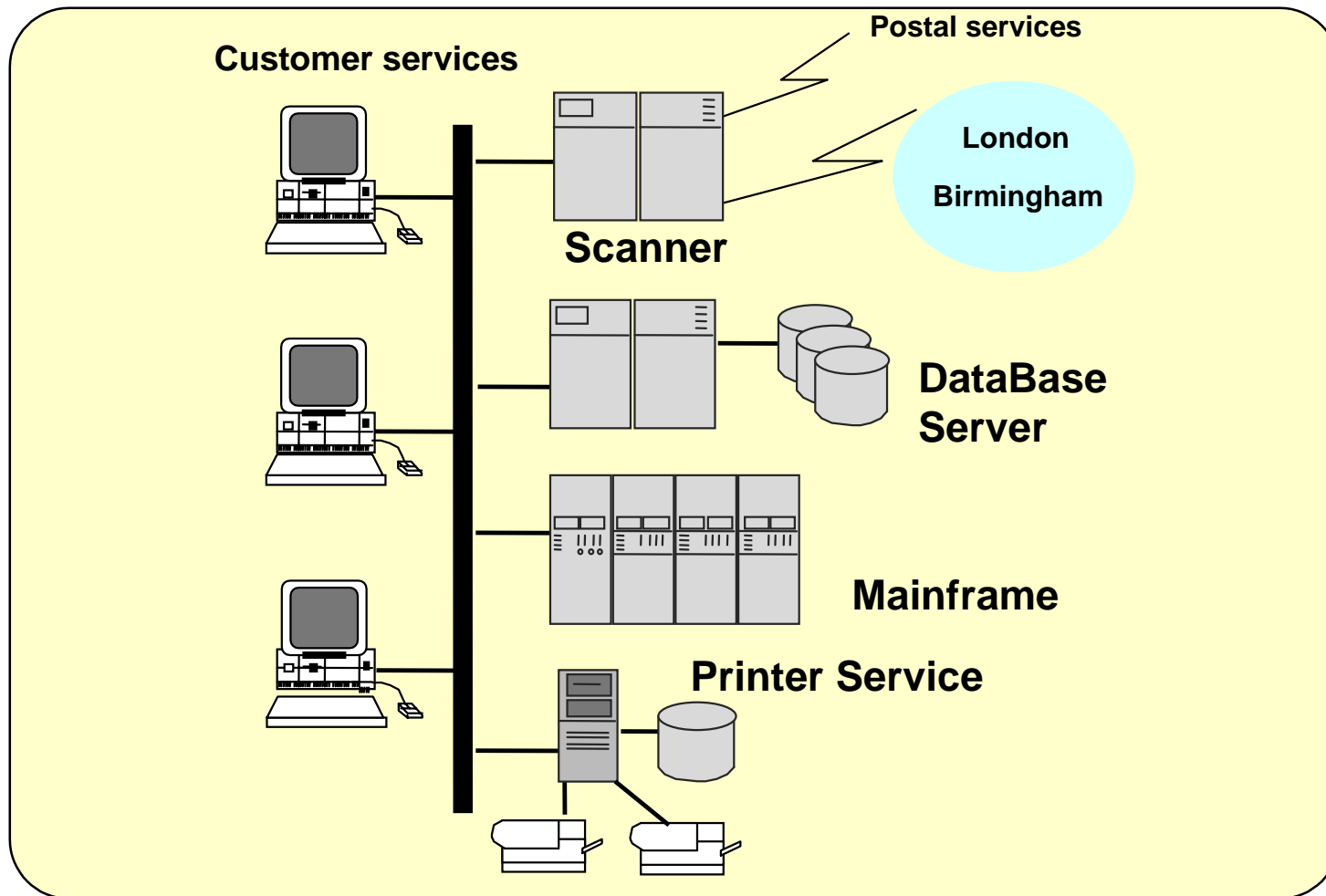
Overview

- System architectures
- Software layers
- Architectural models
 - client-server, peer processes,...
 - mobile code, agents,...
- Design requirements
 - user expectations of the system

Example: Paperless Office

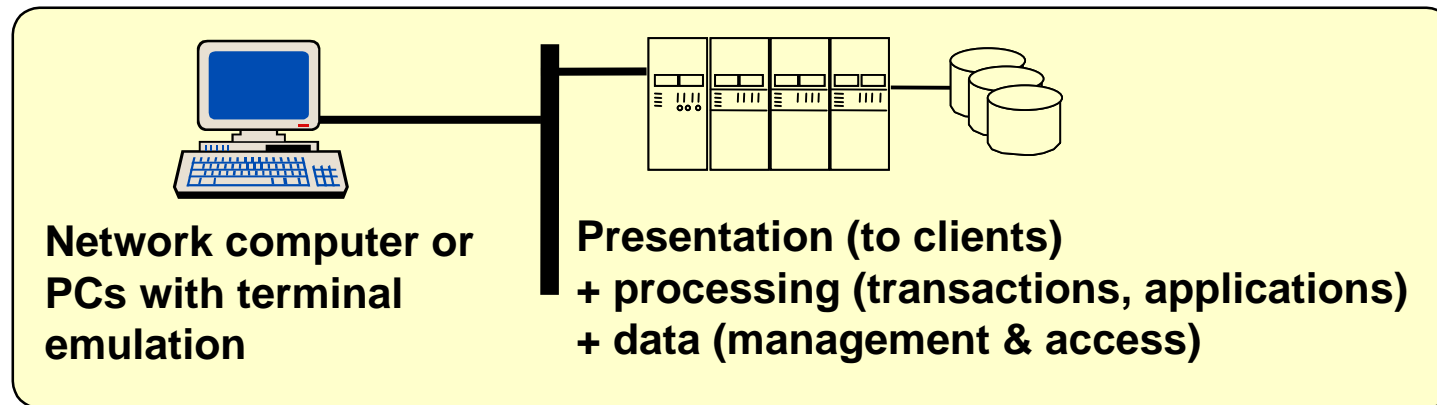
- Requirements
 - input and storage of scanned documents
 - viewing/printing of documents on demand
 - networking for resource sharing and communications
 - accounting and data analysis
- Required properties
 - no loss/corruption/unauthorised access of data
 - fast response
 - should grow as the business expands

Distributed Design

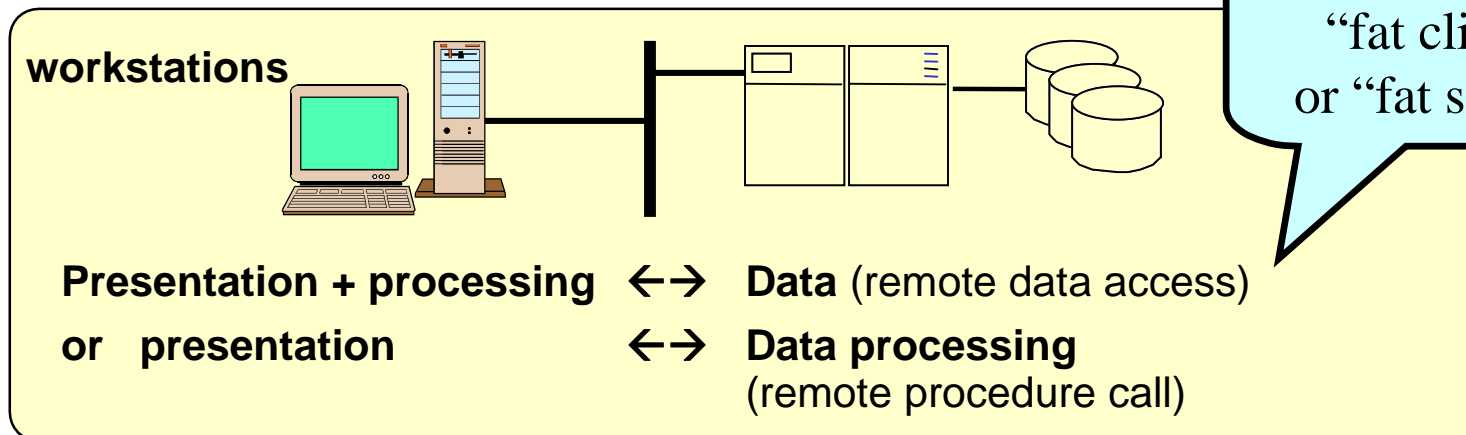


Client Server Systems

One Tier Architecture



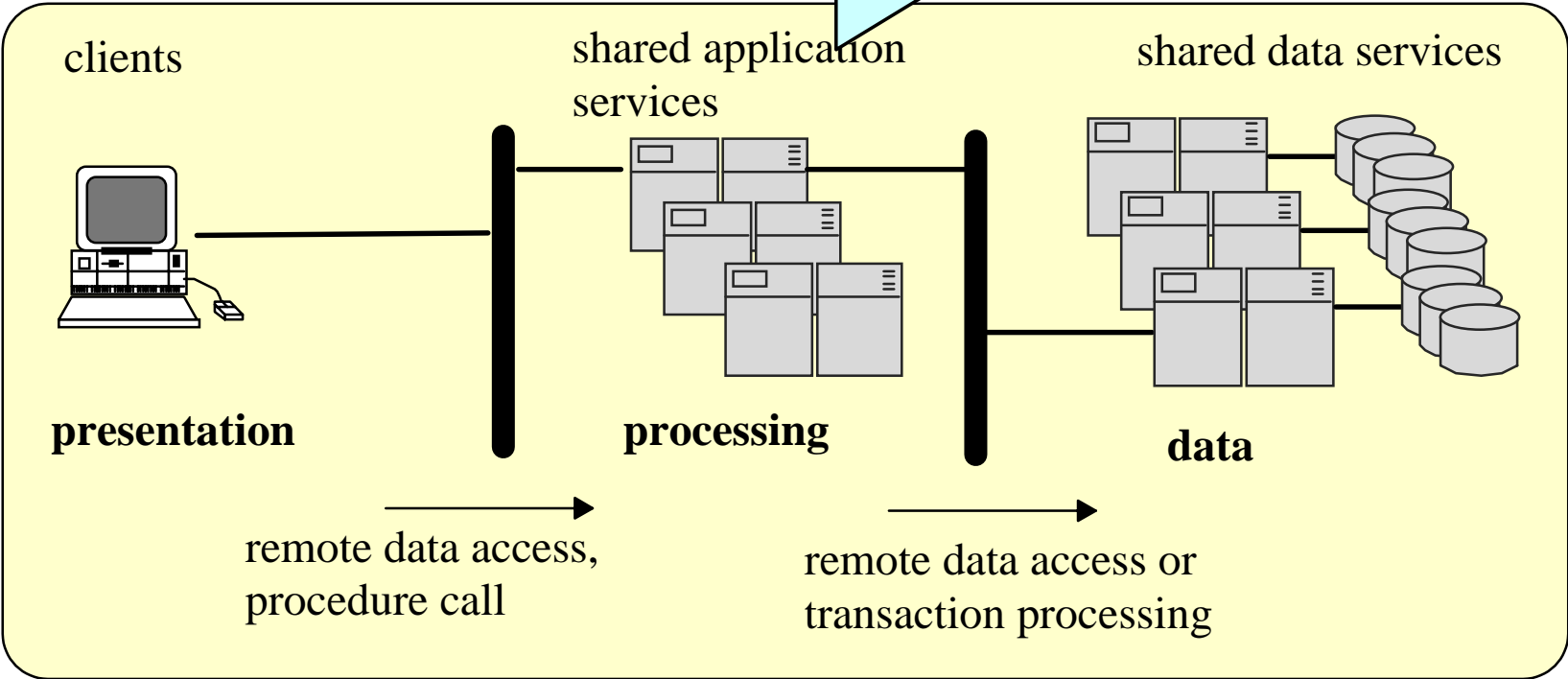
Two Tier Architecture



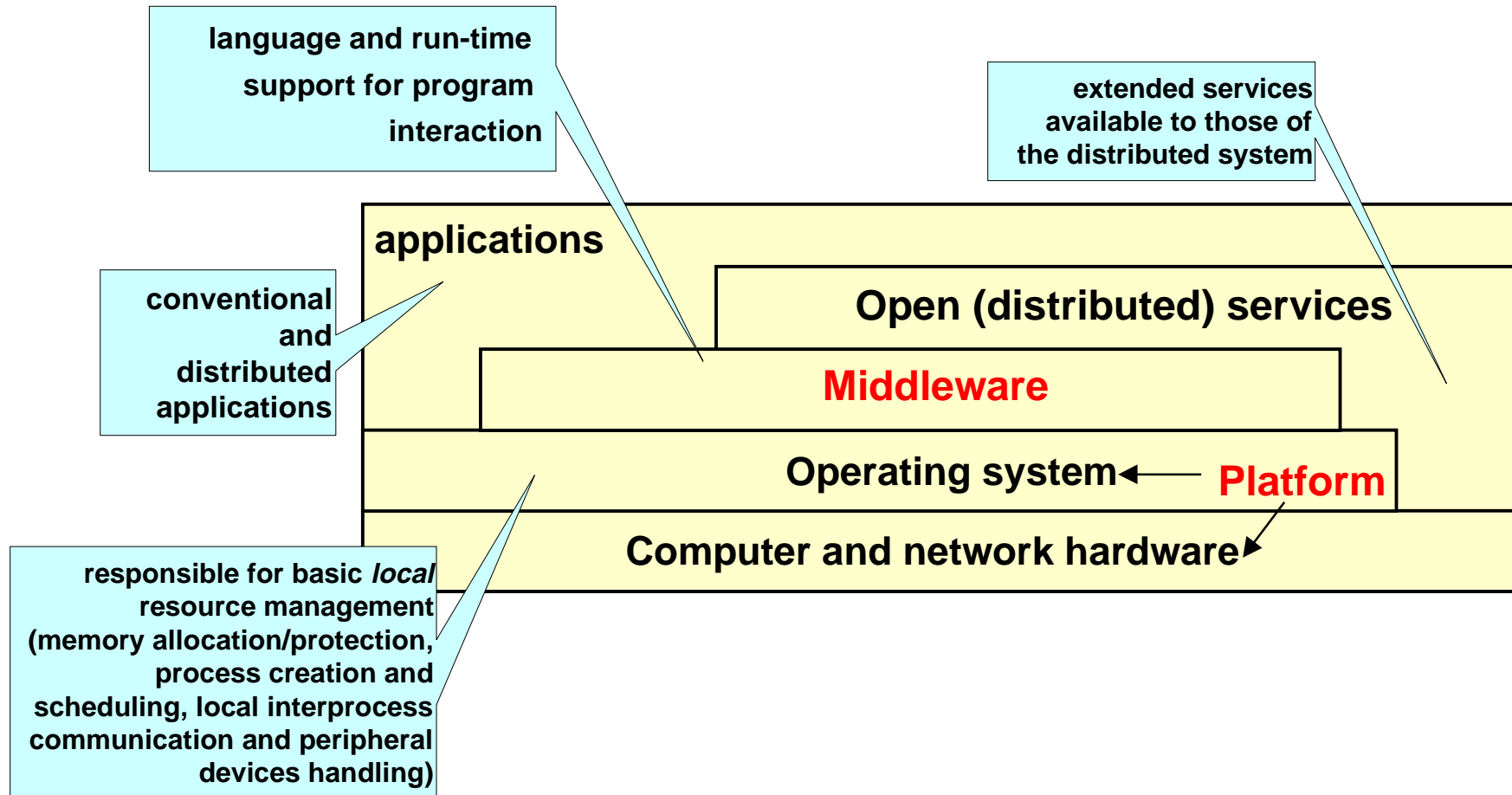
Client Server ctd

Three Tier Architecture

Two tier is satisfactory for simple client-server applications, but for more demanding transaction processing applications*....



Software Layers



Software layers

- Service **layers**
- Higher-level **access services** at lower layers
- Services can be located on different computers
- Process types:
 - **server** processes
 - **client** processes
 - **peer** processes

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, ...
 - products: CORBA, DCOM

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]

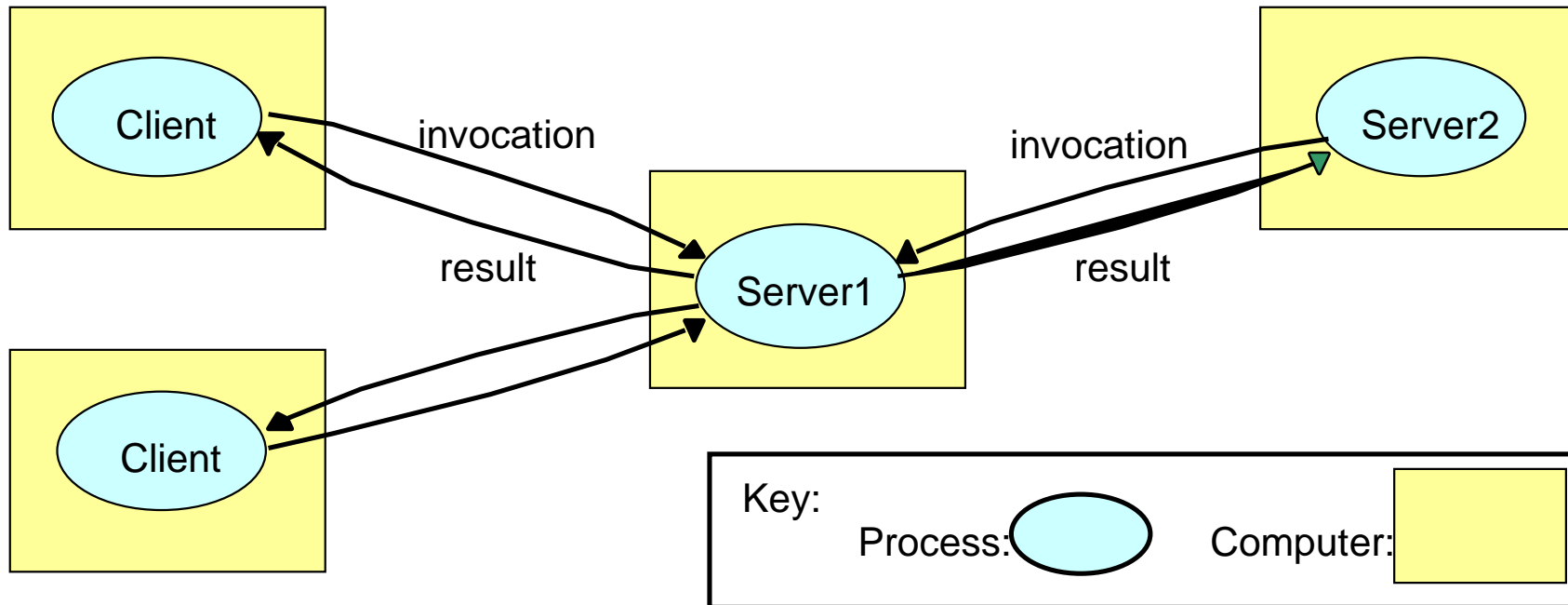
Architectural models

- 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

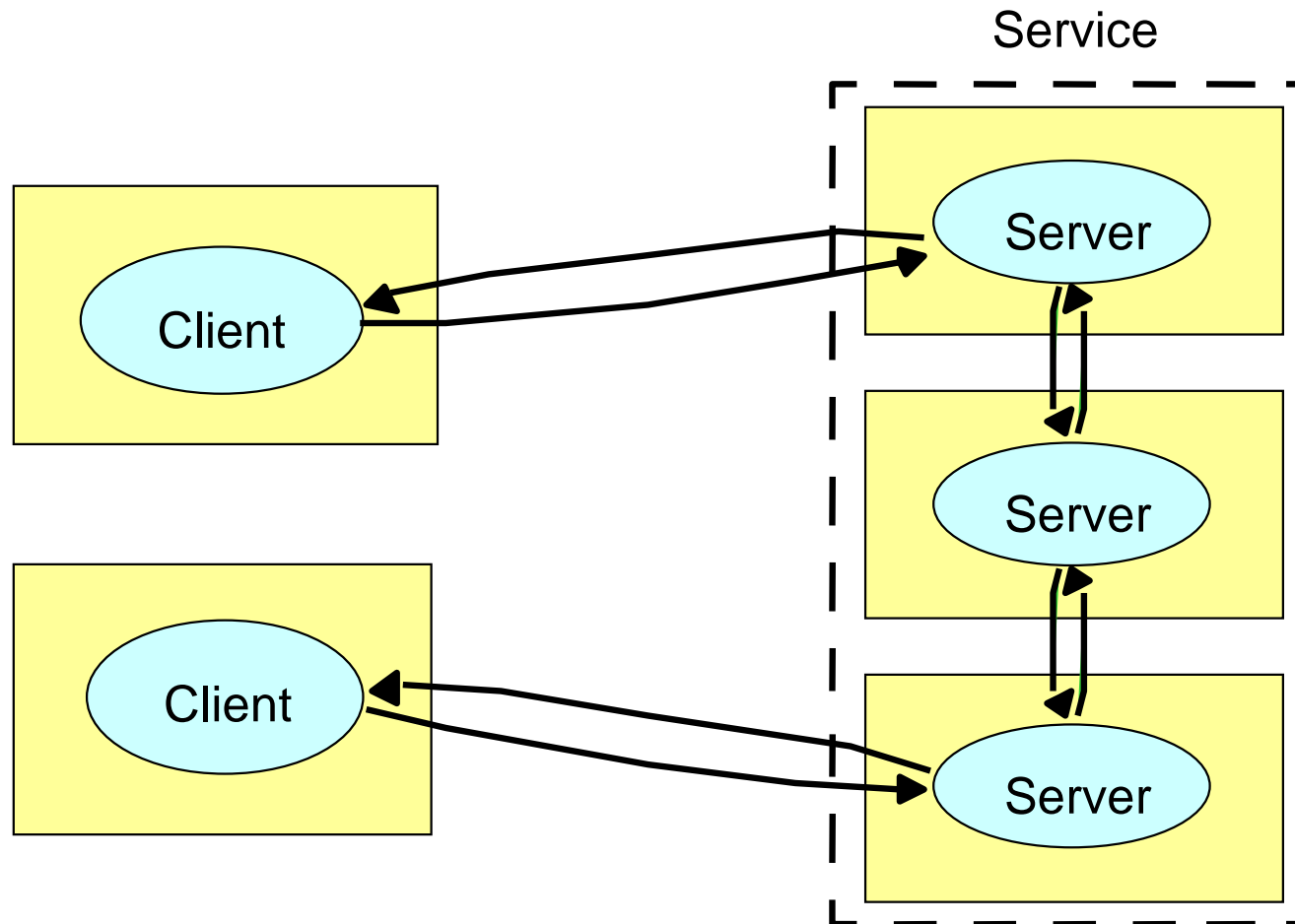
- **Client-server**
 - first and most commonly used
- **Multiple servers**
 - to improve performance and reliability
 - e.g. search engines (1000's of computers)
- **Proxy servers**
 - to reduce load on network, provide access through firewall
- **Peer processes**
 - when faster interactive response needed

Client server



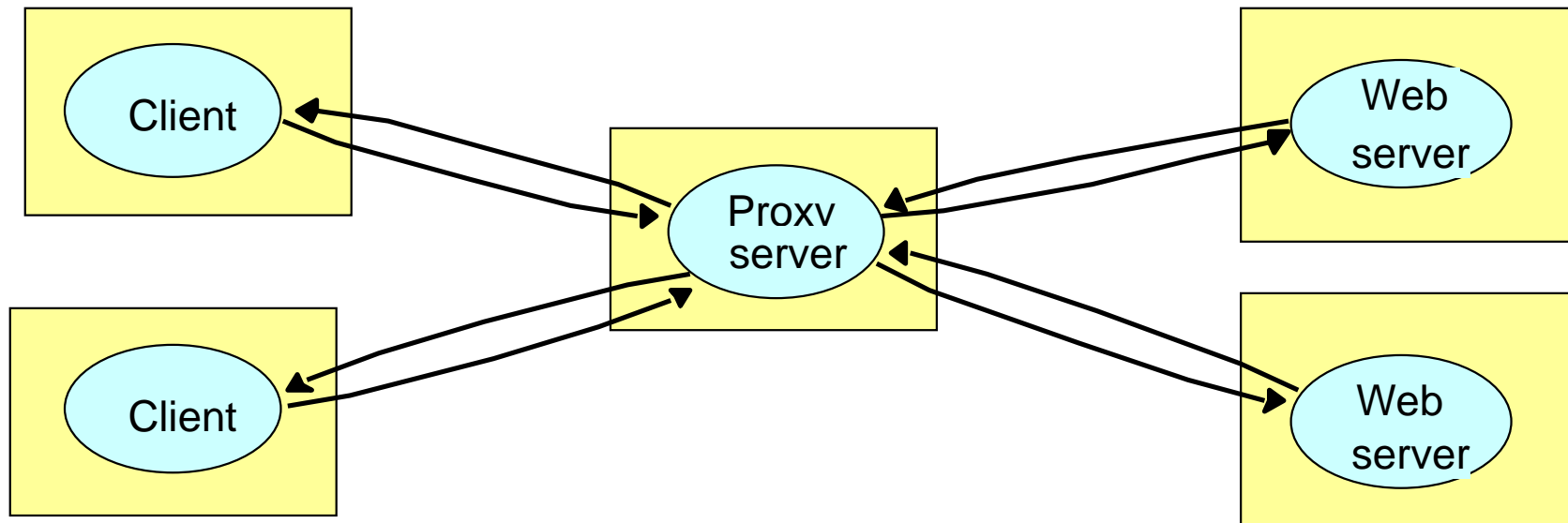
Server1 acts as **client** for Server2

Multiple servers



Servers may **interact**

Proxy servers

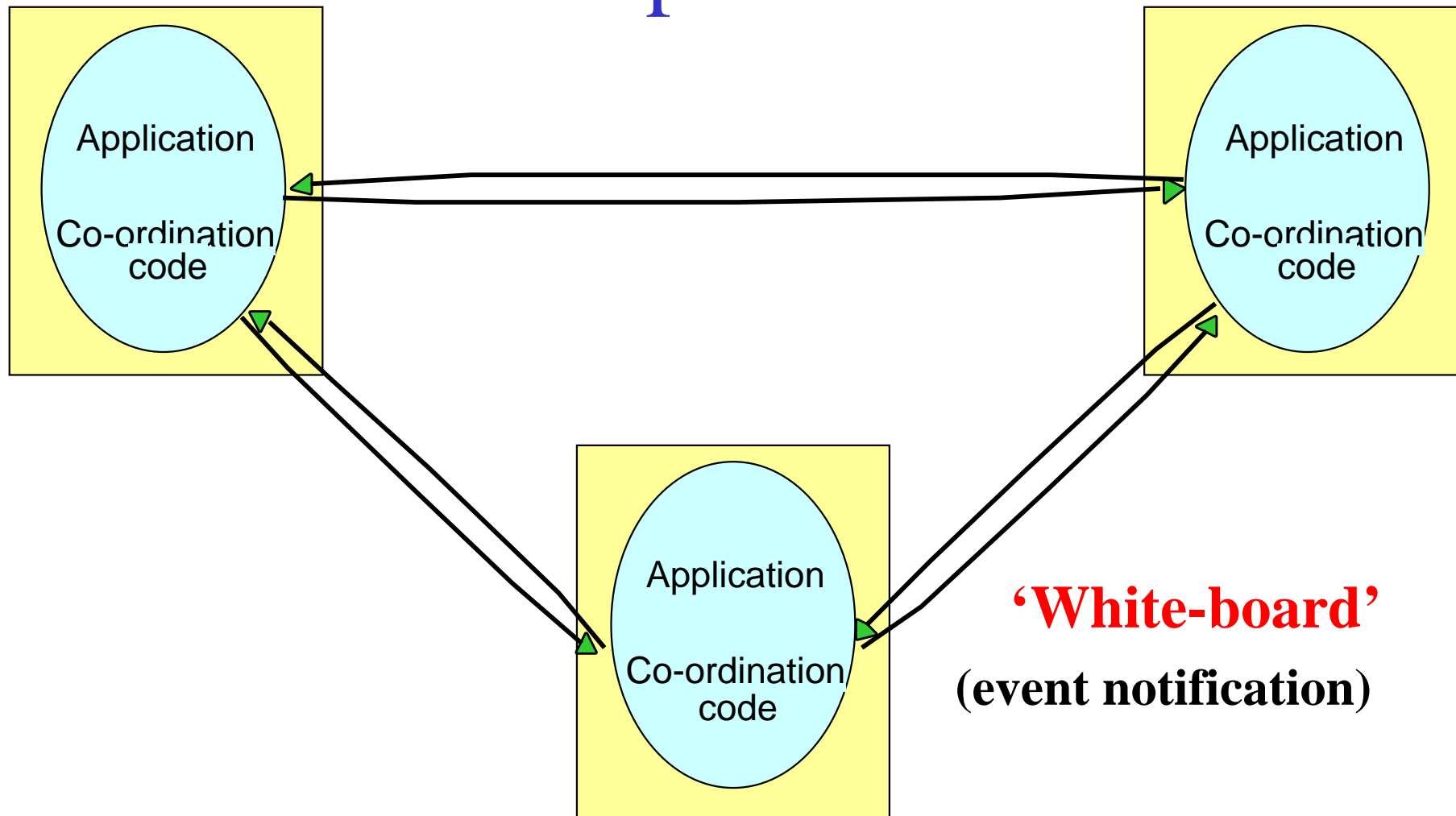


intranet

firewall

outside world

Peer processes



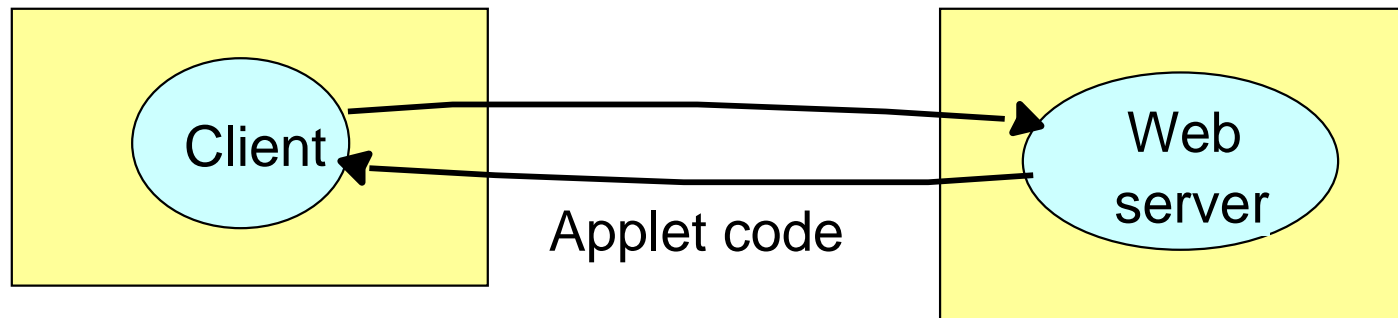
Client server and mobility

- Mobile **code**
 - downloaded from server, runs on locally
 - e.g. web applets
- Mobile **agent** (code + data)
 - travels from computer to another
 - collects information, returning to origin

Beware! Security risks

Web applets

Client **requests** results, applet code is **downloaded**:



Client **interacts** with the applet:



Design Requirements for DSs

Judging how good the architecture is...

- **Performance**
 - how fast will it respond?
- **Quality of Service**
 - are video frames and sound synchronised?
- **Dependability**
 - does it work correctly?

Performance

- **Responsiveness**
 - fast interactive response delayed by remote requests
 - use of caching, replication
- **Throughput**
 - dependent on speed of server and data transfer
- **Load balancing**
 - use of applets, multiple servers

Quality of Service (QoS)

Non-functional properties experienced by users:

- **Deadline** properties
 - **hard** deadlines (must be met within T time units)
 - **soft** deadlines (there is a 90% chance that the video frame will be delivered within T time units)
 - multimedia traffic, video/sound synchronisation
 - depend on availability of sufficient resources
- **Adaptability**
 - ability to adapt to changing system configuration

Dependability

- **Correctness**
 - correct behaviour wrt specification
 - e.g. use of verification
- **Fault-tolerance**
 - ability to tolerate/recover from faults
 - e.g. use of redundancy
- **Security**
 - ability to withstand malicious attack
 - e.g. use of encryption, etc

Summary

- Choose between **one tier**, **two tier**, ...
 - simple versus complex transaction processing
- **Client-server** architecture most common
 - used for WWW, email, ftp, Internet services, etc
- but can lead to **bottlenecks**
 - multiple servers for fast response (e.g. Google search engine based on 6,000 Linux PCs)
 - proxy servers used to limit load (e.g. through firewall)
- Expected to meet requirements of **Performance**, **QoS** and **Dependability**