

# Distributed Systems

## COMP 212

### Lecture 16

Othon Michail

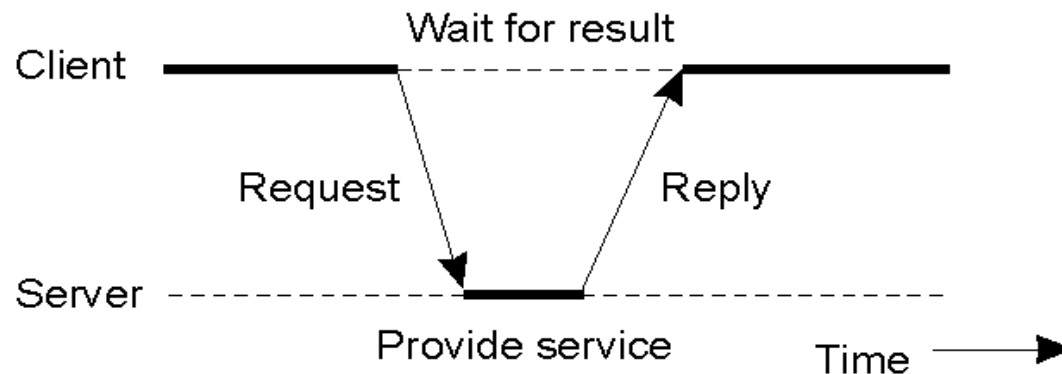
# Architectures & Processes

# Client-Server Model

# Clients and Servers

Processes are divided into

- **Servers**: implementing a specific service
  - e.g., file system, database
- **Clients**: requesting a service from a server by sending it a request and subsequent waiting for the server's reply
- Distributed across different machines
- Follow a **request-reply** behaviour



# What's a Server?

**Definition:** “A process that implements a specific service **on behalf of** a collection of clients”.

Typically, servers are organised to do one of two things:

1. Wait
2. Service

- ... wait ... service ... wait ... service ... wait ...

# Servers: Iterative and Concurrent

- **Iterative**: server handles request, then returns results to the client; any new client requests **must wait** for previous request to complete (also useful to think of this type of server as **sequential**).
- **Concurrent**: server does not handle the request itself; a separate **thread** or **sub-process** handles the request and returns any results to the client; the server is then free to immediately service the next client (i.e., there's no waiting, as service requests are processed **in parallel**).

# Server “States”

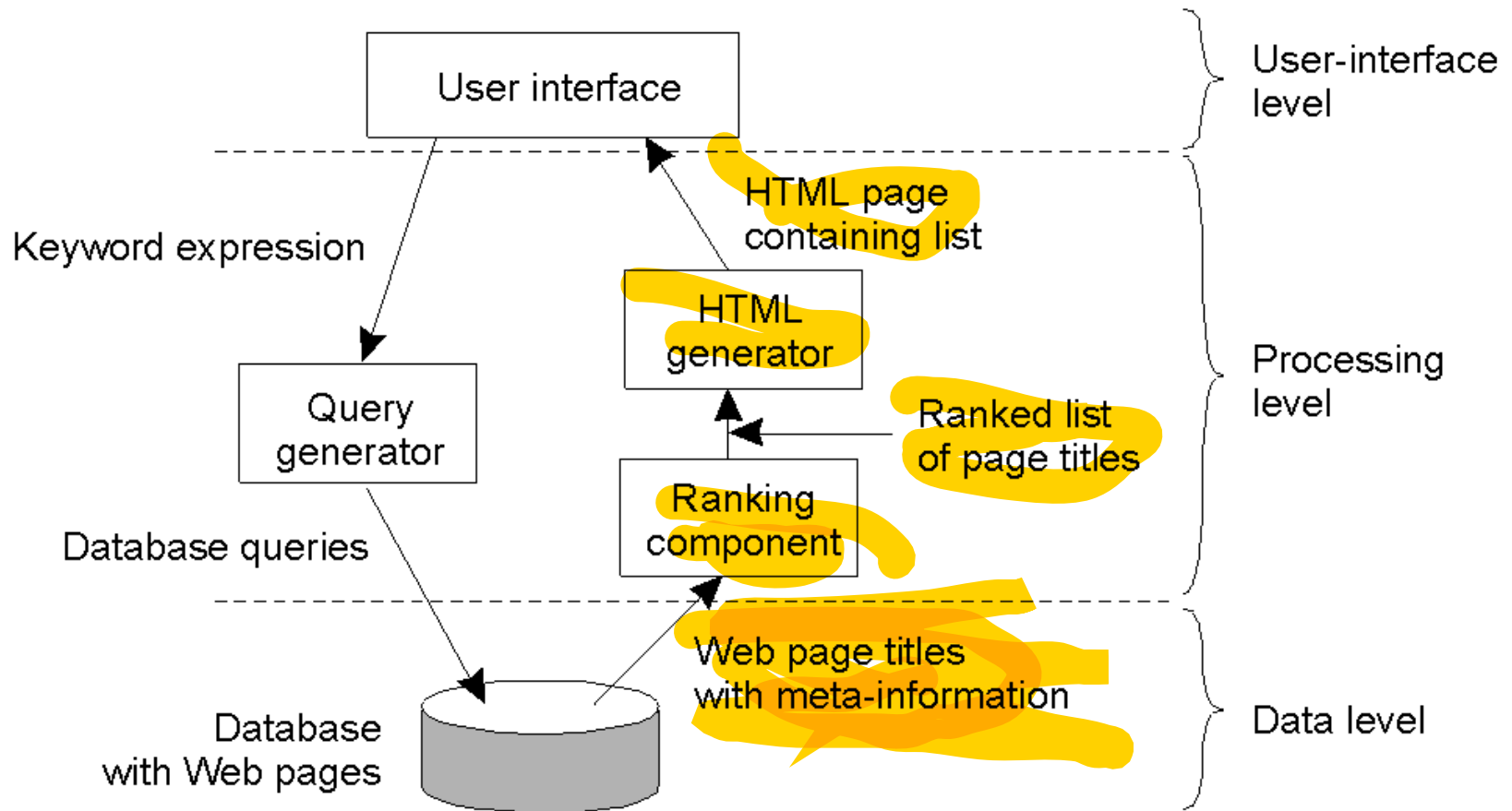
- **Stateless server** – does not keep information on the state of its clients
  - Classic example: the web
  - This type of server is **easy** to implement
- **Stateful server** – generally maintains persistent information on its clients
  - Example: file server that allows clients to keep local copies of files
    - Server must maintain (*client, file*) entries to keep track of the distribution and access rights
  - These are more **difficult** to implement
  - Improves performance but if e.g. crashes has to recover its entire state just before the crash
- **Cookies?**

# Application Layering

- A **three-layered (or level) view** from database applications
  1. **User-interface level**
    - All necessary to directly interface with the user
    - e.g., a webpage with forms
    - Clients typically implement this
  2. **Processing level**
    - Contains the core functionality of the application
  3. **Data level**
    - Contains the programs that maintain the actual data on which the applications operate
    - Servers typically implement this
    - Data may be **persistent** (i.e., maintained even when no client is operating on them)

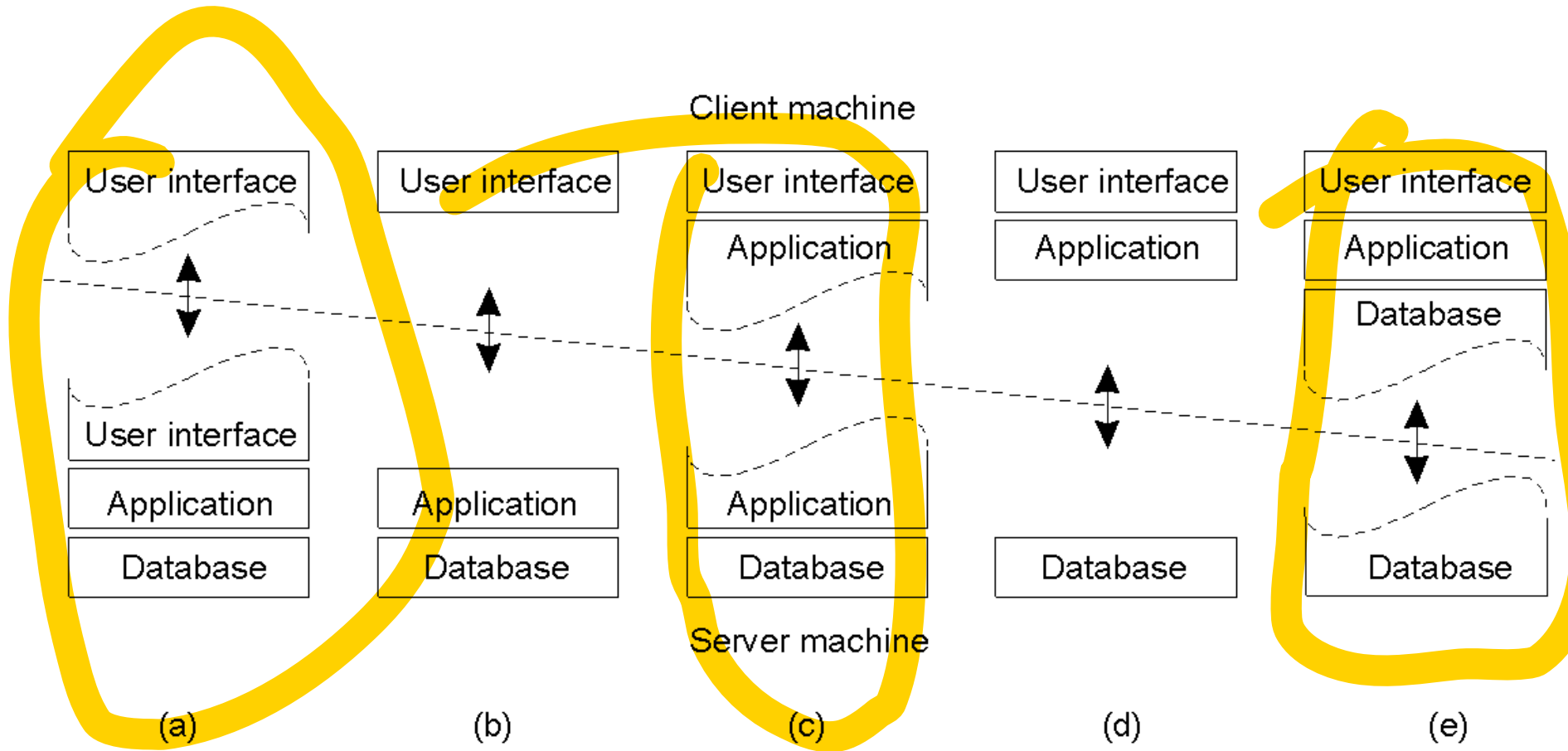


# A Three-layered Example



- The general organisation of an Internet search engine into **three different layers**
  - Often referred to as **tiers**

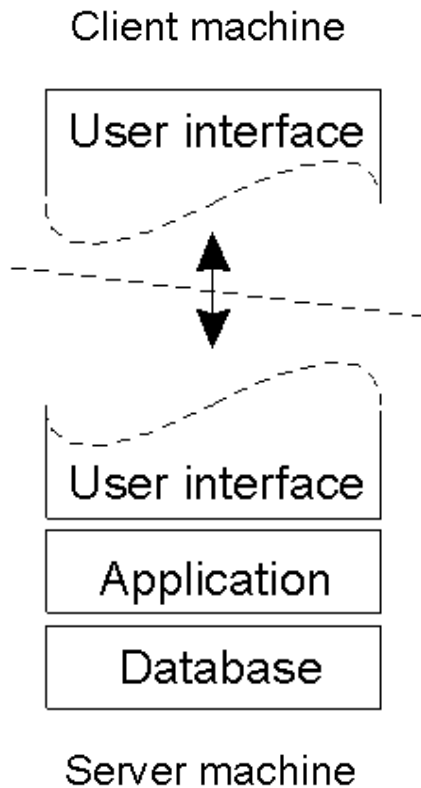
# Two-tiered Architectures



Alternative client-server organisations in the **two-tiered architecture**

- **2 kinds of machines:** clients and servers

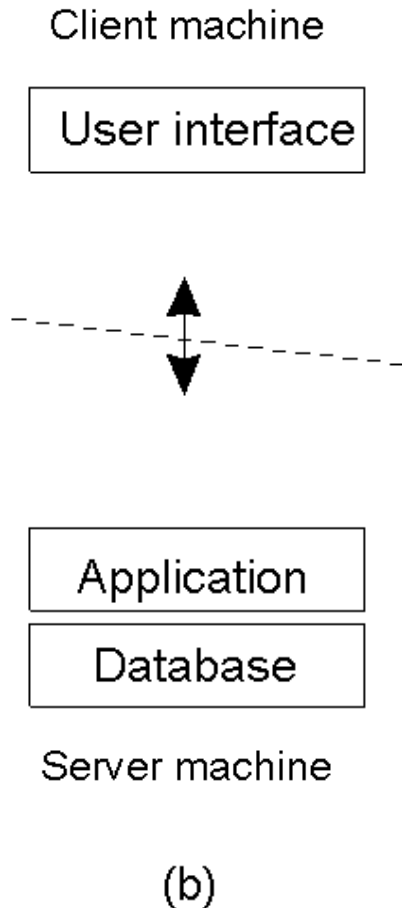
# Two-tiered Architectures



(a)

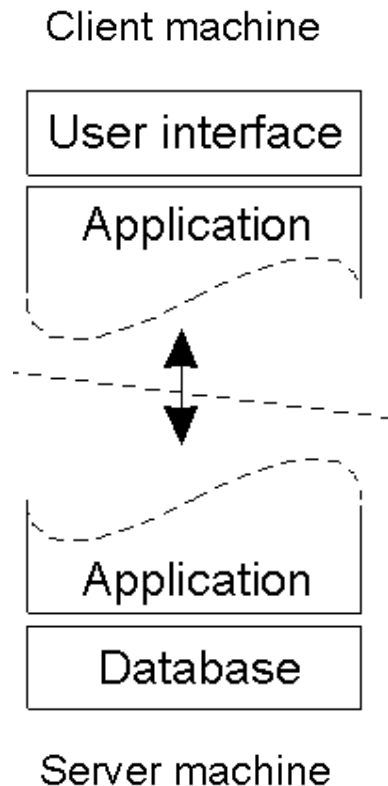
- Client:
  - Only **terminal-dependent part of the user interface**
- Give the applications remote control over the presentation of their data

# Two-tiered Architectures



- Client:
  - Graphical front end
- Communicates with the rest of the application (at the server) through an application-specific protocol
- Client software **only** processes what is necessary for presenting the **interface**

# Two-tiered Architectures

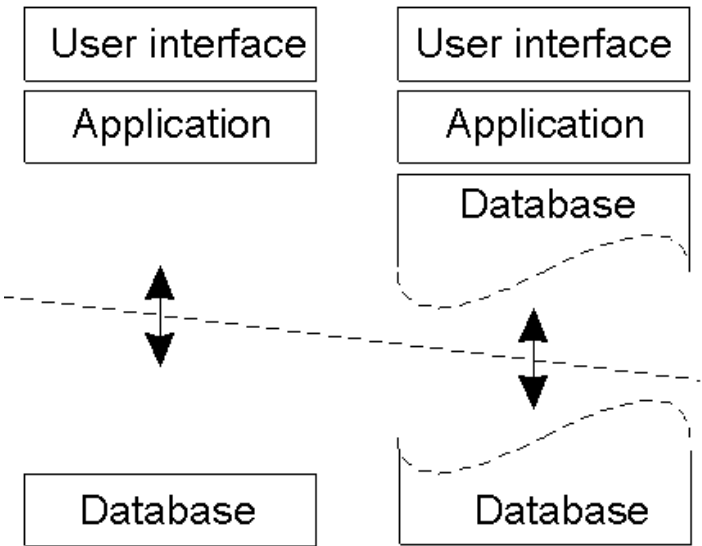


(c)

- Client:
  - Has also **part of the application**
  - e.g., filling in **forms**
  - filled in, checked, interact with user
  - Also word processors, where more advanced checking, e.g. grammar, may occur at the server

# Two-tiered Architectures

Client machine



Server machine

(d)

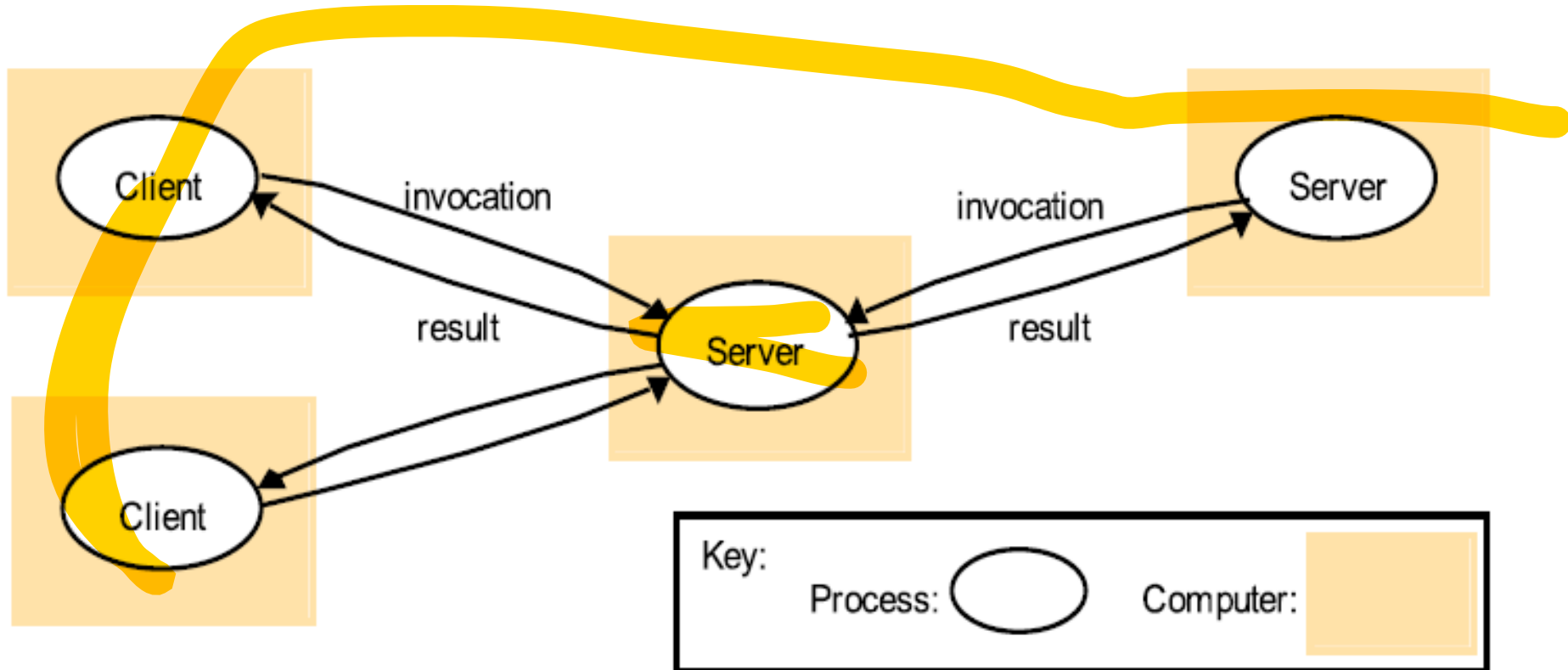
(e)

- Popular organisations
- Client:
  - PC or workstation
  - Connected to a distributed file system or database
  - Running **most of the application**
- Server:
  - Operations on **files** and **database entries**
- e.g., **banking applications**
- (e) : e.g., **caching** at the client

# Multitiered Architectures

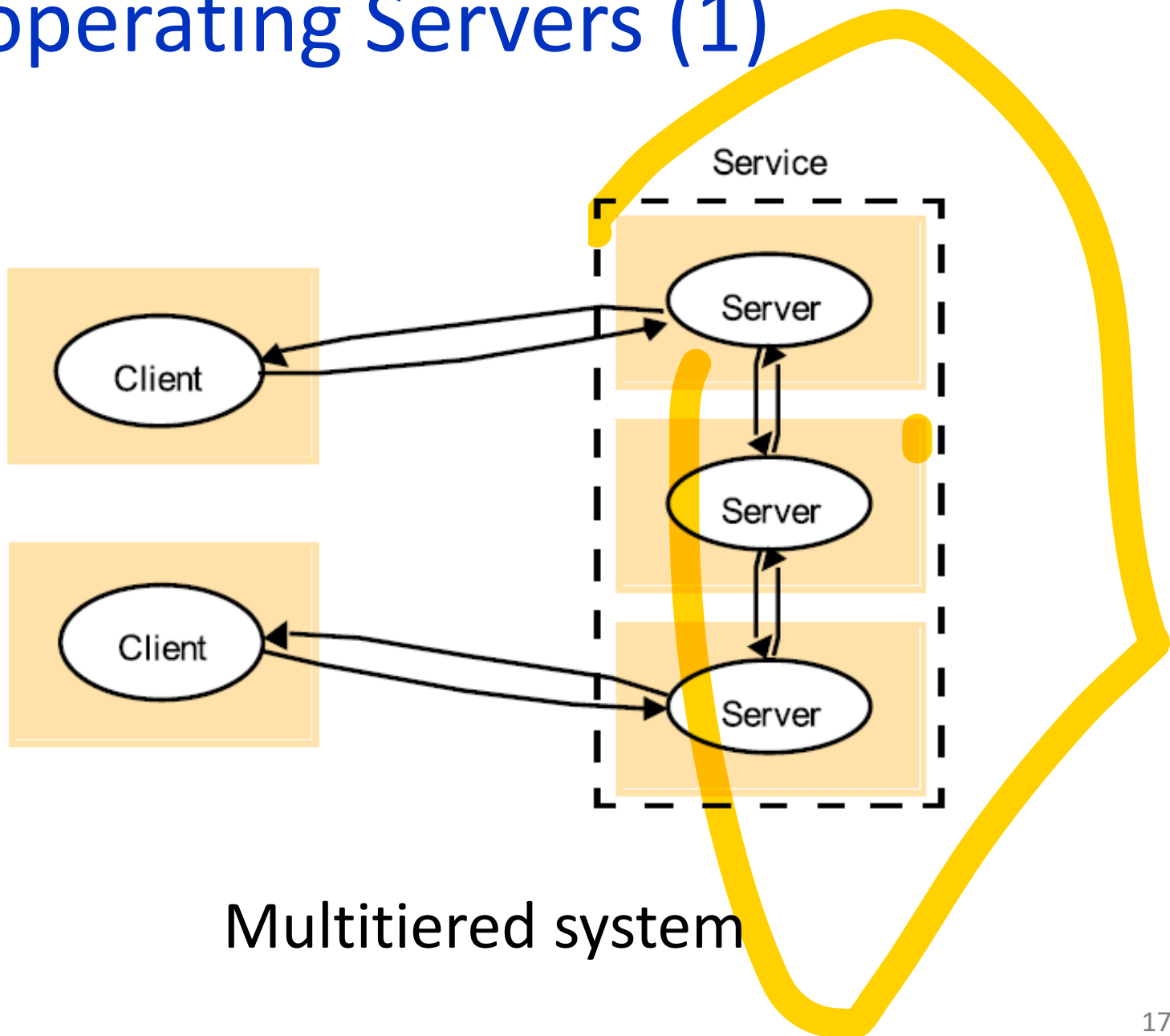
- There is a recent trend to move away from (d) and (e)
- Most of the processing and data storage is moving to the server
- **Thin clients** of (a)-(c) instead of **fat clients** (d), (e)
- Because:
  - Client machines are **difficult to manage**
- Therefore, concentrate functionalities at servers
  - *Is this distributed?*
  - **Yes:** Servers themselves are becoming more and more distributed; replaced by **multiple servers** running on different machines
- A server may sometimes need to also act as a client
  - **Three-tiered** architecture (and multitiered in general)

# Multitiered Architectures

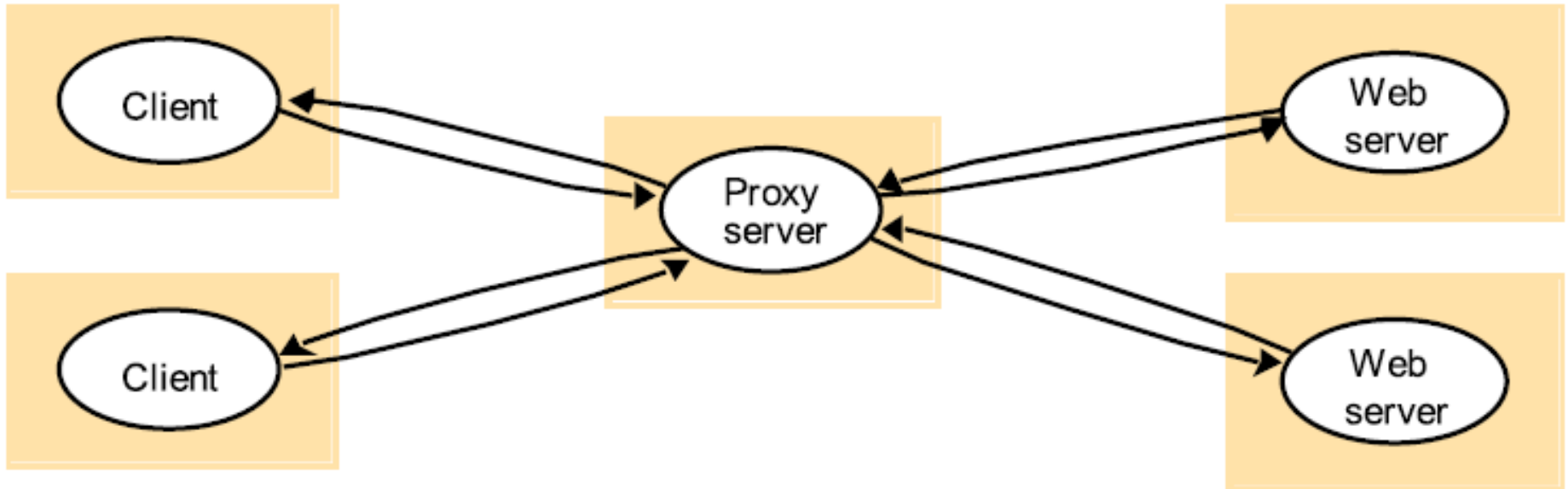




# Cooperating Servers (1)

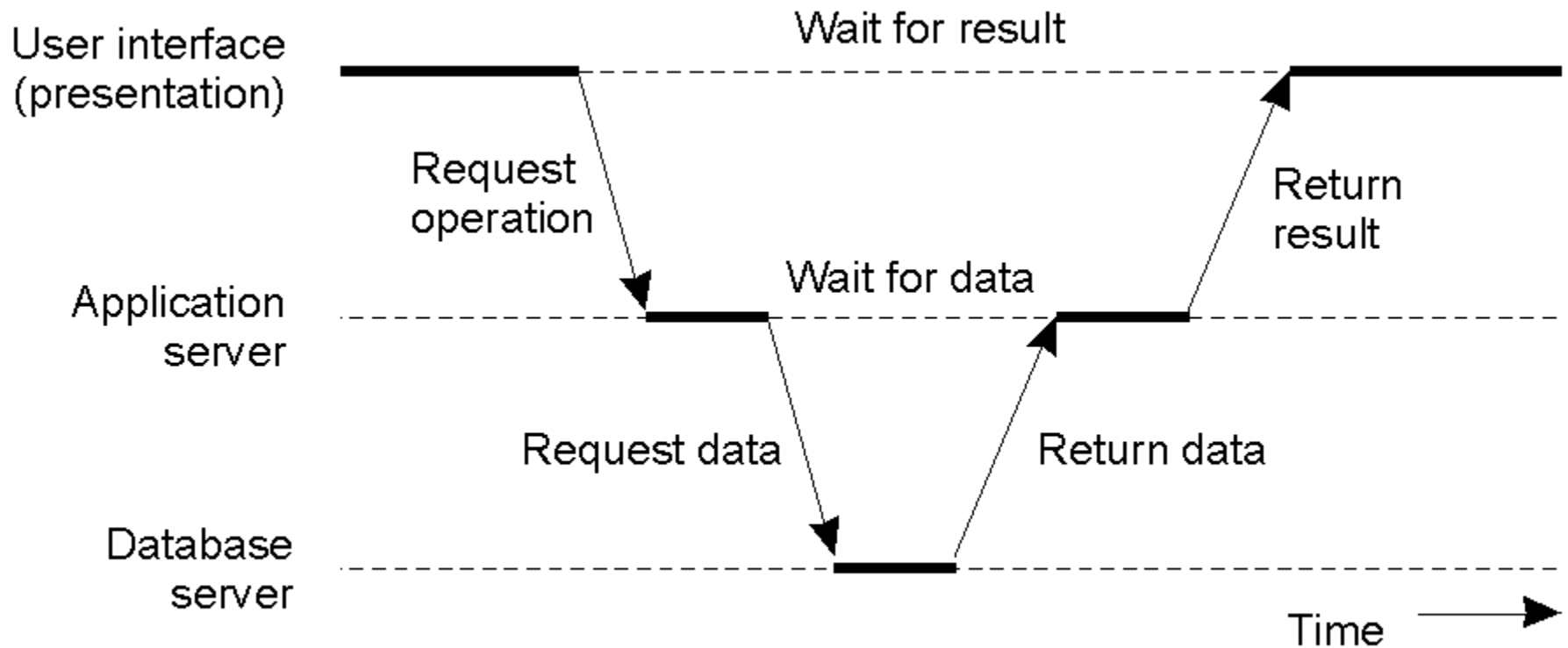


# Cooperating Servers (2)



Proxy server

# Multitiered Architectures

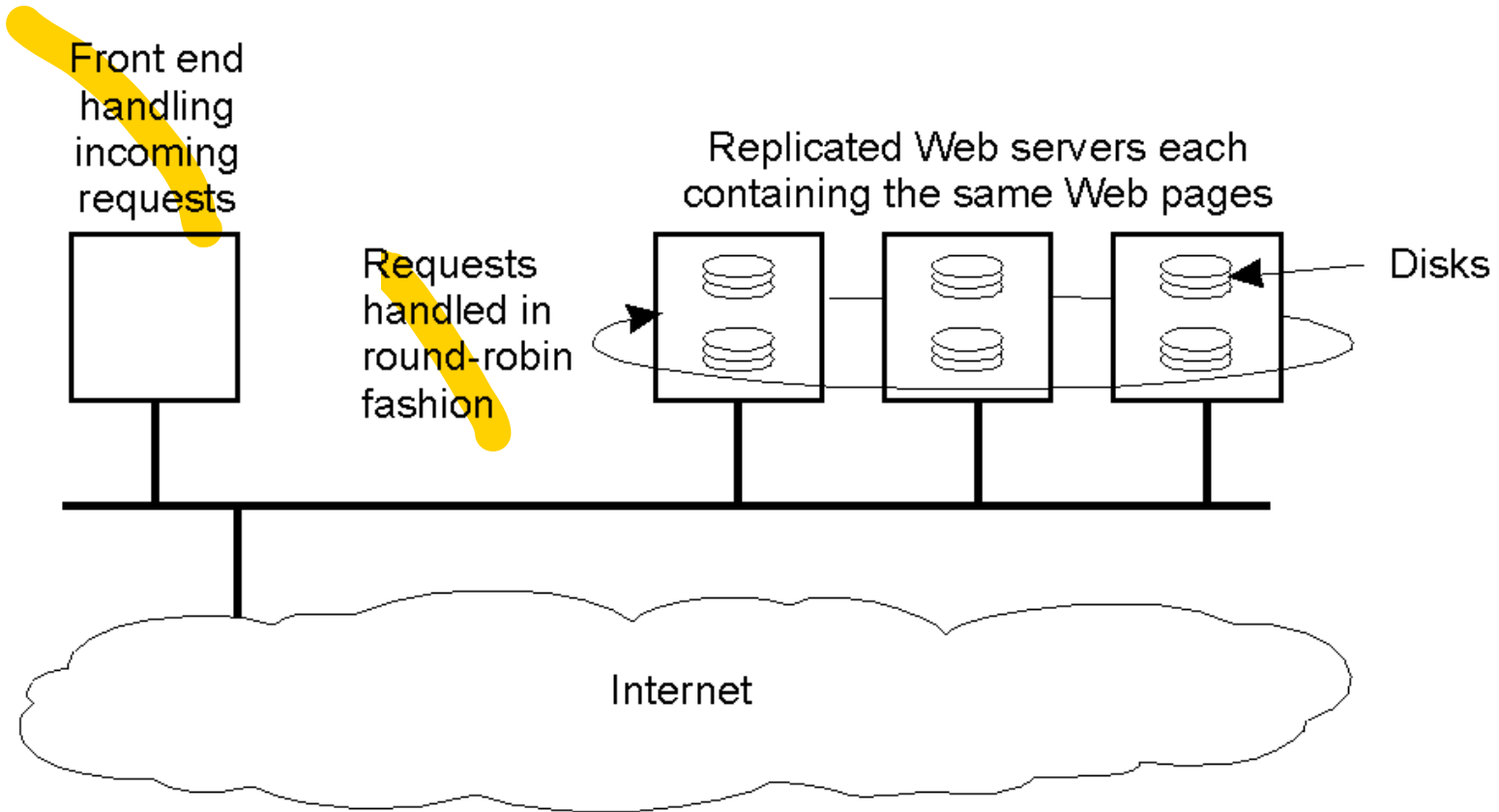


- An example of a server acting as a client
  - This is a very common **vertical distribution model** for distributed systems.
- An application: Web sites
  - Web server (entry point to site) ↔ Application Server (actual processing) ↔ Database server (raw data)

# Vertical vs Horizontal Splitting

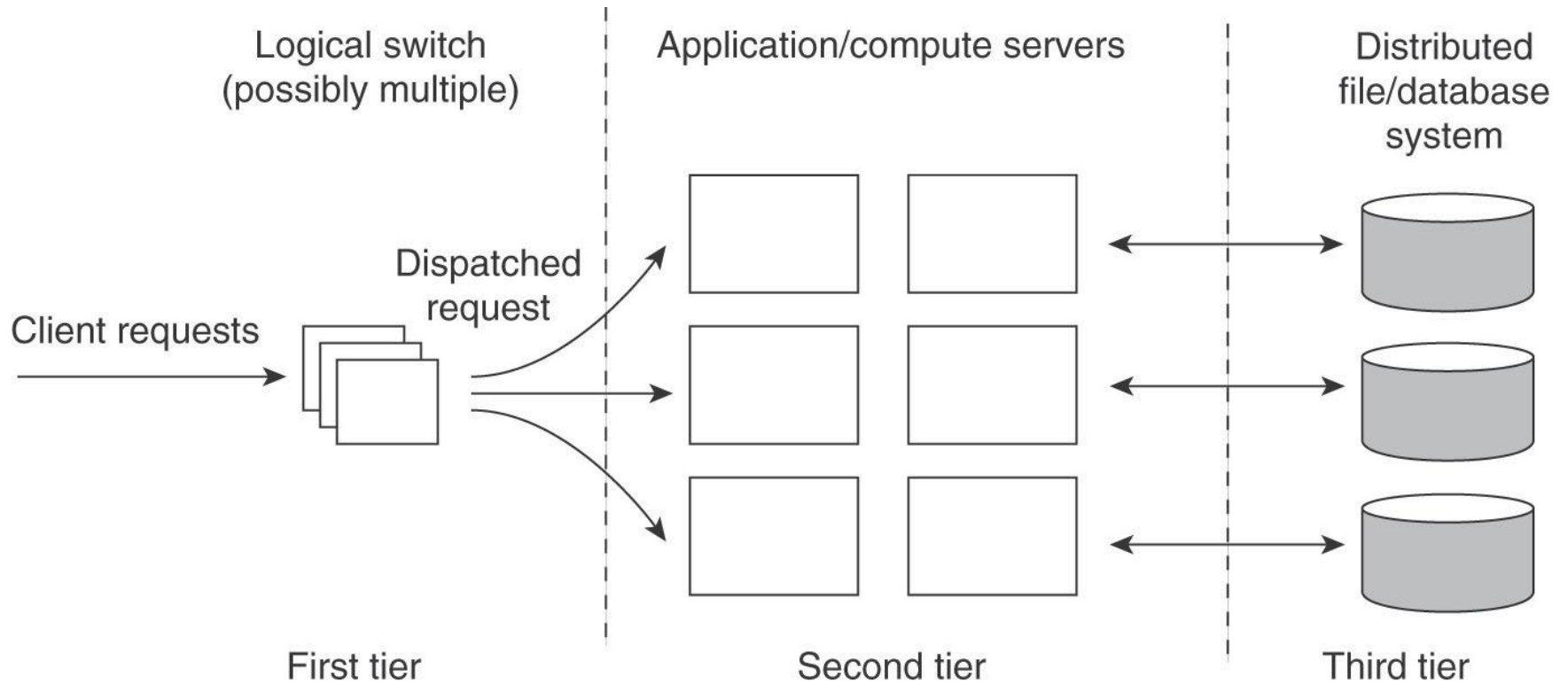
- **Vertical distribution**: placing logically different components on different machines
- **Horizontal distribution**: a client or server may be physically split up into logically equivalent parts, each operating on its own share of the complete data
  - Examples: server farms/clusters, peer-to-peer systems

# An Example of a Modern Architecture



- An example of **horizontal distribution** of a Web service
  - Often also referred to as **clustering**

# More Like This



# Other Architectures

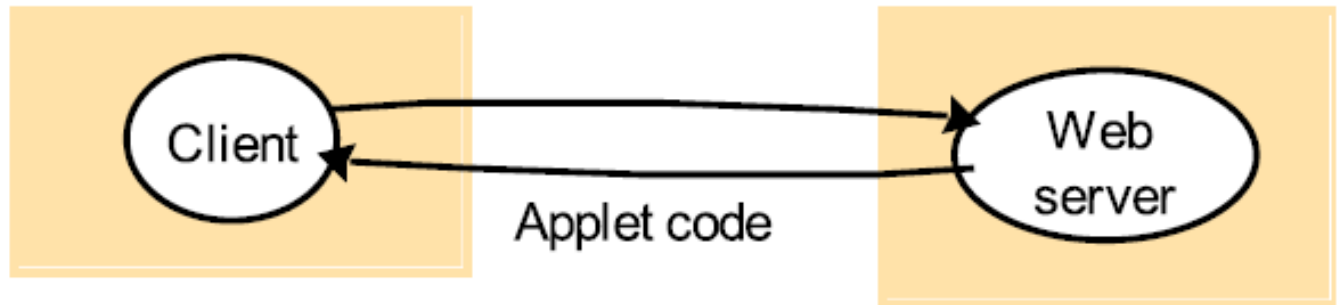
# Alternative Architectures

- Cooperating clients
  - Teleconferencing
- Web applets
- Peer-to-peer networks



# Web Applets

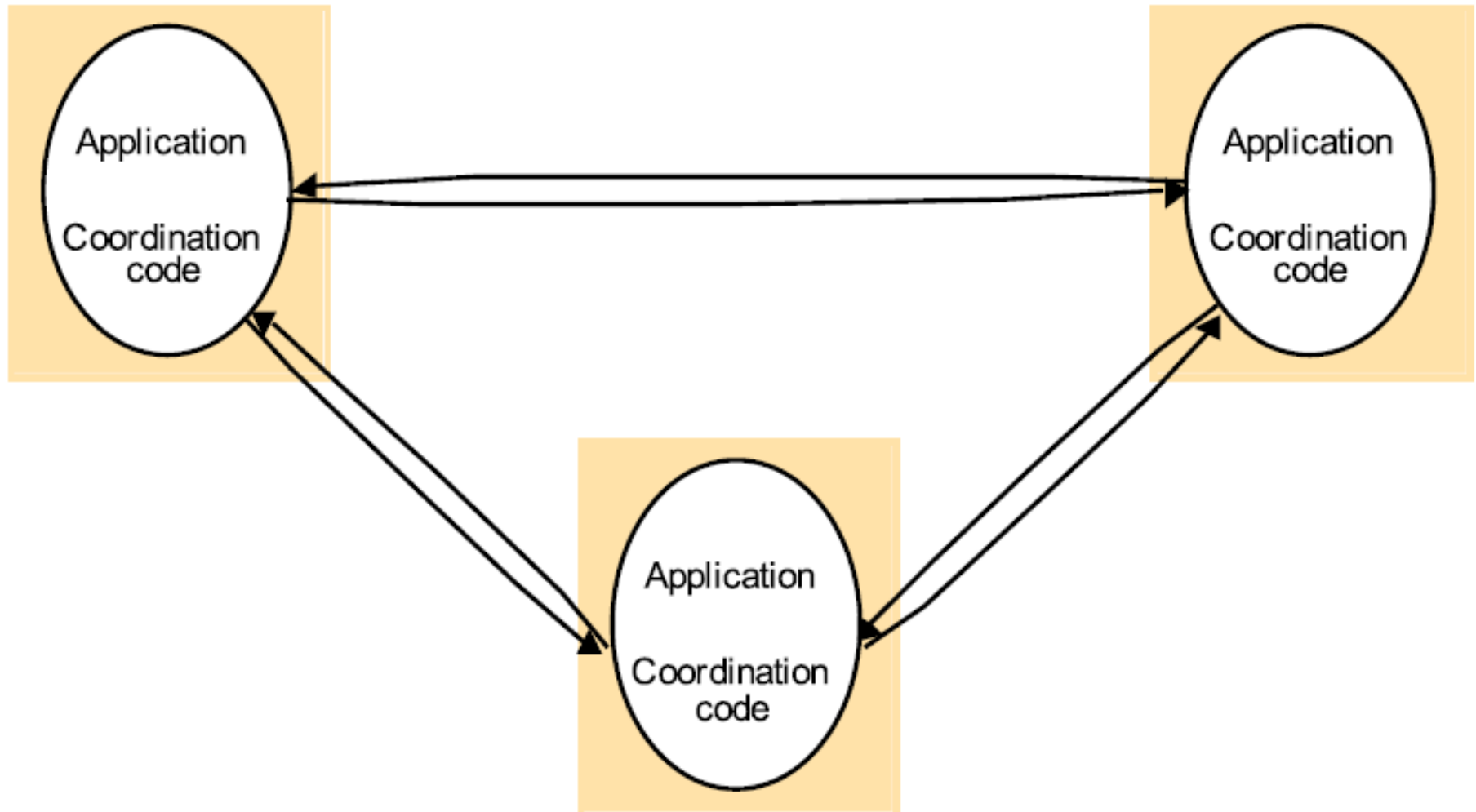
a) client request results in the downloading of applet code



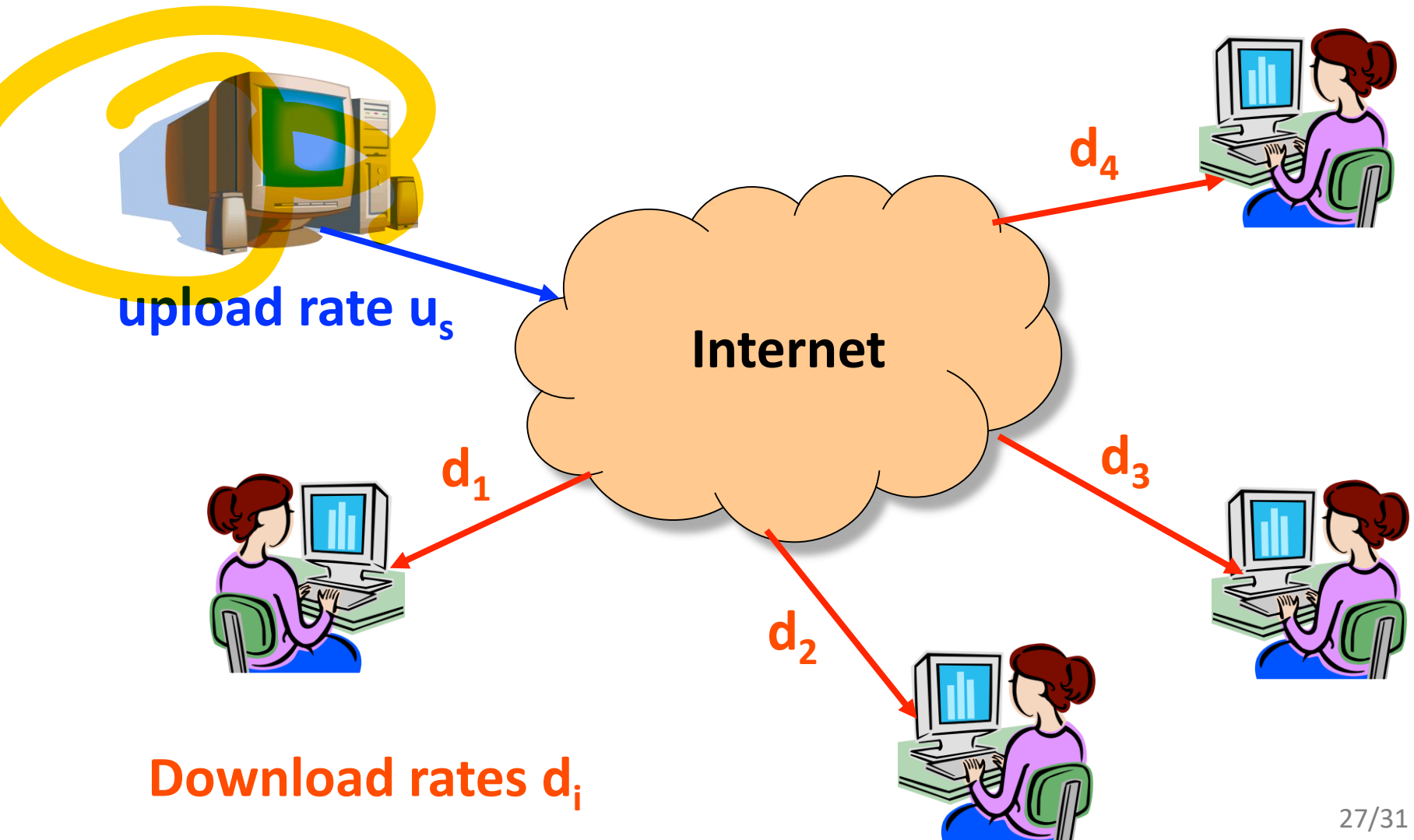
b) client interacts with the applet



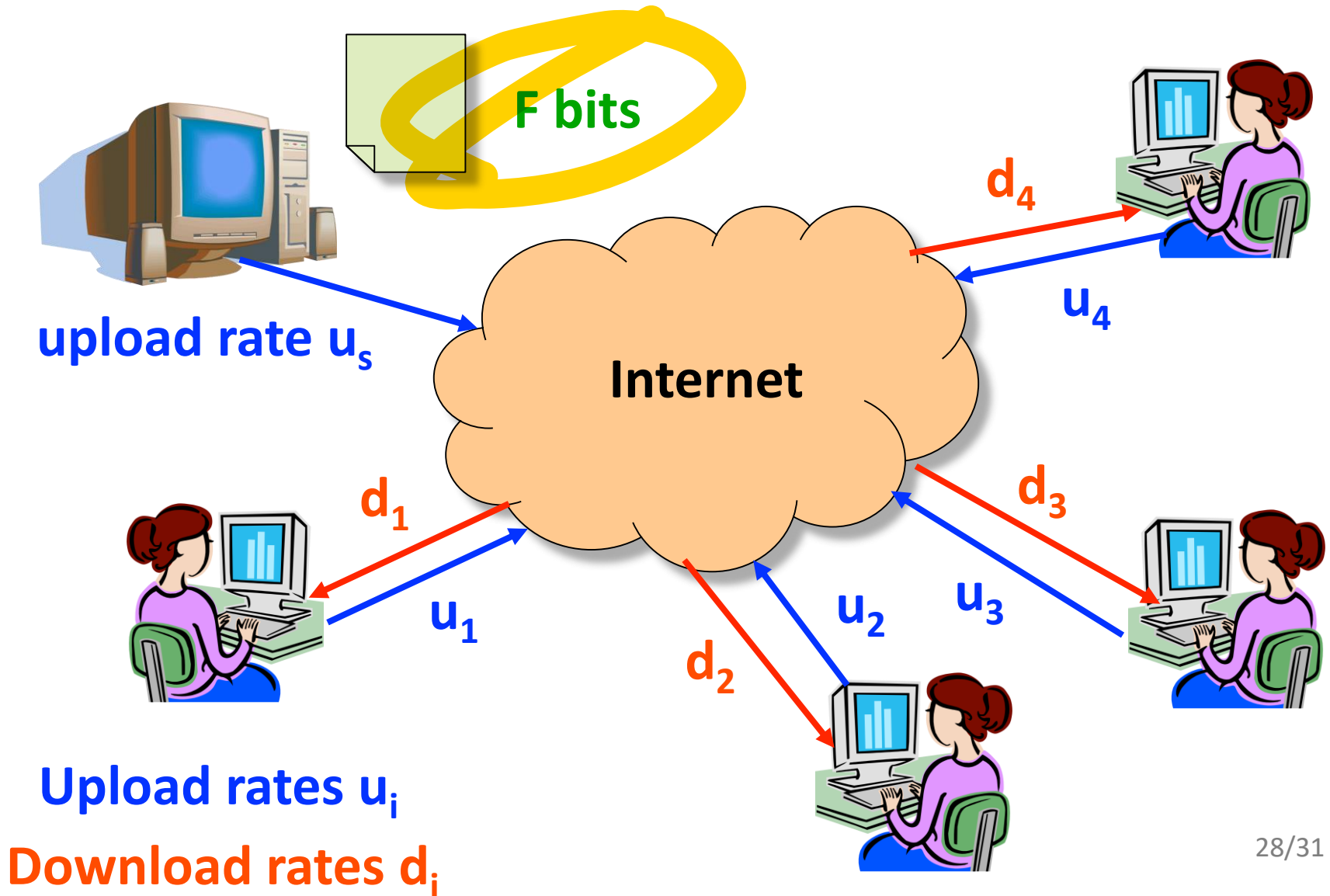
# Peer-to-peer Networks



# Server Distributing a Large File



# Peers Help Distributing a Large File



# Brief History of P2P

- 1999: Napster
  - Centralised inventory of available files
  - Restricted to music files
  - Out of business by mid 2000
- Many followed (with novel characteristics):
  - Truly decentralised P2P networks
  - Decentralised catalogue
  - Search by asking neighbours
  - No control over distributed content

# Bit Torrent

- Developed in 2002 to speed up distribution of large files
- Control over what is being distributed – “trackers”
- Free software distributions
- Blizzard Downloader for WoW, Starcraft II
- Facebook, Twitter to update servers
- A file is split in small pieces
- Rarest pieces are distributed first (+random pieces)
- Faster uploaders get priority
- 30-50% of all traffic

# Conclusions

- Architecture of modern distributed systems can be rather involved
- Tiered servers
- Cooperating servers / clients
- Peer-to-peer
- **Client-server communication** is still the most popular form of communication