Lecture 2

Application Layer

Networks Administration

Outline

- **Application Architectures**
  - Host-Based, Client-Based, and Client-Server Architectures, Choosing Architectures
- **World Wide Web**
  - How the Web Works, Inside an HTTP Request & HTTP Response
- **Electronic Mail**
  - How E-Mail Works, Inside an SMTP Packet
  - Listserv, Attachments in MIME
- **Other Applications**
  - Ftp, Telnet, Instant Messaging, Videoconferencing

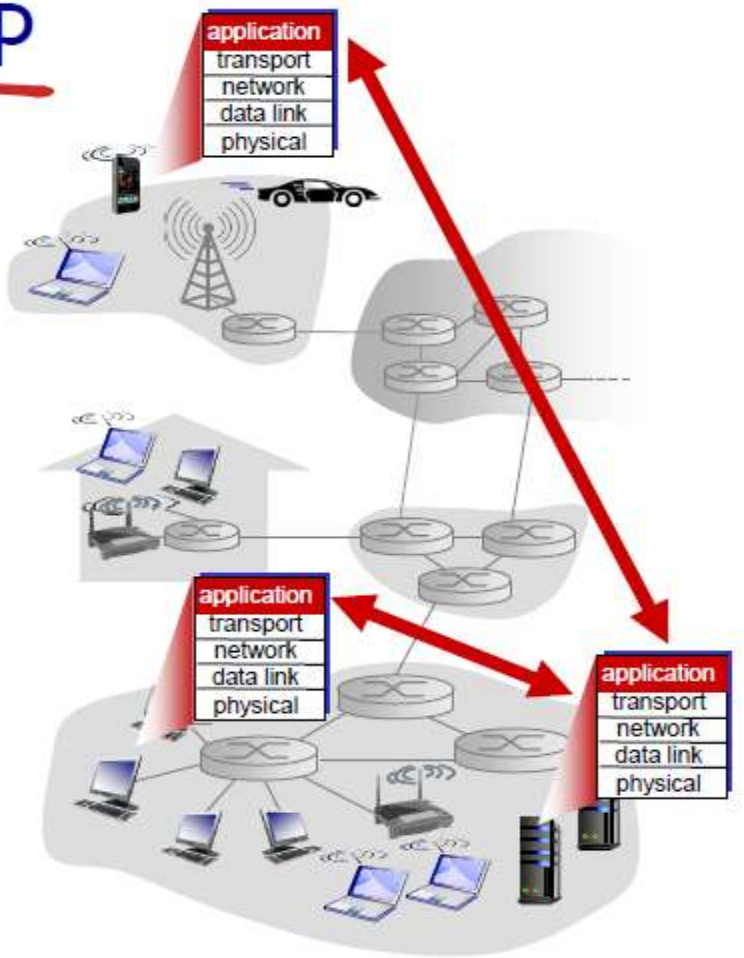
Creating a network app

write programs that:

- run on (different) *end systems*
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation

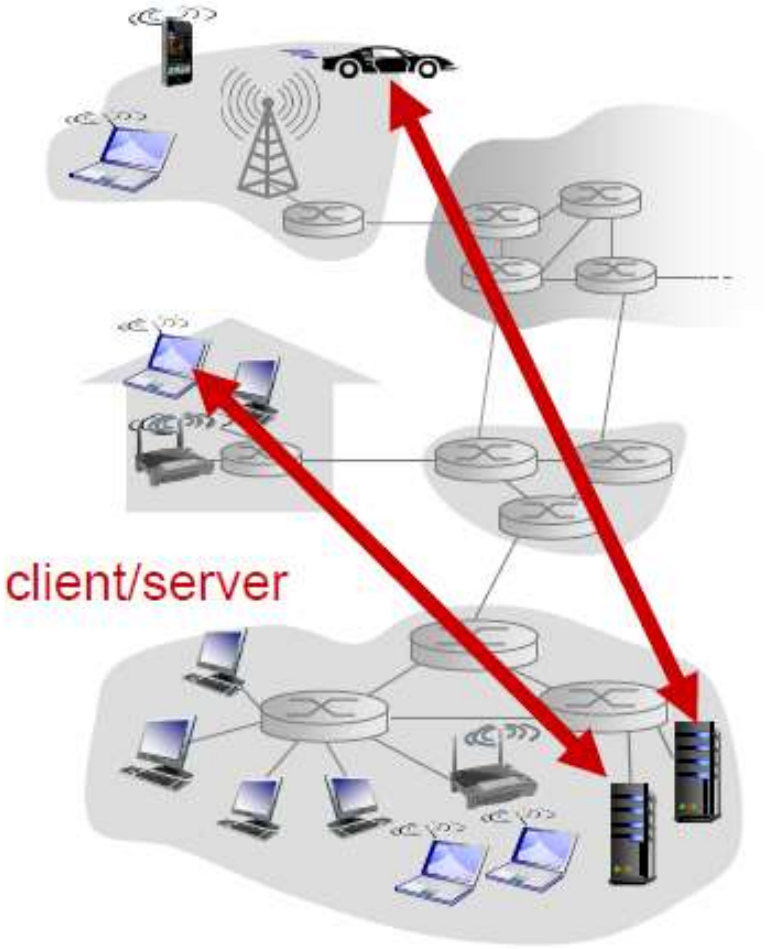


Application architectures

possible structure of applications:

- client-server
- peer-to-peer (P2P)

Client-server architecture



server:

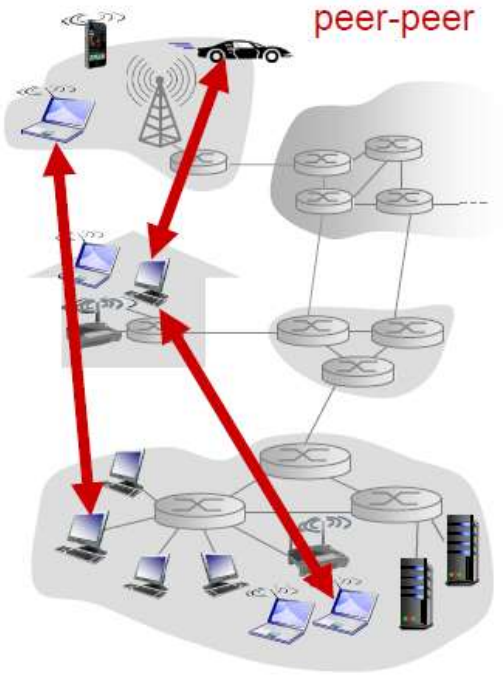
- always-on host
- permanent IP address
- data centers for scaling

clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management



Processes communicating

process: program running within a host

- within same host, two processes communicate using inter-process communication (defined by OS)
- processes in different hosts communicate by exchanging messages

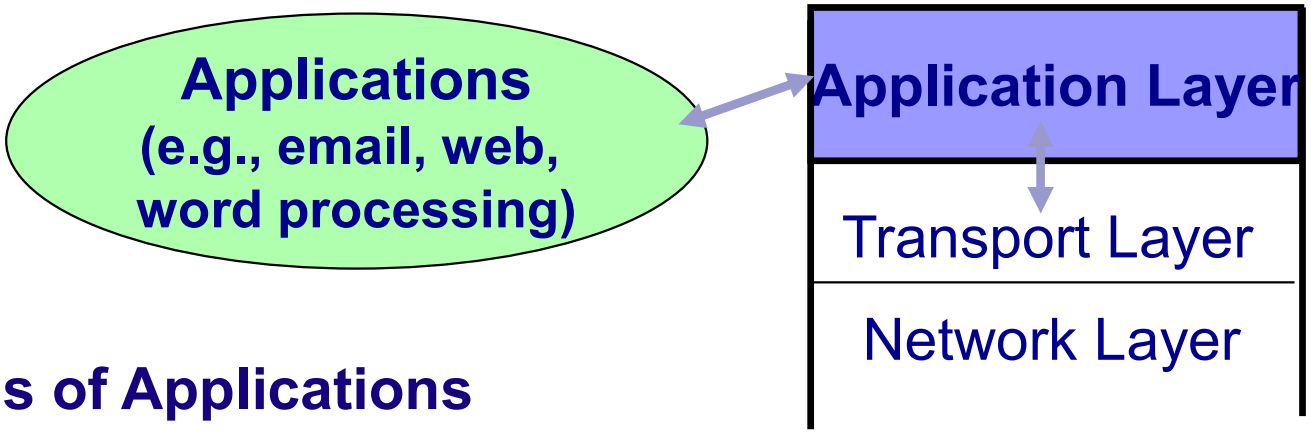
clients, servers

client process: process that initiates communication

server process: process that waits to be contacted

- aside: applications with P2P architectures have client processes & server processes

Application Layer - Introduction

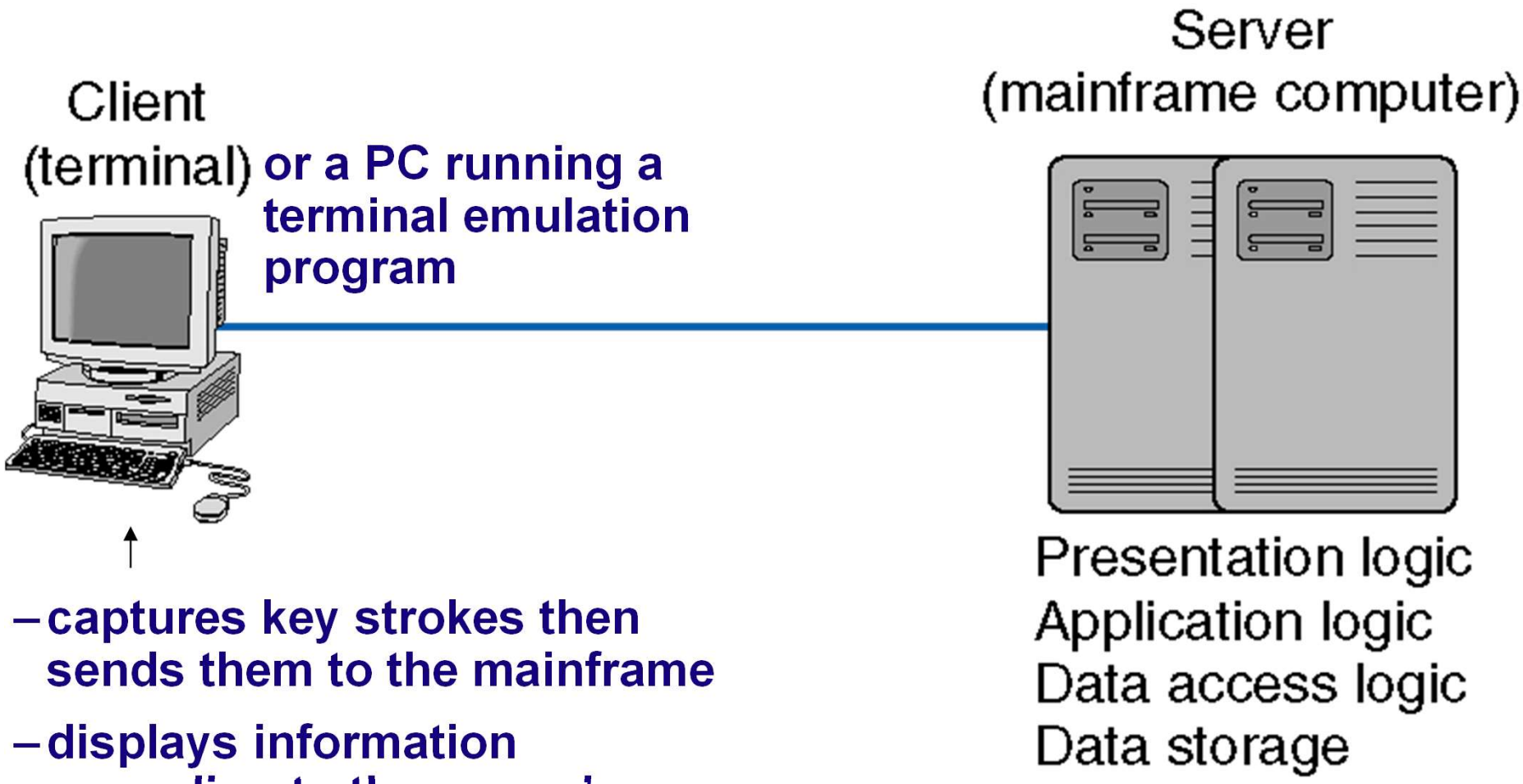


- **Functions of Applications**
  - **Data storage**
    - **Storing of data generated by programs (e.g., files, records)**
  - **Data access logic**
    - **Processing required to access stored data (e.g., SQL)**
  - **Application logic**
    - **Business logic**
  - **Presentation logic**
    - **Presentation of info to user and acceptance of user commands**

Application Architectures

- Determined by how functions of application programs are spread among clients and servers
  - **Host-based Architectures**
    - Server performs almost all functions
  - **Client-based architectures**
    - Client performs most functions
  - **Client-server architectures**
    - Functions shared between client and server

Host-Based Architectures

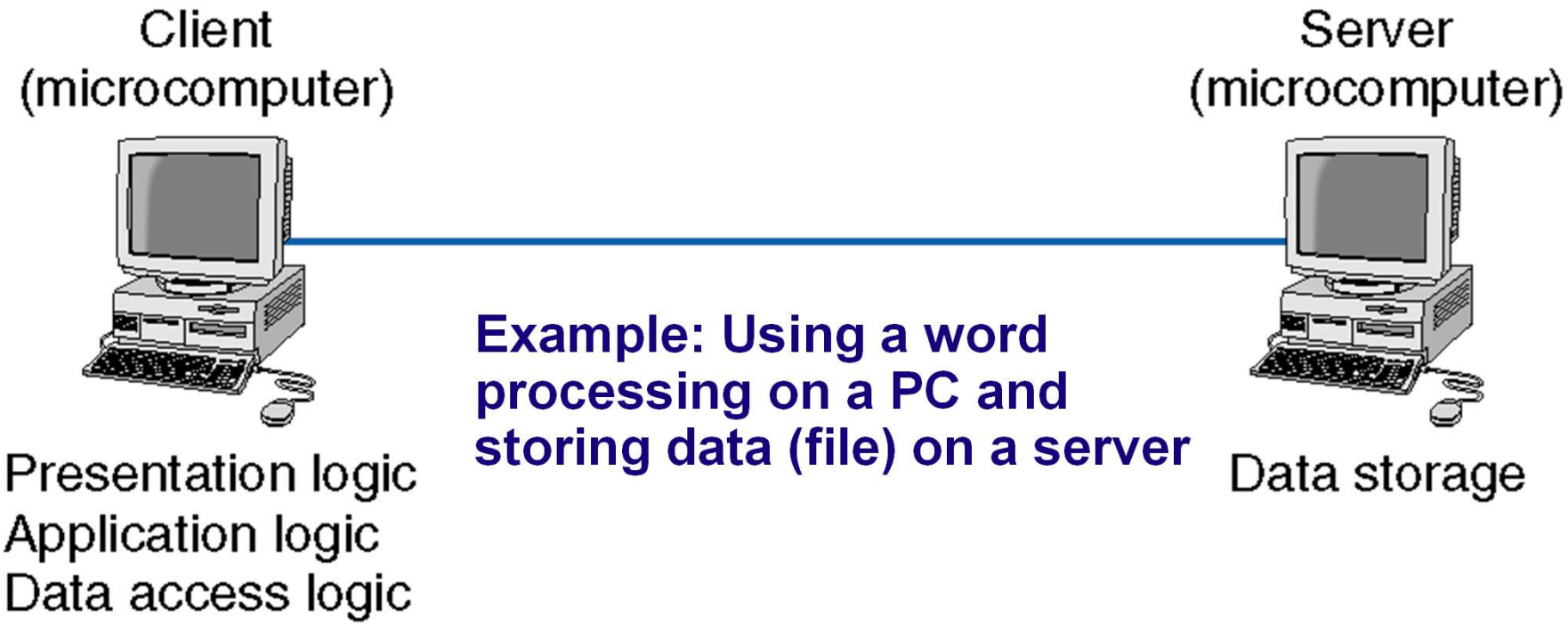


- captures key strokes then sends them to the mainframe
- displays information according to the server's instructions

Problems with Host-based Arch.

- Host becoming a bottleneck
  - All processing done by the host, which can severely limit network performance
- Upgrades typically expensive
  - Available upgrades require big jumps in processing and memory → $$$
  - Network demand grows more incrementally, so this often means a poor fit (too much or too little) between host performance and network demand.

Client-Based Architectures



Was popular in 1980s with the widespread use of PCs, LANs, and programs such as spreadsheets and word processors

Problems with Client-Based Arch.

- Data MUST travel back and forth between server and client
  - Example: when the client program is making a database query, the ENTIRE database must travel to the client before the query can be processed
  - Result in **poor network performance**

Client-Server Architectures

Used by most networks today

Client
(PC)

Server
(PC, mini, mainframe)



Example: Using a Web browser to get pages from

Presentation logic
Application logic

Data Access logic
Data Storage
Application logic

may reside on the client, server or be split up between the two

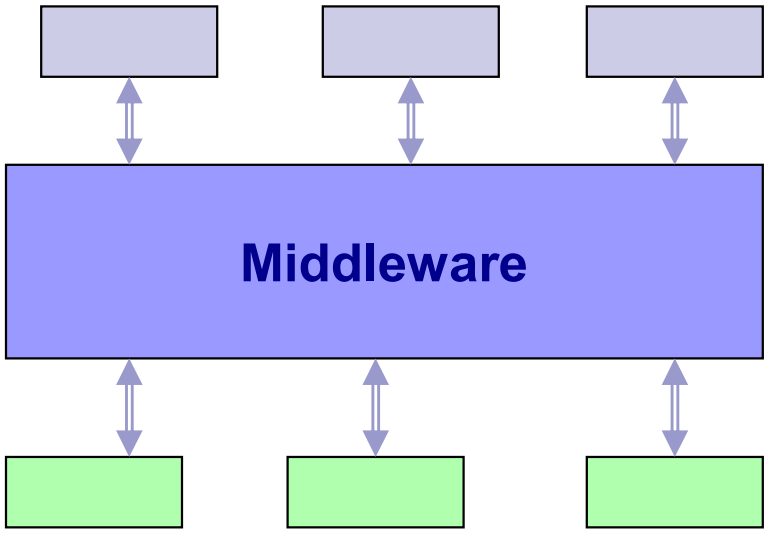
Client-Server Architectures

- **Advantages**
  - More efficient because of distributed processing
  - Allow hardware and software from different vendors to be used together
- **Disadvantages**
  - Difficulty in getting software from different vendors to work together smoothly
  - ➔May require Middleware, a third category of software

Middleware

client application programs

a standard way of translating between software from different vendors



server application programs

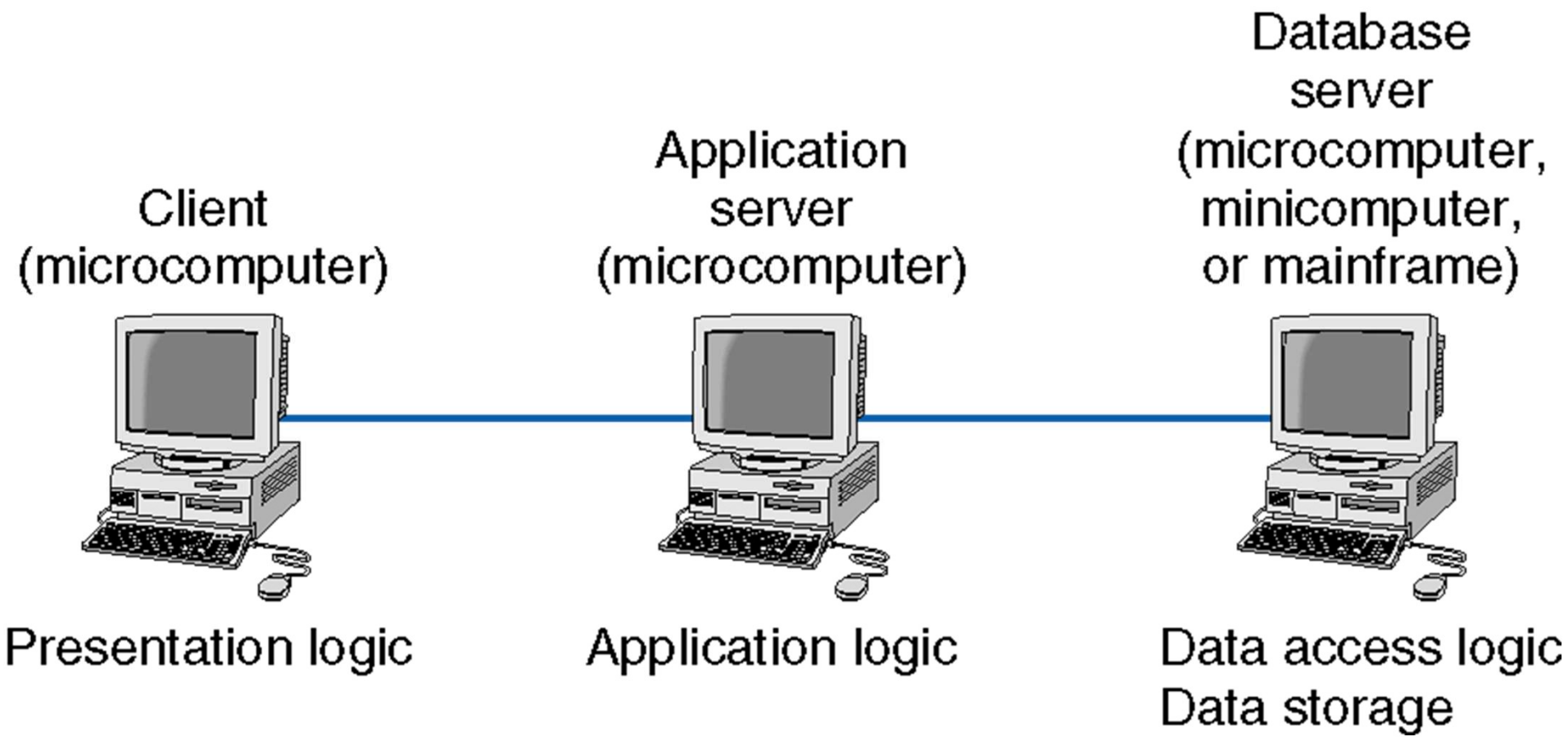
- **Manages message transfers**
- **Insulates network changes from the clients ((e.g., adding a new server)**

- Examples:
  - Distributed Computing Environment (DCE)
  - Common Object Request Broker Architecture (CORBA)
  - Open Database Connectivity (ODBC)

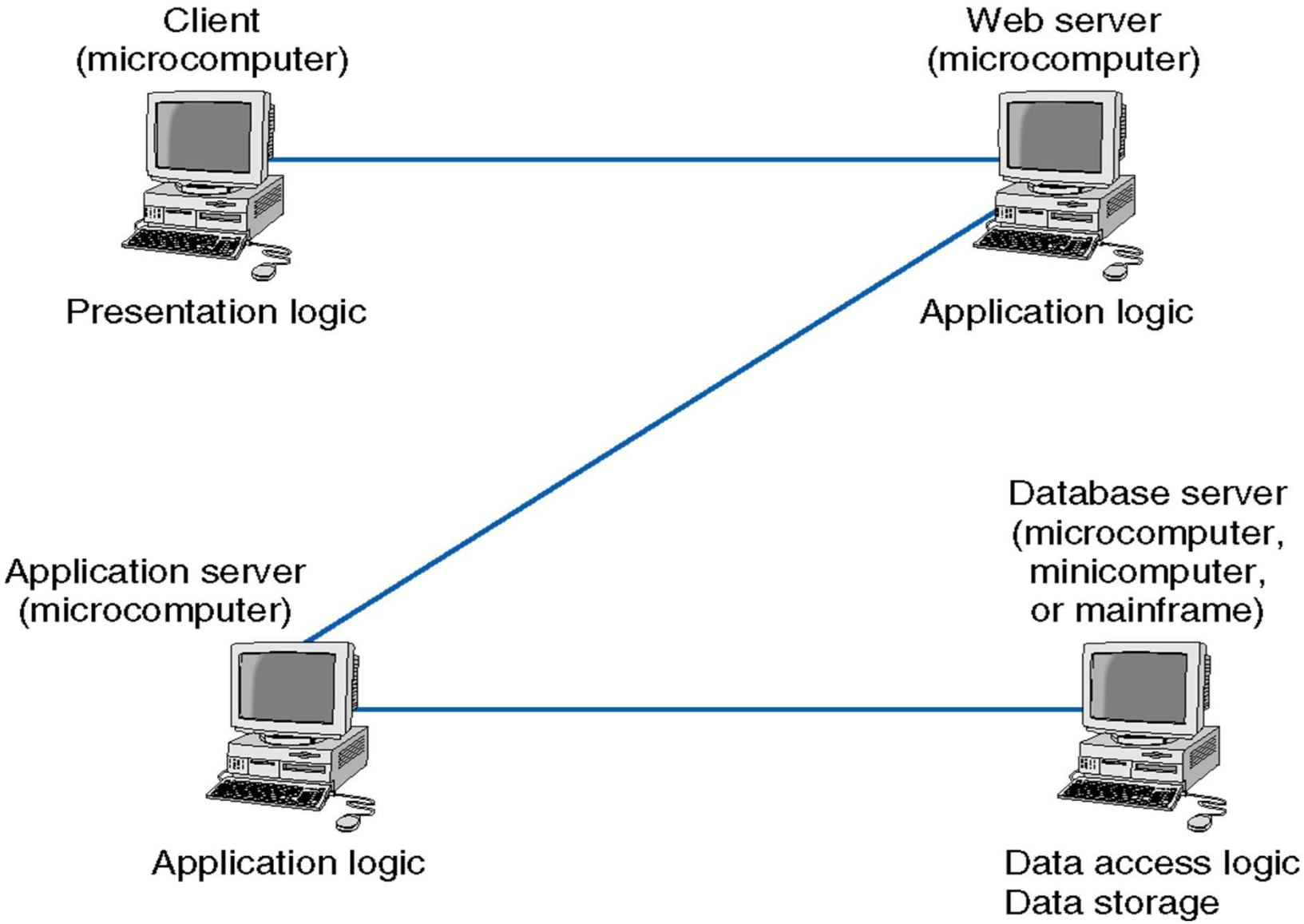
Multi-tier Architectures

- Involve more than two computers in distributing application program logic
  - 2-tier architecture (architectures discussed so far)
  - 3-tier architecture
    - 3 sets of computers involved
  - N-tier architecture
    - more than three sets of computers used

3-tier Architecture



N-tier Architecture



Multi-tier Architectures

- **Advantages**
  - Better load balancing:
    - More evenly distributed processing. (e.g., application logic distributed between several servers.)
  - More scalable:
    - Only servers experiencing high demand need be upgraded

- **Disadvantages**
  - Heavily loaded network:
    - More distributed processing → more exchanges
  - Difficult to program and test due to increased complexity

Fat vs. Thin Clients

- Depends on how much of the application logic resides on the client
  - **Fat client**: (a.k.a., thick client)
    - All or most of the application logic
  - **Thin client:**
    - Little or no application logic
    - Becoming popular because easier to manage, (only the server application logic generally needs to be updated)
    - The best example: World Wide Web architecture (uses a two-tier, thin client architecture)

Thin-Client Example: Web Architecture

