



UNIVERSITAS
INDONESIA

Veritas, Probitas, Iustitia

FAKULTAS
ILMU
KOMPUTER

TOPIC 9

**PHYSICAL
ARCHITECTURE
DESIGN**

ANALISIS DAN PERANCANGAN SISTEM INFORMASI
CSIM603183

Learning Objectives

1. Able to explain the differences among the components of computer physical architecture
2. Able to explain server-based, client-based, and client-server physical architecture.
3. Able to create network model by using deployment diagram
4. Able to explain the influence of non-functional requirement toward physical architecture layer design
5. Able to create hardware and software architecture document

Learning Objectives

- **Architecture design**

- Plans for how the system will be distributed across computers and what the hardware and software will be used for each computer.
- Architectural Component → consist of software and hardware

- **Hardware and software specification**

- Describes the hardware/software components in detail to aid those responsible for purchasing those products.

Architectural Components (Functions) of Software

1. Data storage
2. Data access logic
 - Processing required to access stored data
3. Application logic
 - Logical processing of the application
4. Presentation logic
 - Information display and user command processing

Architectural Design Purpose

- Determine what parts of the application software will be assigned to what hardware.
- Hardware options:
 - Clients
 - Input/output devices employed by users
 - PCs, laptops, handheld devices, cell phones
 - Servers
 - Larger computers storing software
 - Accessible by many users

Computer Devices

1. Mainframe
2. Minicomputer
3. Microcomputer (personal computer)

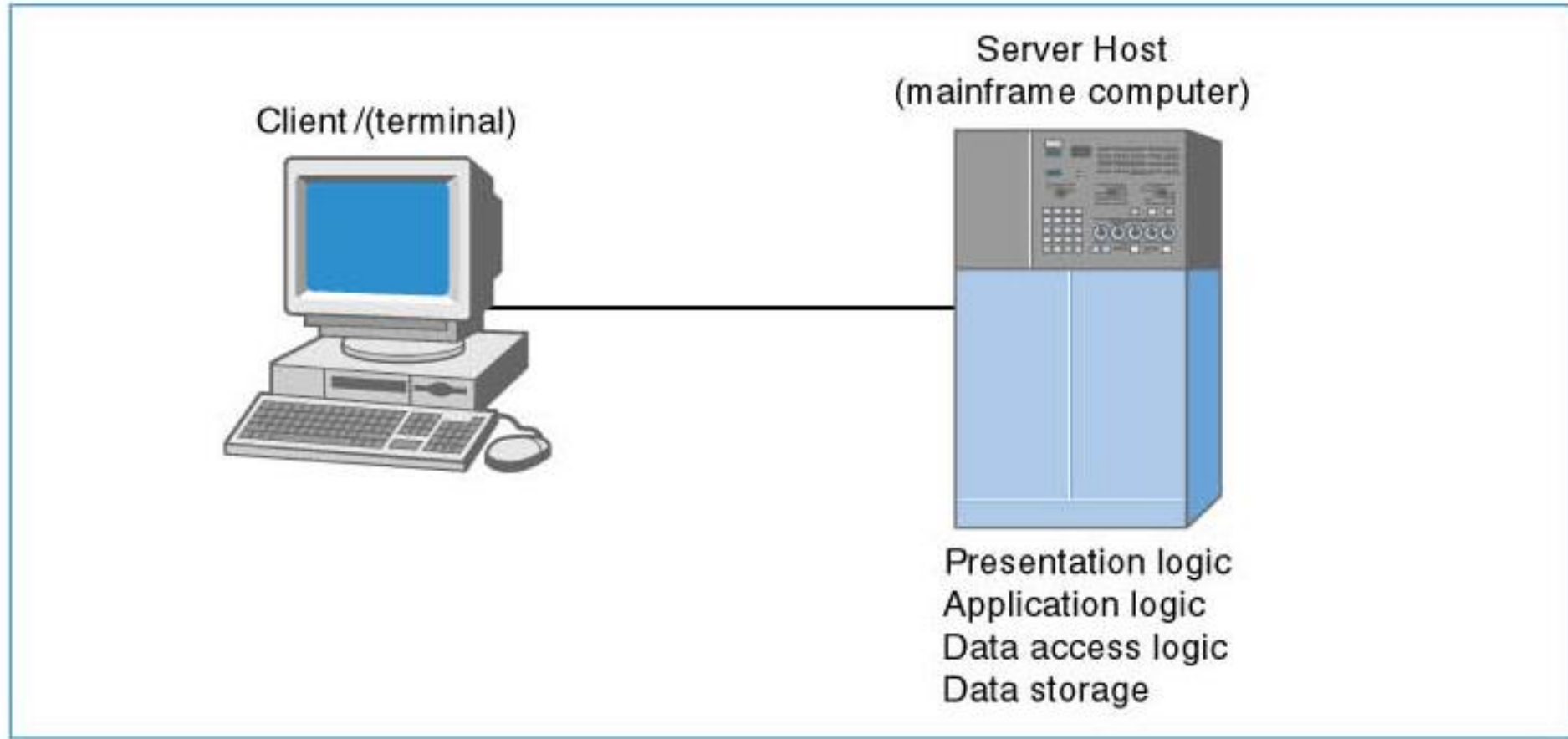
Client Devices

1. Terminals
2. Microcomputer (personal computer)
3. Special purpose terminals(ATMs, kiosks, Palm Pilots, and many others)

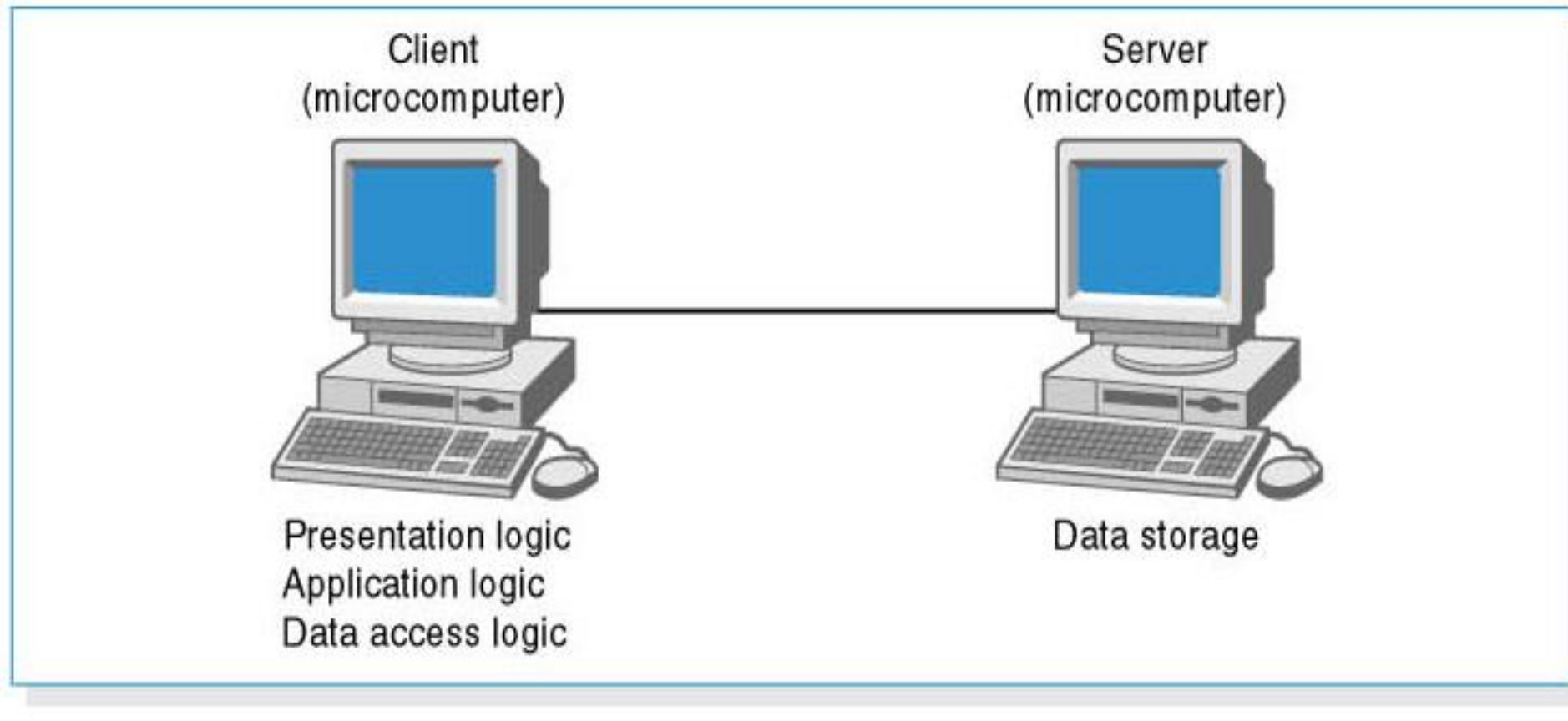
Architecture Choices

1. Server-based Architecture
2. Client-based Architecture
3. Client-server based Architecture

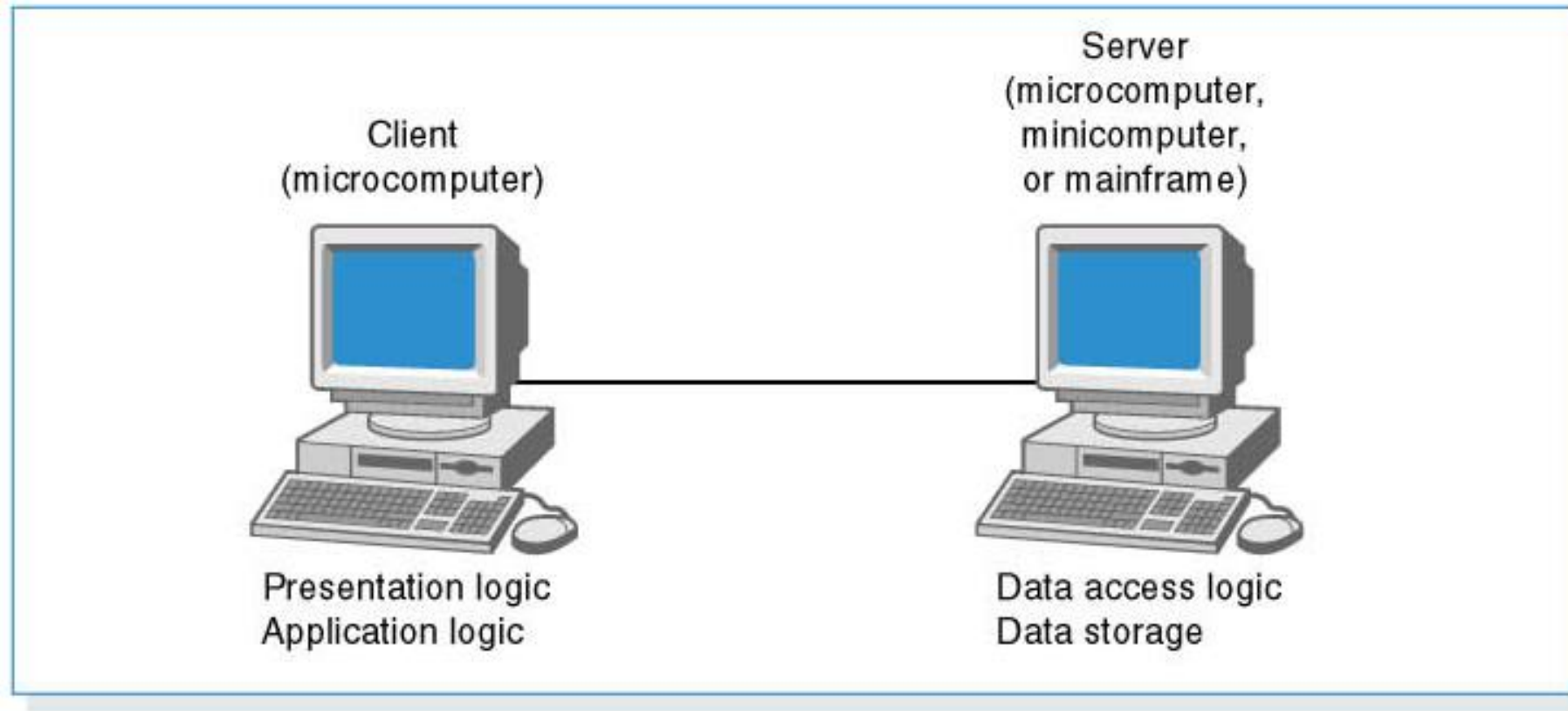
Server-Based Architecture



Client-Based Architecture



Client-Server Architecture (two-tiered)



Your Turn 😊

Discuss characteristic, cost, and benefit of:

1. Server-based Architecture
2. Client-based Architecture
3. Client-server based Architecture

Client-Server Attributes

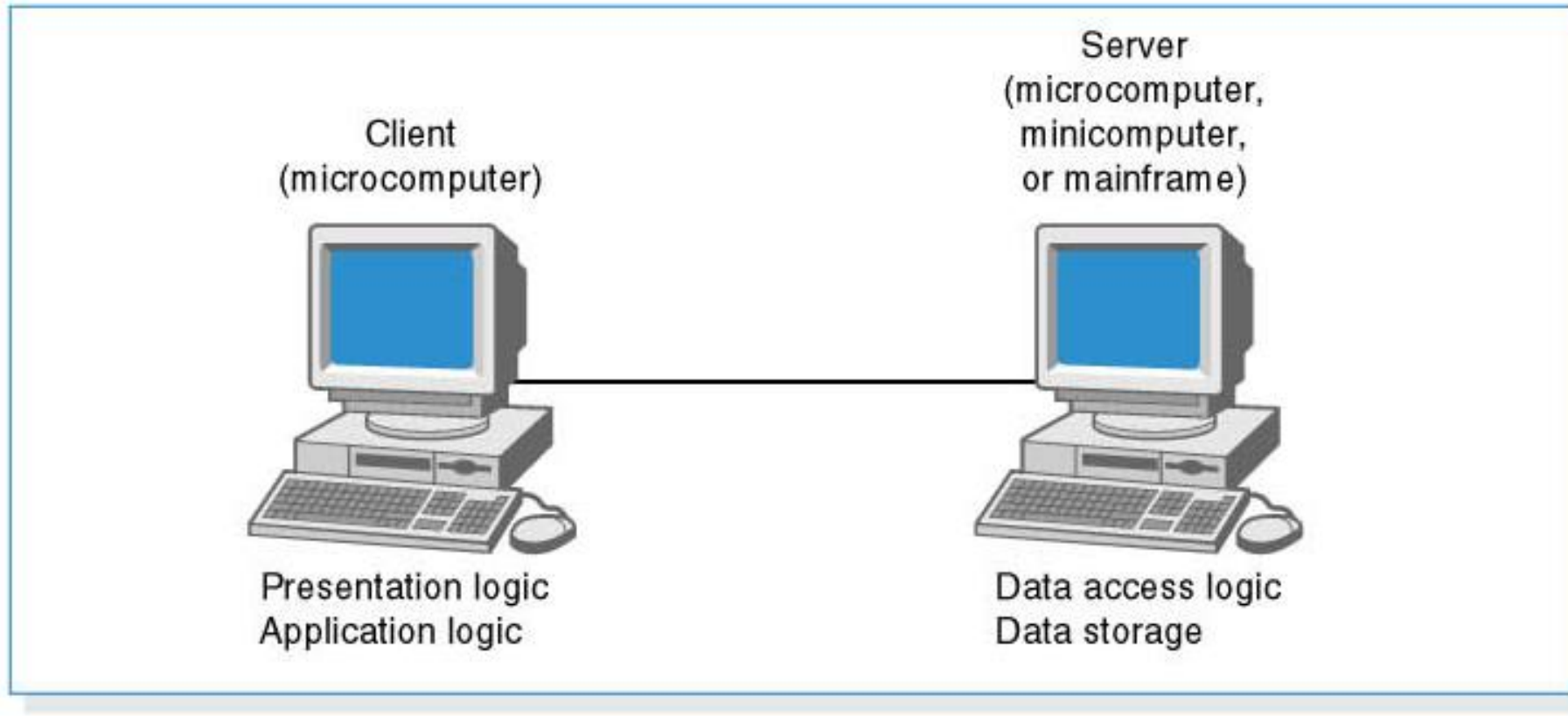
Benefits

- Scalable
- Works with multiple vendors/products of clients and servers through **middleware**
- Improved modularity of web-based systems
- No central point of failure (backup servers)

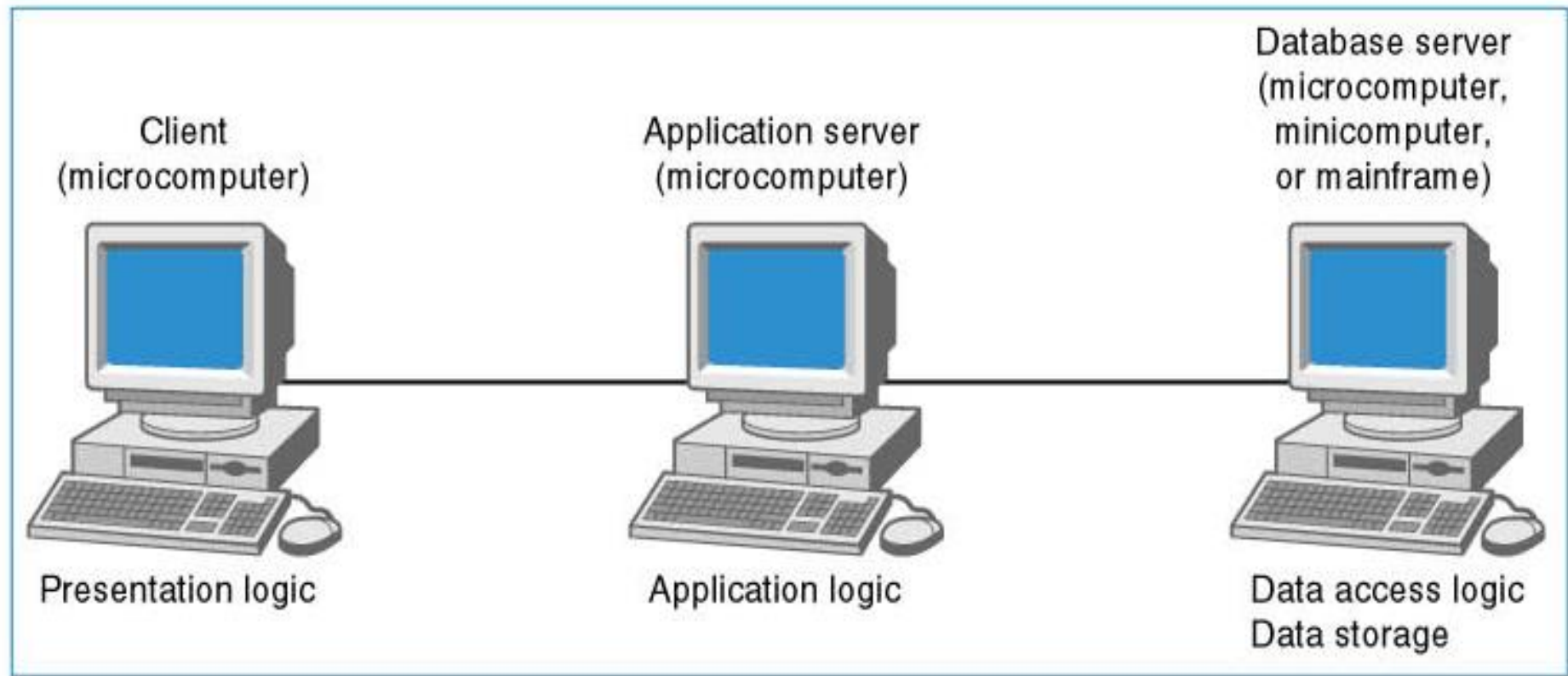
Limitations

- Complexity to update in all clients and servers
- Cost
 - Server based arch. provides highest economic of scale
 - Total Cost of Ownership (4-5 times compare to server based arch.)

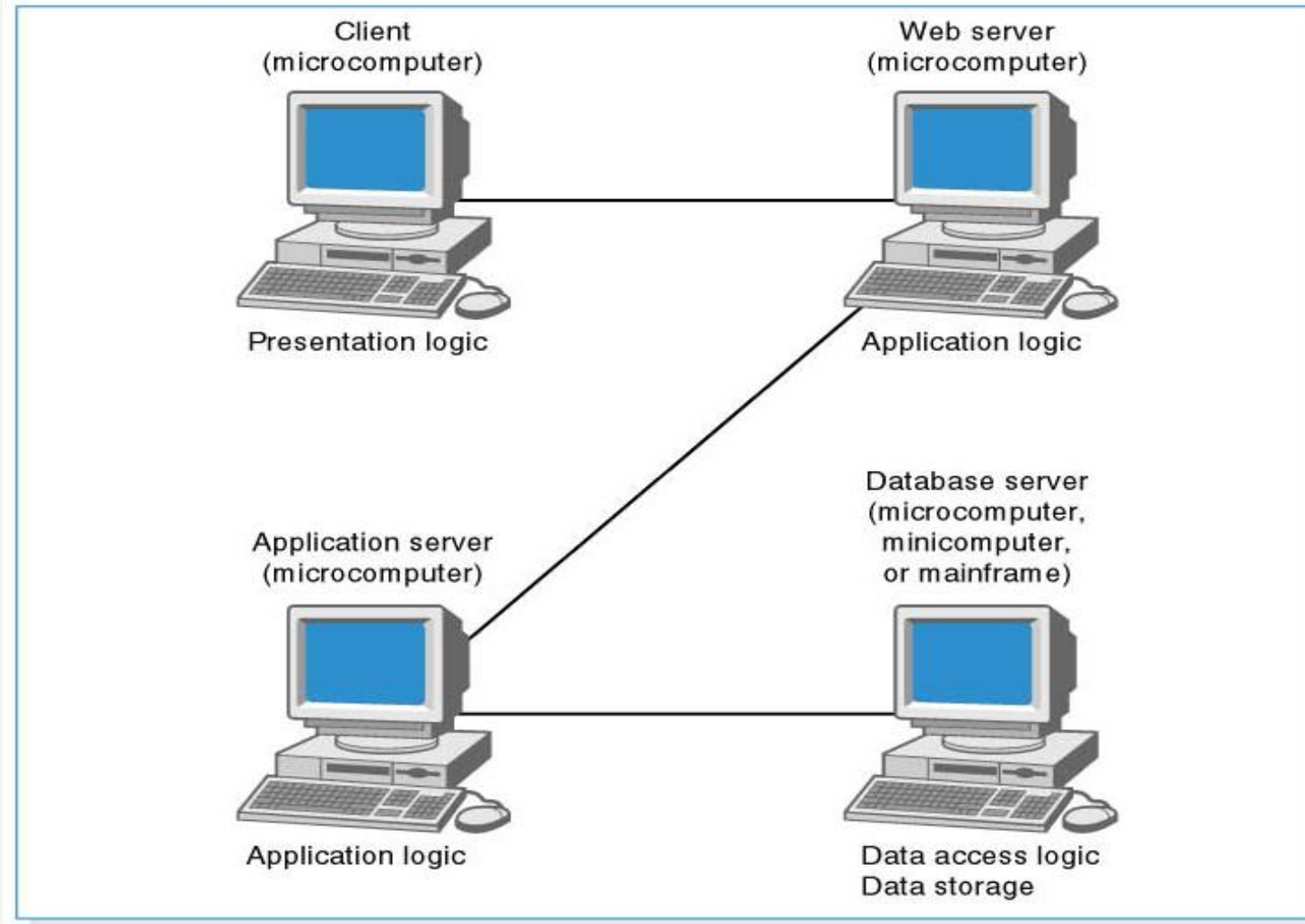
Thick Client-Server Architecture (Two-Tiered)



Three-Tiered (Thin) Client-Server Architecture



Four-Tiered Client-Server Architecture



Selecting an Architecture Design

- Most systems are built to **use the existing infrastructure** in the organization, so often the current infrastructure **restricts** the choice of architecture.
 - For example, if the new system will be built for a mainframe-centric organization, a server-based architecture may be the best option.
- Other factors such as **corporate standards, existing licensing agreements, and product/vendor relationships** can also mandate what architecture the project team needs to design.
- However, many organizations now have a variety of infrastructures available or are openly looking for pilot projects to test new architectures and infrastructures, enabling a project team to select an architecture based on other important factors

Selecting an Architecture Design

Characteristic	Server-based	Client-based	Client–Server
Cost of infrastructure	Very high	Medium	Low
Cost of development	Medium	Low	High
Ease of development	Low	High	Low to medium
Interface capabilities	Low	High	High
Control and security	High	Low	Medium
Scalability	Low	Medium	High

Cost of Infrastructure

- Personal computers are more than **1,000 times cheaper** than mainframes for the same amount of computing power.
 - The personal computers on our desks today have more processing power, memory, and hard disk space than the typical mainframe of the past, and the cost of the personal computers is a fraction of the cost of the mainframe.
- Client–server architectures also tend to be cheaper than client-based architectures because they place **less of a load on networks** and thus require less network capacity.

Cost of Development

- Developing application software for client–server computing is extremely complex, and most experts believe that it costs **four to five times more to develop and maintain application software for client–server** computing than it does for server-based computing.
- Developing application software for client-based architectures is usually cheaper still, because there are many GUI development tools for simple stand-alone computers that communicate with database servers.

Ease of Development

- In most organizations today, there is a huge backlog of mainframe applications, systems that have been approved but that lack the staff to implement them.
 - The tools for mainframe-based systems often are not user friendly and require highly specialized skills—skills that new graduates often don't have and aren't interested in acquiring.
- In contrast, client-based and client–server architectures can rely on graphical user interface (GUI) development tools that can be intuitive and easy to use.
 - The development of applications for these architectures can be fast and painless. Unfortunately, the applications for client–server systems can be very complex because they must be built for several layers of hardware (e.g., database servers, Web servers, client workstations) that need to communicate effectively with one another.

Interface Capabilities

- Typically, server-based applications contain plain, character-based interfaces, which can be quite difficult to use.
- Reversely, most users of systems expect a GUI or a Web-based interface that they can operate using a mouse and graphical objects. GUI and Web development tools typically are created to support client-based or client–server applications; rarely can server-based environments support these types of applications.

Control and Security

- The server-based architecture was originally developed to control and secure data, and it is much easier to administer because all the data are stored in a single location.
- In contrast, client–server computing requires a high degree of coordination among many components, and the chance for security holes or control problems is much more likely.
- Also, the hardware and software used in client–server architecture are still maturing in terms of security.

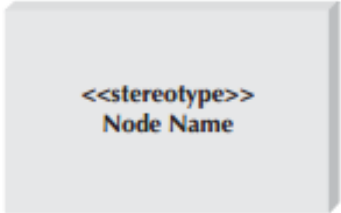
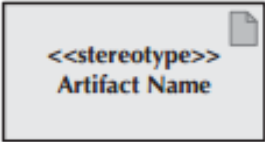
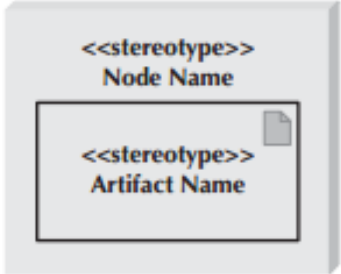

Scalability

- Scalability refers to the ability to increase or decrease the capacity of the computing infrastructure in response to changing capacity needs.
- The most scalable architecture is client–server computing:
 - servers can be added to (or removed from) the architecture when processing needs change.
 - Also, can be upgraded at a pace that most closely matches the growth of the application.
- In contrast, server-based architectures rely primarily on mainframe hardware that needs to be scaled up in large, expensive increments.

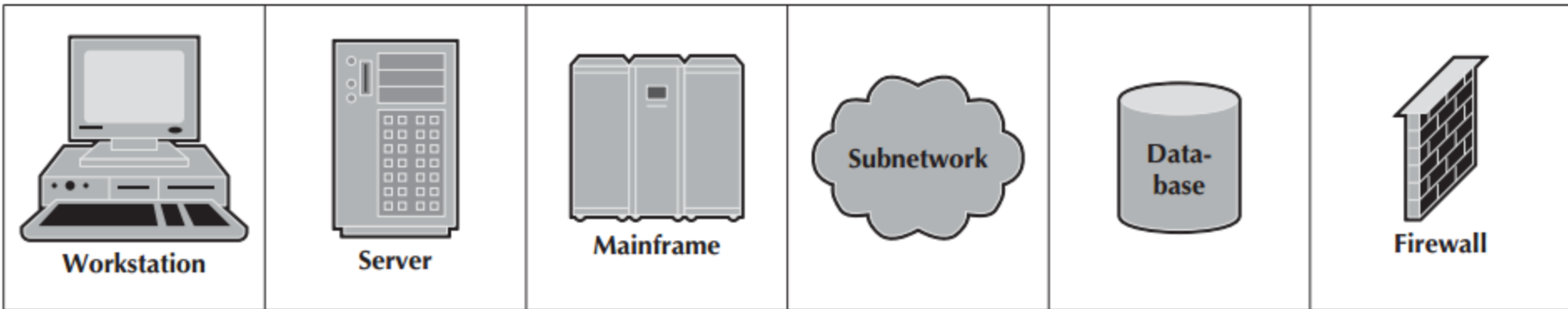
Deployment Diagram

- Deployment diagrams are used to represent the relationships between the hardware components used in the physical infrastructure of an information system.
- There are times that the notation in deployment diagram should be extended to better communicate the design of the physical architecture layer.

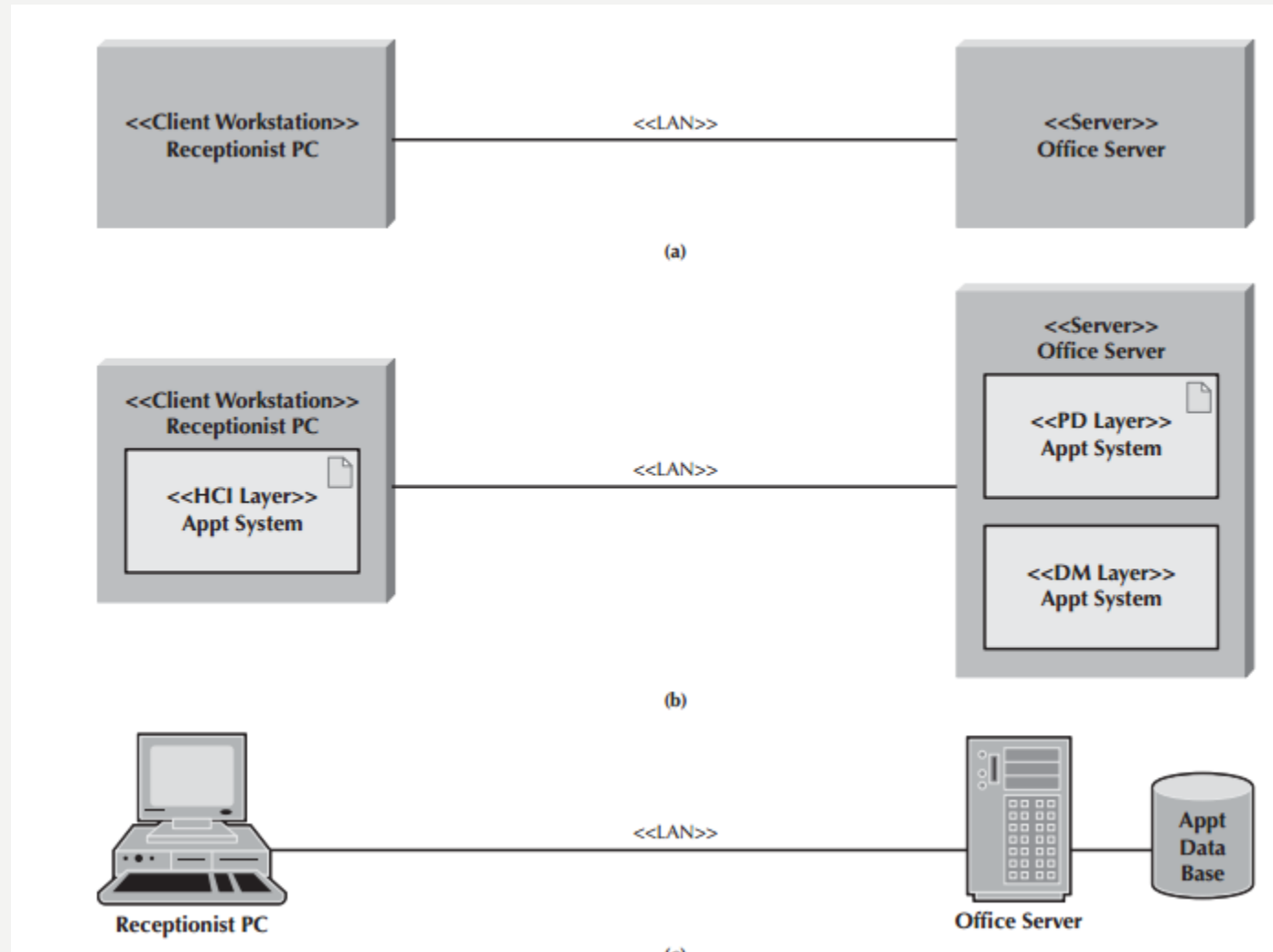
Component of Deployment Diagram

<p>A node:</p> <ul style="list-style-type: none"> ■ Is a computational resource, e.g., a client computer, server, separate network, or individual network device. ■ Is labeled by its name. ■ May contain a stereotype to specifically label the type of node being represented, e.g., device, client workstation, application server, mobile device, etc. 	
<p>An artifact:</p> <ul style="list-style-type: none"> ■ Is a specification of a piece of software or database, e.g., a database or a table or view of a database, a software component or layer. ■ Is labeled by its name. ■ May contain a stereotype to specifically label the type of artifact, e.g., source file, database table, executable file, etc. 	
<p>A node with a deployed artifact:</p> <ul style="list-style-type: none"> ■ Portrays an artifact being placed on a physical node. 	
<p>A communication path:</p> <ul style="list-style-type: none"> ■ Represents an association between two nodes. ■ Allows nodes to exchange messages. ■ May contain a stereotype to specifically label the type of communication path being represented, (e.g., LAN, Internet, serial, parallel). 	

Extended Component of Deployment Diagram for Node Representation



Example of Deployment Diagram



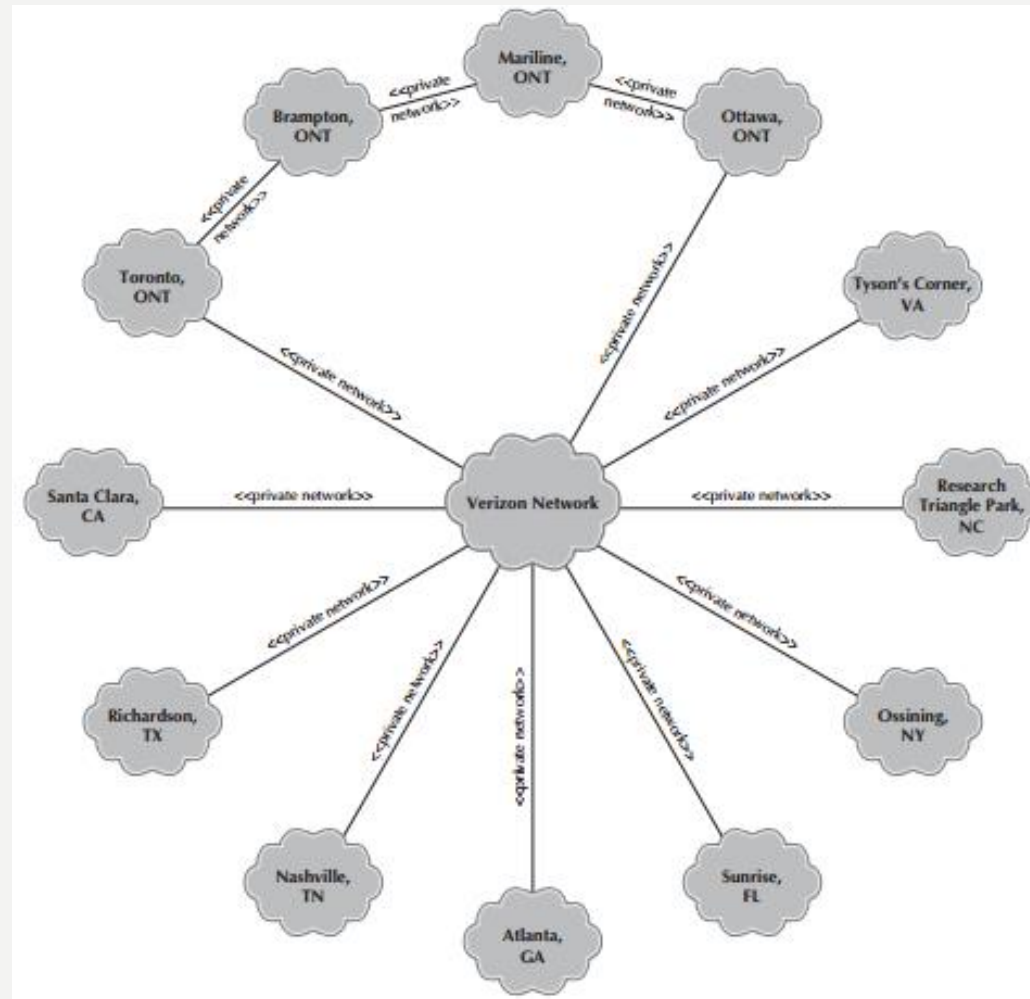
Network Model Diagram

- The network model is a diagram that shows the major components of the information system (e.g., servers, communication lines, networks) and their geographic locations throughout the organization.
- There is no one way to depict a network model, and in many experience, analysts create their own standards and symbols, using presentation applications (e.g., PowerPoint) or diagramming tools (e.g., Visio).
- In this case, we use UML's deployment diagram.
- Creating the network model is a top-down exercise whereby we first graphically depict all the locations where the application will reside.

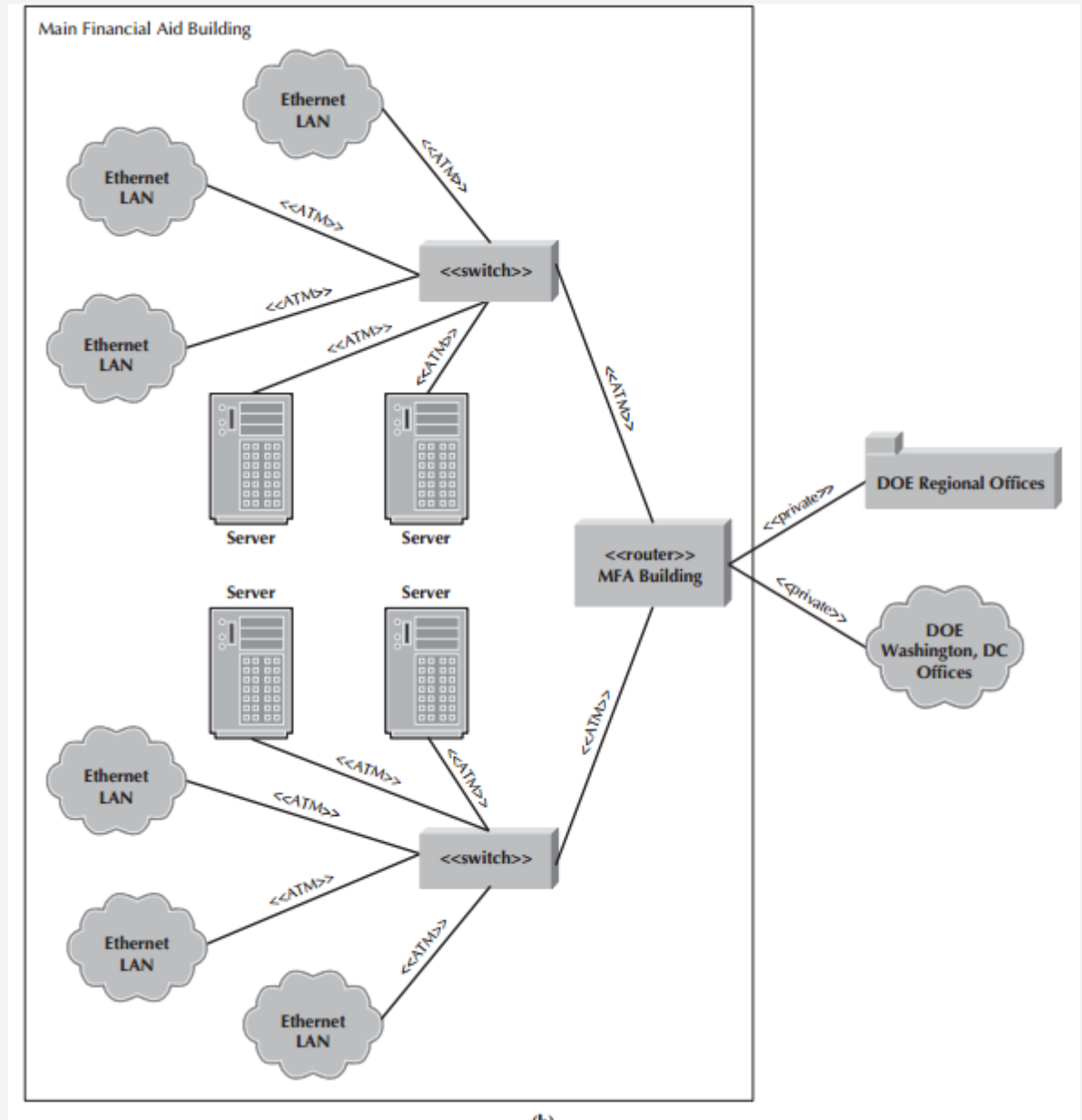
Network Model Diagram

- The purpose of the network model is twofold:
 - to convey the complexity of the system and to show how the system's software components will fit together.
 - the diagram also helps the project team develop the hardware and software specification.
- The components of the network model are
 1. clients (e.g., personal computers, kiosks),
 2. servers (e.g., database, network, communications, printer),
 3. network equipment (e.g., Wi-Fi connections, Ethernet, cell phone network, satellite links), and
 4. external systems or networks (e.g., Internet service providers) that support the application.
- Locations are the geographic sites related to these components.

Example of Network Model Diagram - High level model



Example of Network Model Diagram – Low level model



Hardware and Software Specification

- The hardware and software specification is a document that describes what hardware and software are needed to support an application.
- The actual acquisition of hardware and software should be left to the purchasing department or the area in the organization that handles capital procurement.

Sample of Hardware and Software Specification

Specification	Standard Client	Standard Web Server	Standard Application Server	Standard Database Server
Operating System	<ul style="list-style-type: none">• Windows• Internet Explorer	<ul style="list-style-type: none">• Linux	<ul style="list-style-type: none">• Linux	<ul style="list-style-type: none">• Linux
Special Software	<ul style="list-style-type: none">• Acrobat Reader• Adobe Flash• QuickTime	<ul style="list-style-type: none">• Apache	<ul style="list-style-type: none">• Java	<ul style="list-style-type: none">• Oracle
Hardware	<ul style="list-style-type: none">• 8 GB Memory• 500 GB disk drive• Intel Core i5• 2--22" monitors	<ul style="list-style-type: none">• 16 GB Memory• 1TB disk drive• Intel Xenon E%-2400• 1--22" monitor	<ul style="list-style-type: none">• 32 GB Memory• 2--1 TB disk drives• Intel Xenon E5-2600• 1--22" monitor	<ul style="list-style-type: none">• 32 GB Memory• 4--1 TB Hotplug disk drives• Intel Xenon E5-2600• 1--22" monitor
Network	<ul style="list-style-type: none">• 100 Mbps Ethernet• High-speed Wireless	<ul style="list-style-type: none">• 100 Mbps Ethernet	<ul style="list-style-type: none">• 100 Mbps Ethernet	<ul style="list-style-type: none">• 100 Mbps Ethernet

Creating Hardware & Software Spec

- **First**, we need to **define the software** that will run on each component.
 - **Operating system** (e.g., Windows, Linux) and
 - **Special-purpose software** on the client and servers (e.g., Oracle database).

This document should consider any additional costs, such as technical training, maintenance, extended warranties, and licensing agreements (e.g., a site license for a software package).

Creating Hardware & Software Spec

- **Second**, we must **create a list of the hardware** that is needed to support the future system.
 - With the advent of mobile computing, cloud computing, the IoT, and Green IT, this step is much more involved than it used to be.
 - However, the **low-level network model** provides a good starting point for recording the project's hardware needs because each component on the diagram corresponds to an item on this list.
 - **The list** can include things like database servers, network servers, peripheral devices (e.g., printers, scanners), backup devices, storage components, and any other hardware component that is needed to support an application.
 - At this time, you also should note the **quantity of each item** that will be needed.

Creating Hardware & Software Spec

- **Third**, we must describe, in as much detail as possible, **the minimum requirements** for each piece of hardware. Typically, the project team must convey requirements like the ***amount of processing capacity, the amount of storage space, and any special features that should be included.***

Non-functional Requirements and Physical Architecture Layer Design

- Creating a physical architecture layer design begins with the nonfunctional requirements.
 1. Refine the nonfunctional requirements into more-detailed requirements that are then used to help select the architecture to be used (server-based, client-based, or client–server)
 2. Define what software components will be placed on each device. In a client–server architecture, one also has to decide whether to use a two-tier, three-tier, or n-tier architecture.
 3. Then the nonfunctional requirements and the architecture design are used to develop the hardware and software specification.

Non-functional Requirements and Physical Architecture Layer Design

Four primary types of nonfunctional requirements can be important in designing the architecture:

1. operational requirements
2. performance requirements
3. security requirements
4. cultural/political requirements

Your turn 😊

Discuss four non-functional requirements that might influence physical architecture design:

1. operational requirements
2. performance requirements
3. security requirements
4. cultural/political requirements

Operational Requirements

Type of Requirement	Definition	Examples
Technical Environment Requirements	Special hardware, software, and network requirements imposed by business requirements	<ul style="list-style-type: none">• The system will work over the Web environment with Internet Explorer.• All office locations will have an always-on network connection to enable real-time database updates.• A version of the system will be provided for customers connecting over the Internet via a tablet or smartphone.
System Integration Requirements	The extent to which the system will operate with other systems	<ul style="list-style-type: none">• The system must be able to import and export Excel spreadsheets.• The system will read and write to the main inventory database in the inventory system.
Portability Requirements	The extent to which the system will need to operate in other environments	<ul style="list-style-type: none">• The system must be able to work with different operating systems (e.g., Linux, Mac OS, and Windows).• The system might need to operate with handheld devices, such as Android and Apple iOS devices.
Maintainability Requirements	Expected business changes to which the system should be able to adapt	<ul style="list-style-type: none">• The system will be able to support more than one manufacturing plant with six months' advance notice.• New versions of the system will be released every six months.

Performance Requirements

Type of Requirement	Definition	Examples
Speed Requirements	The time within which the system must perform its functions	<ul style="list-style-type: none">• Response time must be less than 7 seconds for any transaction over the network.• The inventory database must be updated in real time.• Orders will be transmitted to the factory floor every 30 minutes.
Capacity Requirements	The total and peak number of users and the volume of data expected	<ul style="list-style-type: none">• There will be a maximum of 100–200 simultaneous users at peak use times.• A typical transaction will require the transmission of 10K of data.
Availability and Reliability Requirements	The extent to which the system will be available to the users and the permissible failure rate due to errors	<ul style="list-style-type: none">• The system will store data on approximately 5,000 customers for a total of about 2 MB of data.• Scheduled maintenance shall not exceed one 6-hour period each month.• The system shall have 99% uptime performance.

Security Requirements

Type of Requirement	Definition	Examples
System Value Estimates	Estimated business value of the system and its data	<ul style="list-style-type: none">• The system is not mission critical, but a system outage is estimated to cost \$50,000 per hour in lost revenue.• A complete loss of all system data is estimated to cost \$20 million.
Access Control Requirements	Limitations on who can access what data	<ul style="list-style-type: none">• Only department managers will be able to change inventory items within their own department.• Telephone operators will be able to read and create items in the customer file but cannot change or delete items.
Encryption and the Authentication Requirements	Defines what data will be encrypted Where and whether authentication will be needed for user access	<ul style="list-style-type: none">• Data will be encrypted from the user's computer to website to provide secure ordering.• Users logging in from outside the office will be required to authenticate.
Virus Control Requirements	Requirements to control the spread of viruses	<ul style="list-style-type: none">• All uploaded files will be checked for viruses before being saved in the system.

Cultural/Political Requirements

Type of Requirement	Definition	Examples
Customization Requirements	Specification of what aspects of the system can be changed by local users	<ul style="list-style-type: none">• Country managers will be able to define new fields in the product database to capture country-specific information.• Country managers will be able to change the format of the telephone number field in the customer database.
Legal Requirements	The laws and regulations that impose requirements on the system	<ul style="list-style-type: none">• Personal information about customers cannot be transferred out of European Union countries into the United States.• It is against U.S. federal law to divulge information on who rented what videotape, so access to a customer's rental history is permitted only to regional managers.

Designing the Architecture

- Technical environment requirements, driven by business requirements, often define the application architecture
- If not, other nonfunctional requirements become important

Nonfunctional Requirements and their Implications for Architecture Design

Requirements	Server-Based	Client-Based	Thin Client-Server	Thick Client-Server
Operational Requirements				
System Integration Requirements	✓		✓	✓
Portability Requirements			✓	
Maintainability Requirements	✓		✓	
Performance Requirements				
Speed Requirements			✓	✓
Capacity Requirements			✓	✓
Availability/Reliability Requirements	✓		✓	✓
Security Requirements				
High System Value	✓		✓	
Access Control Requirements	✓			
Encryption/Authentication Requirements			✓	✓
Virus Control Requirements	✓			
Cultural/Political Requirements				
Multilingual Requirements			✓	
Customization Requirements			✓	
Making Unstated Norms Explicit			✓	
Legal Requirements	✓		✓	✓