# Fundamentals of Networking

## Fundamental Network Characteristics

# What is Protocols

When two humans converse, they may have to use the same language but they generally understand each other without having to adhere to rigid rules of grammar or formal language frameworks.

Computers, on the other hand, have to have everything explicitly defined and structured. If computers wish to communicate with one another, they have to know in advance exactly how information is to be exchanged and precisely what the format will be. Therefore, standard methods of transmitting and processing various kinds of information are used and these methods are called "protocols".

Protocols are established by international agreement and ensure that computers everywhere can talk to one another. There are a variety of protocols for different kinds of information and functions. This article will discuss some of the common protocols that the average PC user is likely to encounter.

In information technology, a protocol is the special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities.

In order for computers to communicate with one another, standard methods of information transfer and processing have been devised. These are referred to as "protocols" and some of the more common ones such as TCP, IP, UDP, POP, SMTP, HTTP, and FTP are discussed here.

The word protocol comes from the Greek protocollon, meaning a leaf of paper glued to a manuscript volume that describes the contents.

- Protocols exist at several levels in a telecommunication connection. For example, there are protocols for the data interchange at the hardware device level and protocols for data interchange at the application program level.
- In the standard model known as Open Systems Interconnection (OSI), there are one or more protocols at each layer in the telecommunication exchange that both ends of the exchange must recognize and observe.
- Protocols are often described in an industry or international standard.
- Modern protocols for computer networking all generally use packet switching techniques to send and receive messages in the form of packets, messages subdivided into pieces that are collected and re-assembled at their destination.
- Hundreds of different computer network protocols have been developed each designed for specific purposes and environments.

# Purpose of Protocols

Without protocols, devices would lack the ability to understand the electronic signals they send to each other over network connections. Network protocols serve these basic functions:

- Address data to the correct recipient(s)
- Physically transmit data from source to destination, with security protection if needed
- Receive messages and send responses appropriately

Consider a comparison between network protocols with how a postal service handles physical paper mail. Just as the postal service manages letters from many sources and destinations, so too do network protocols keep data flowing along many paths continuously. Unlike physical mail, however, network protocols also provide some advanced capabilities like delivering a constant flow of messages to one destination (called streaming) and automatically making copies of a message and delivering it to multiple destinations at once (called broadcasting).

# Internet Protocols

- **Transmission Control Protocol (TCP)**

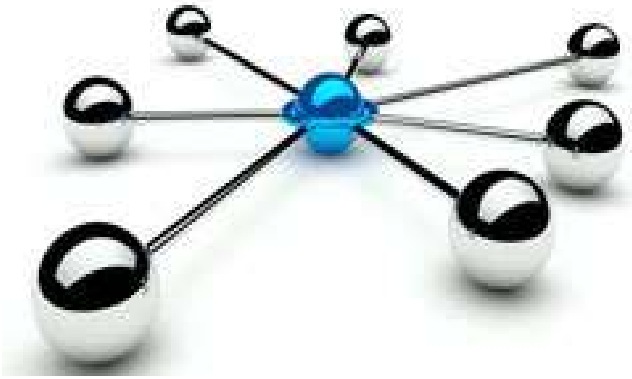
which uses a set of rules to exchange messages with other Internet points at the information packet level

- **Internet Protocol (IP),**

which uses a set of rules to send and receive messages at the Internet address level

- **Hypertext Transfer Protocol (HTTP)**

is used for accessing and receiving Hypertext Markup Language (HTML) files on the internet.



- **Simple Mail Transfer Protocol (SMTP)**

is used for transferring e-mail between computers.

- **File Transfer Protocol (FTP)**

is used for showing files to be copied between devices.

- **PoP3**

is the most common account type for personal e-mail. Messages are typically deleted from the server when you check your e-mail.

- **Internet Message Access Protocol (IMAP)**

is a protocol for e-mail retrieval and storage, IMAP servers let you work with e-mail messages without downloading them to your computer first, developed by Mark Crispin in 1986 at Stanford University as an alternative to POP.

# Wireless Network Protocols

Thanks to Wi-Fi, Bluetooth, GPRS (General Packet Radio Service), EVDO (Evolution Data Optimized, Evolution Data Only), HSDPA (High-Speed Downlink Packet Access) and LTE (Long Term Evolution), wireless networks have become commonplace.

Network protocols designed for use on wireless networks must support roaming mobile devices and deal with issues such as variable data rates and network security.

# Network Routing Protocols

Routing protocols are special-purpose protocols designed specifically for use by network routers on the Internet.

A routing protocol can identify other routers, manage the pathways (called routes) between sources and destinations of network messages, and make dynamic routing decisions.

Common routing protocols include EIGRP, OSPF and BGP.

# How Network Protocols Are Implemented

- Modern operating systems contain built-in software services that implement support for some network protocols.
- Applications like Web browsers contain software libraries that support the high level protocols necessary for that application to function.
- For some lower level TCP/IP and routing protocols, support is implemented in directly hardware (silicon chipsets) for improved performance.
- Each packet transmitted and received over a network contains binary data (ones and zeros that encode the contents of each message).
- Most protocols add a small header at the beginning of each packet to store information about the message's sender and its intended destination.
- Some protocols also add a footer at the end.  Each network protocol has the ability to identify messages of its own kind and process the headers and footers as part of moving data among devices.

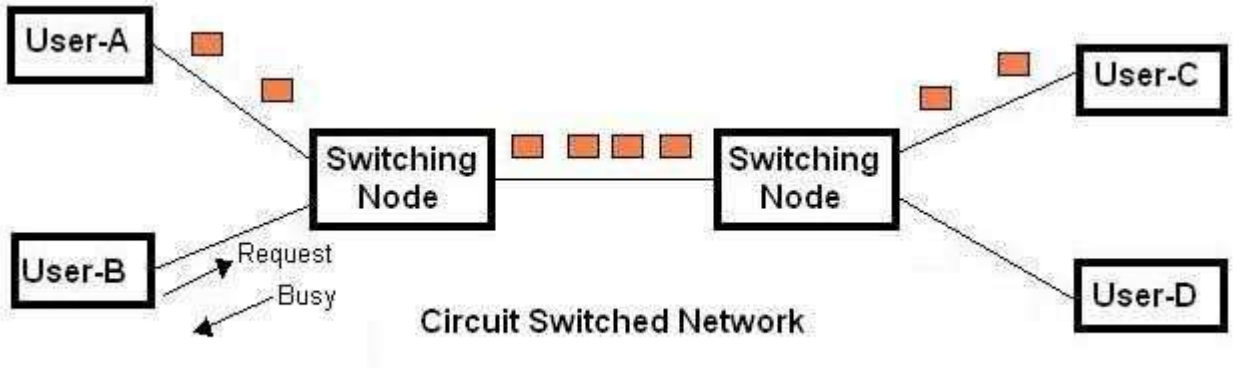
# How Network Devices Use Protocols

- The operating systems of network devices include built-in support for some lower level network protocols.
- All modern desktop computer operating systems support both Ethernet and TCP/IP, for example, while many smartphones support Bluetooth and protocols from the Wi-Fi family.
- These protocols ultimately connect to the physical network interfaces of a device, like its Ethernet ports and Wi-Fi or Bluetooth radios.
- Network applications, in turn, support the higher level protocols which talk to the operating system.
- A Web browser, for example, is capable of translating addresses like *http://google.com/* into HTTP packets that contain the necessary data that a Web server can receive and in turn send back the correct Web page.
- The receiving device is responsible for re-assembling individual packets into the original message, by stripping off the headers and footers and concatenating packets in the correct sequence.

# Circuit Switching and Packet Switching Networks

# Circuit Switching

In circuit switching network dedicated channel has to be established before the call is made between users. The channel is reserved between the users till the connection is active. For half duplex communication, one channel is allocated and for full duplex communication, two channels are allocated. It is mainly used for voice communication requiring real time services without any much delay.



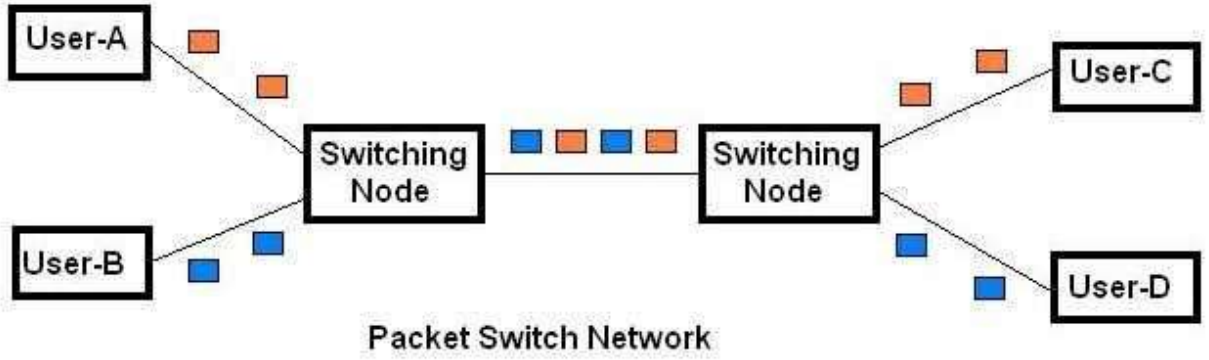

As shown in the figure, if user-A wants to use the network; it need to first ask for the request to obtain the one and then user-A can communicate with user-C. During the connection phase if user-B tries to call/communicate with user-D or any other user it will get busy signal from the network.

# Example of Circuit Switching

You pick up your land phone and dial your friend. At that point, the telecom provider creates a dedicated circuit for that session and connects you to your friend's telephone. No matter how long you keep the line open with your friend, the circuit will remain, and packets flowing between both telephones will always follow the same path. This is an example of a circuit-switched network.

# Packet Switching

In packet switching network unlike CS network, it is not required to establish the connection initially. The connection/channel is available to use by many users. But when capacity or number of users increases then it will lead to congestion in the network. Packet switched networks are mainly used for data and voice applications requiring non-real time scenarios.



Packet Switch Network

As shown in the figure, if user-A wants to send data/information to user-C and if user-B wants to send data to user-D, it is simultaneously possible. Here information is padded with header which contains addresses of source and destination. This header is sniffed by intermediate switching nodes to determine their route and destination.

# Packet Switching

- In packet switching, station breaks long message into packets. Packets are sent one at a time to the network. Packets are handled in two ways, viz. datagram and virtual circuit.
- **In datagram**, each packet is treated independently. Packets can take up any practical route. Packets may arrive out of order and may go missing.
- **In virtual circuit**, preplanned route is established before any packets are transmitted. The handshake is established using call request and call accept messages. Here each packet contains virtual circuit identifier(VCI) instead of the destination address. In this type, routing decisions for each packet are not needed.

# Example of Packet Switching

you switch on your PC and connect to your favorite site that offers a number of applications you can download from, so you begin downloading one application at a time. Each packet has to find its own route to the destination, i.e., your computer. Each packet finds its way using the information it carries, such as the source and destination IP address. If network congestion occurs, the routers responsible for routing packets between networks will automatically select different paths to ensure data is transferred as required. This is an example of a packet-switched network.

# Comparison between CS vs. PS networks

Packet switched (PS) networks quality of service (QoS) is not guaranteed while in circuit switched (CS) networks quality is guaranteed.

PS is used for time insensitive applications such as internet/email/SMS/MMS/VOIP) etc.

In CS even if user is not talking the channel cannot be used by any other users, this will waste the resource capacity at those intervals.

The example of circuit switched network is PSTN (Public Switched Telephone Network) and example of packet switched network is GPRS (General Packet Radio Service)/EDGE (Enhanced Data for Global Evolution).

# Comparison between CS vs. PS networks

| Circuit Switching | Packet Switching(Datagram type) | Packet Switching(Virtual Circuit type) |
|---|---|---|
| Dedicated path | No Dedicated path | No Dedicated path |
| Path is established for entire conversation | Route is established for each packet | Route is established for entire conversation |
| Call setup delay | packet transmission delay | call setup delay as well as packet transmission delay |
| Overload may block call setup | Overload increases packet delay | Overload may block call setup and increases packet delay |
| Fixed bandwidth | Dynamic bandwidth | Dynamic bandwidth |
| No overhead bits after call setup | overhead bits in each packet | overhead bits in each packet |

# Client / Server Networking

# &

# Peer to Peer Networking

# Introduction

# Introduction

- **Communication Network**

A system of interconnected end systems, intermediate systems, and other equipment allowing information to be exchanged. A network can be of any size, from 2 to 1000's devices

- **Intermediate system**

A device that operates as a relay element between 2 or more end systems (networks)

E.g., a repeater, a hub, a bridge, a switch, a router

- **End system**

A device that uses or provides end-user applications or network services

They are labeled "end systems" because they sit at the edge of a network. End systems that are connected to the Internet are also referred to as "hosts"; this is because they host (run) Internet applications

E.g., a Desktop PC, a Web server, a DNS server

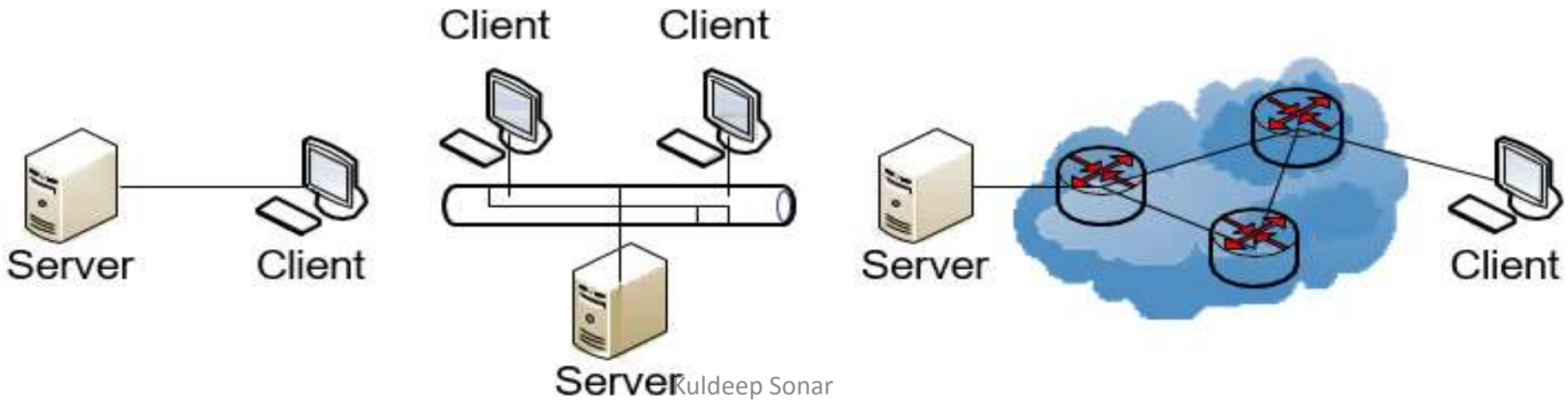
# Introduction

End systems can be positioned on a network in different ways relative to each other I.e., they can be made to communicate and share resources according to different interaction models

2 fundamental interaction models :

- Client/server
- Peer-to-peer (P2P)

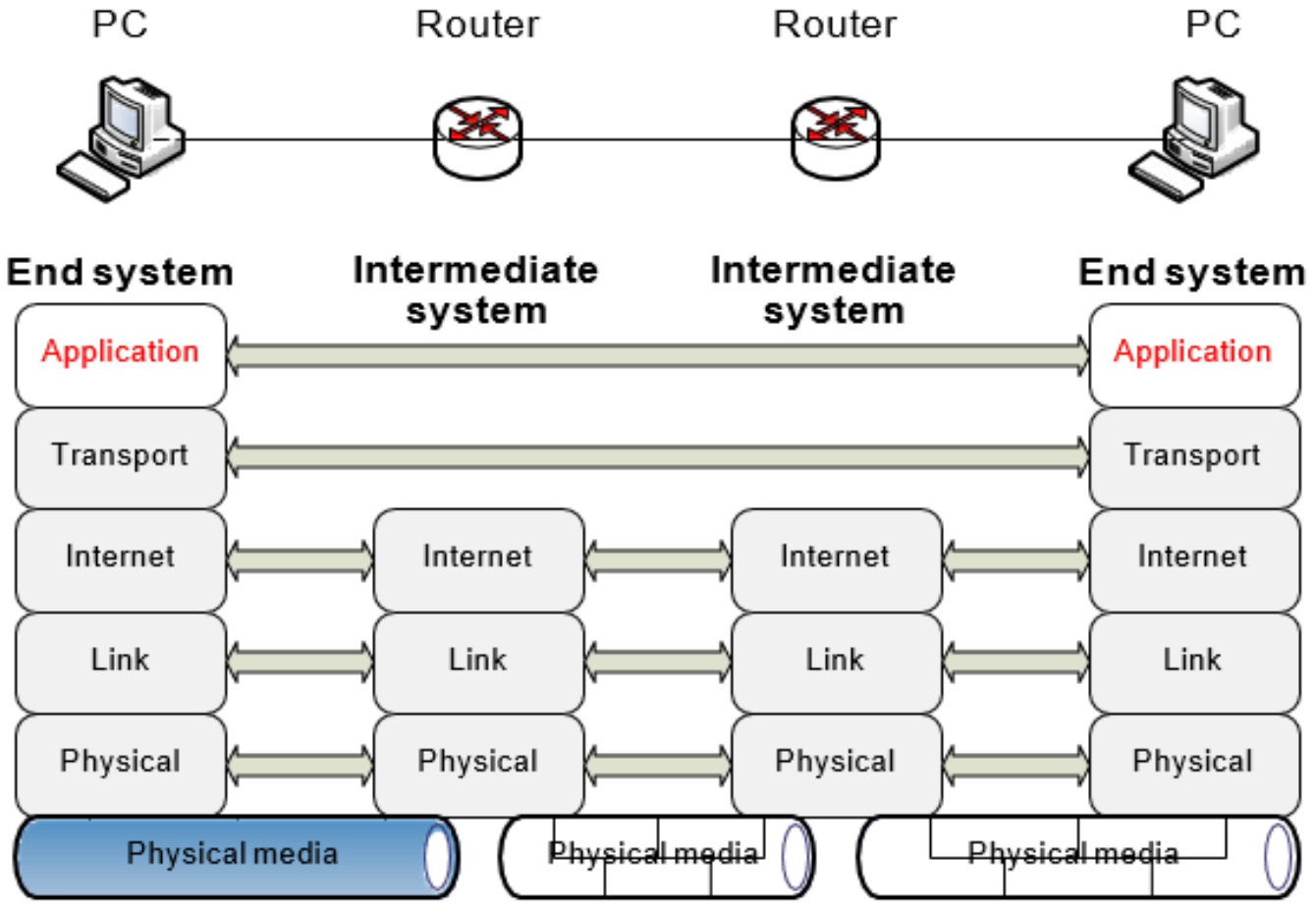
These models are relevant to end systems only, regardless of how the end systems are connected to each other



# Introduction

Client/server and P2P protocols operate at the application layer of the TCP/IP model

Application layer protocols are end-to-end protocols

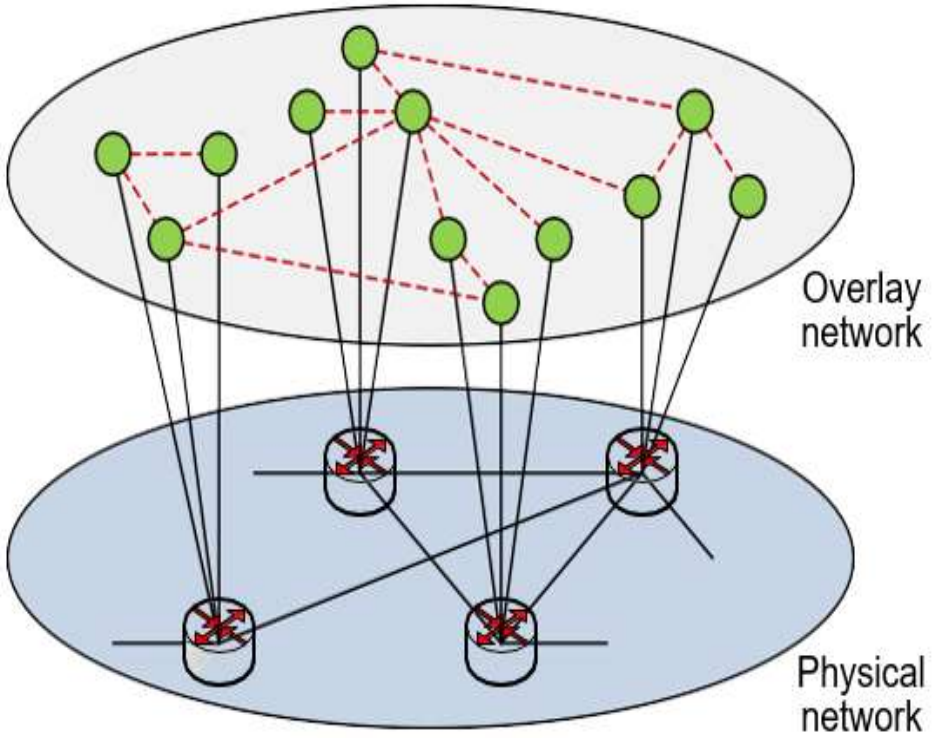


# Introduction

Client/server and P2P systems are implemented as virtual networks of nodes and logical links built on top of an existing (aka underlay) network, typically the Internet.

These virtual networks are called overlay networks.

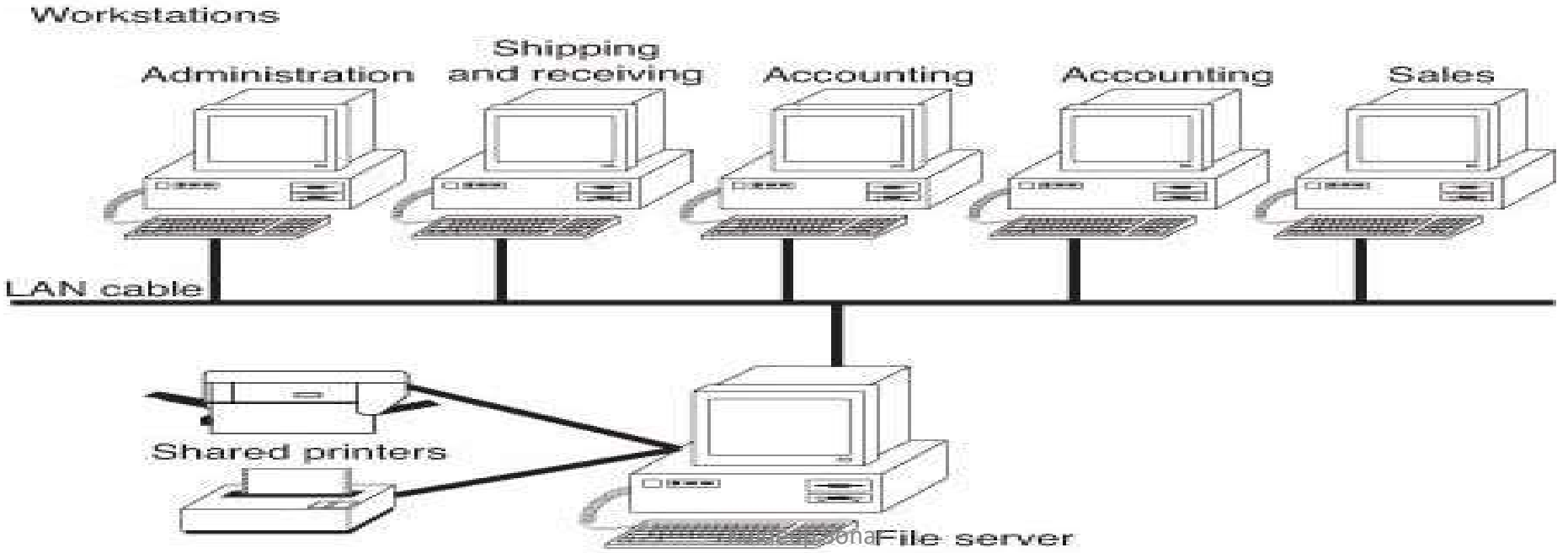
The overlay is a logical view that might not directly mirror the physical network topology



# Client / Server Networking

# Client / Server Networking

On a client/server network, every computer has a distinct role: that of either a client or a server. A server is designed to share its resources among the client computers on the network. Typically, servers are located in secured areas, such as locked closets or data centers (server rooms), because they hold an organization's most valuable data and do not have to be accessed by operators on a continuous basis.



# Client / Server Networking

A **dedicated server** computer often has faster processors, more memory, and more storage space than a client because it might have to service dozens or even hundreds of users at the same time.

High-performance servers typically use from two to eight processors (and that's not counting multi-core CPUs), have many gigabytes of memory installed, and have one or more server-optimized **network interface cards (NICs)**, RAID (Redundant Array of Independent Drives) storage consisting of multiple drives, and redundant power supplies. Servers often run a special network OS—such as Windows Server, Linux, or UNIX—that is designed solely to facilitate the sharing of its resources.

These resources can reside on a single server or on a group of servers. When more than one server is used, each server can "specialize" in a particular task (file server, print server, fax server, email server, and so on) or provide redundancy (duplicate servers) in case of server failure.

In the client/server model, all end systems are divided into clients and servers each designed for specific purposes

**Clients** have an active role and initiate a communication session by sending requests to servers.
Clients must have knowledge of the available servers and the services they provide.
Clients can communicate with servers only; they cannot see each other

**Servers** have a **passive** role and respond to their clients by acting on each request and returning results.
One server generally supports numerous clients

# Hardware Roles

The terms ”client” and ”server” usually refer to the primary roles played by networked hardware

A ”client” is usually something like a PC used by an individual, and primarily initiates conversations by sending requests

A ”server” is usually a powerful machine dedicated to responding to client requests, sitting in a server room somewhere that nobody but its administrator ever sees

# Software Roles

TCP/IP uses different pieces of software for many protocols to implement ”client” and ”server” roles
Client software is usually found on client hardware and server software on server hardware, but not always
Some devices may run both client and server software

**Web clients**: Mozilla Firefox, Internet Explorer, Google Chrome, . . .

**Web servers**: Apache, Microsoft IIS, GWS, . . .

# Transactional Roles

In any exchange of information, the client is the entity that initiates communication or sends a query; the server responds, usually providing information.

Again, usually client software on a client device initiates a transaction, but this is not always the case

E.g., when 2 SMTP servers communicate to exchange electronic mail, even though they are both server programs running on server hardware, during any transaction one device acts as client, while the other acts as server

# Servers

The purpose of servers is to provide some predefined services for clients

**2 types of servers** :

**Iterative servers** iterate through the following steps:

Wait for a client request to arrive
Process the request and send the response back to the client Go back to Step 1

Thus, iterative servers handle clients **sequentially**, finishing with one client before servicing the next

**Concurrent servers** perform the following steps:

Wait for a client request to arrive
Use a new process/task/thread to handle the request Go back to Step 1

Thus, concurrent servers handle client requests **in parallel**

# Iterative or Concurrent?

**Iterative design** is quite simple and is most suitable for short-duration services that exhibit relatively little variation in their execution time

- I.e., if the time to handle a client can be long, the waiting time experienced by subsequent clients may be unacceptable
- Internet services like **echo** (RFC 862) and **daytime** (RFC 867) are commonly implemented as iterative servers

**Concurrent design** is more complex but yields better performance

- It allows to improve responsiveness and reduce latency when the rate at which requests are processed is less than the rate at which requests arrive at the server
- Internet services like **HTTP**, **telnet**, and **FTP** are commonly implemented as concurrent servers

As a rule, TCP-based servers are concurrent and UDP-based servers are iterative

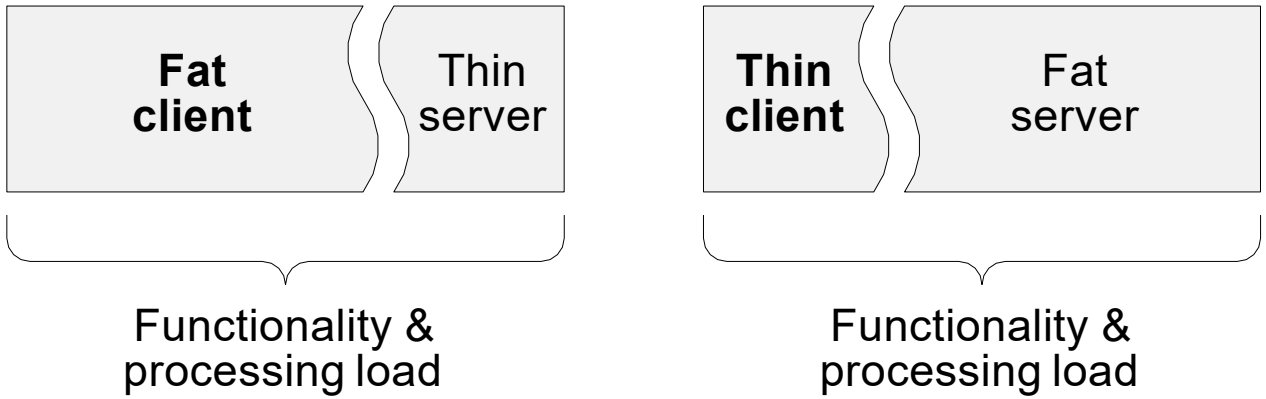
# Clients

Clients are devices/programs that request services from servers Clients (and, hence, servers) can be distinguished according to the

**functionality** they provide and the amount of **processing load** they carry

**2 types of clients** :

Fat (aka thick or full) Thin (aka slim or lean)



**Fat clients** are devices/programs that are powerful enough and operate with limited dependence on their server counterparts

**Fat clients as devices** – a user workstation that is powerful and fully-featured in its own right

E.g., a desktop PC, a laptop, a netbook

**Fat clients as programs** – a client carries a relatively large proportion of the processing load

E.g., the Lineage II gaming client (more than 2 GB in size)

**Thin clients** are devices/programs that have very limited functionality and depend heavily on their server counterparts

**Thin clients as devices** – a user workstation that contains a minimal operating system and little or no data storage

E.g., Sun Ray thin clients in Lintula, room TC215

**Thin clients as programs** – a client mainly provides a user interface, while the bulk of processing occurs in the server

E.g., the OnLive gaming client (about 10 MB in size)

# Fat or Thin?

Both technologies have their positive and negative aspects

**Benefits** of thin-client systems:

No viruses, spyware, spam, thefts, etc.

Easy to keep the software properly configured and patched Lower TCO (Total Cost of ownership)

Fewer points of failure

**Shortcomings** of thin-client systems:

Servers are the central point of failure Systems tend to be proprietary

Multimedia-rich applications require a significant amount of bandwidth to function to their maximum potential

Today, there are a lot of factors driving IT towards thin clients:

Desktop virtualization and

Increase in Web-based applications

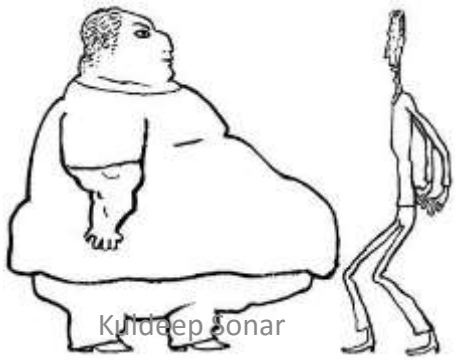
High-bandwidth LAN and WAN connections

Data security concerns

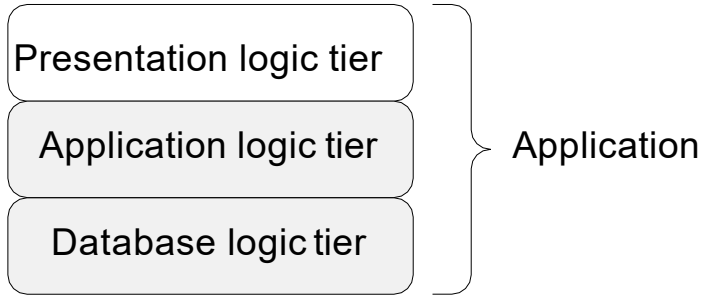
Energy savings

Advances in low-cost computers

***<u>"Thin clients can reduce TCO but the trick is using them for the right people and applications."</u>***



# Logical Tiers




In general, application software can be divided into

- Presentation logic
- Application logic (aka business rules)
- Database logic

Each tier in the software is responsible for a specific task in the application, This layering is a reference model; in practice, the boundaries may be not so sharp

The logical layering of application software does not need to be the same as its physical layering

**Presentation logic**

is responsible for displaying the information and interacting with the user (i.e., user interface)

It must make the information available in a suitable form to the user and must respond appropriately to input from the user

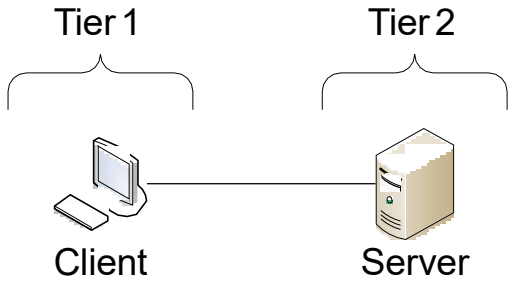
**Application logic** processes commands, makes logical decisions, performs calculations, and coordinates the application

It also moves and processes data between the presentation logic and database logic tiers

**Database logic** refers to the management of underlying databases It is responsible for storing and retrieving then data according to the requirements of the application logic tier. A static content is typically stored in a **file system**

# Physical Tiers

**1-tier architecture** is used to describe systems in which all of the processing is done on a single host Users can access such systems (aka **mainframes**) through display terminals (aka **dumb terminals**) but what is displayed and how it appears is controlled by the mainframe



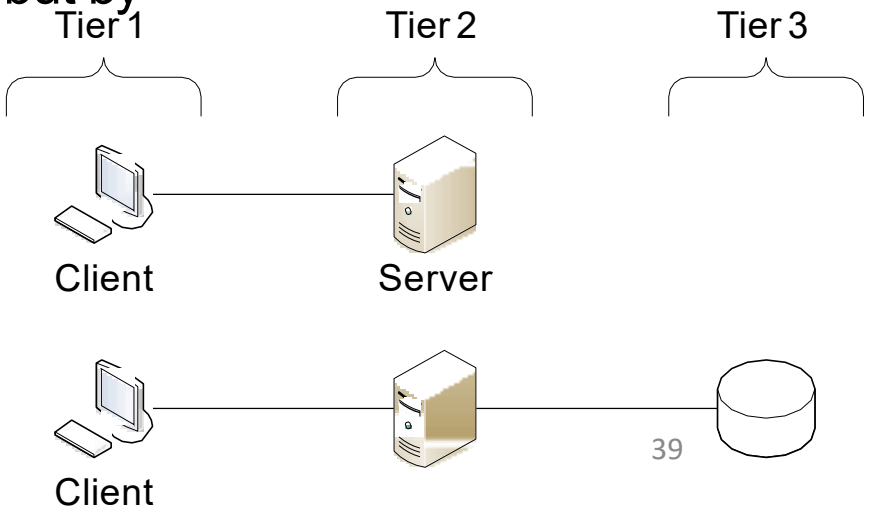

**2-tier architecture** (aka **flat**) is used to describe client/server systems, where clients request resources and servers respond directly to these requests, using there own resources

**3-tier architecture** is used to describe client/server systems consisting of:

**Clients** which request services

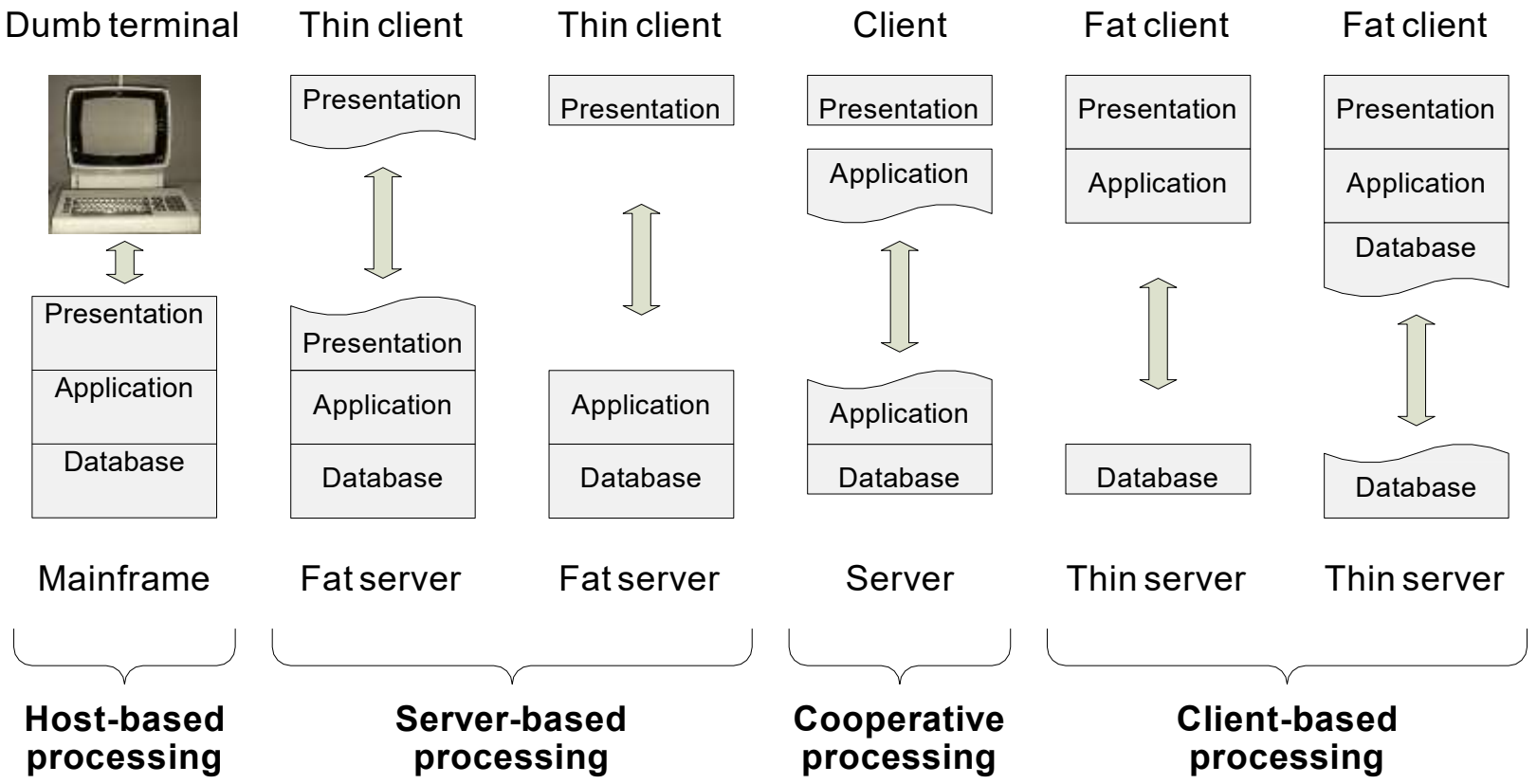
**Application servers** whose task is to provide the requested resources, but by calling on database servers

**Database servers** which provide the application servers with the data they require



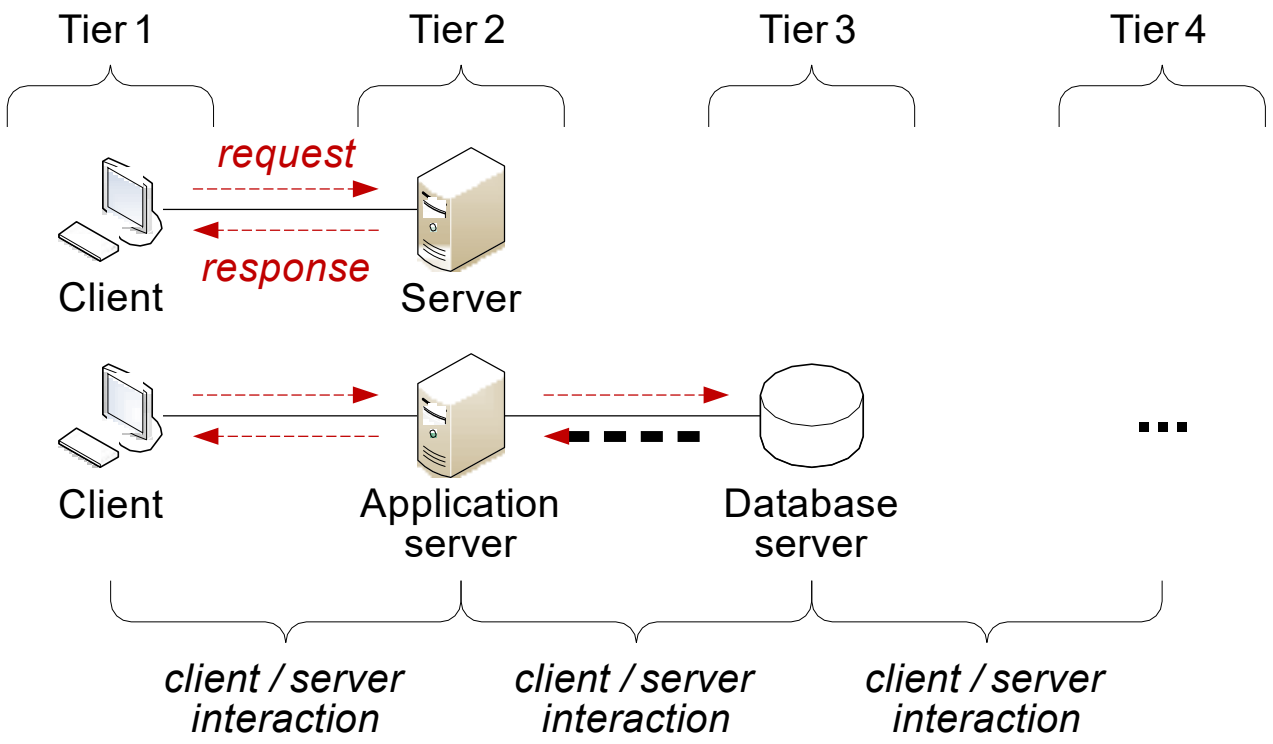
# Physical Tiers

Distributing logical tiers between 2 physical tiers



## N-tier architecture

is used to describe client/server systems consisting of more than 3 tiers



**Benefits** of 3- and N-tier (aka **multi-tier** or **hierarchical**) architectures:

Increased flexibility, as changes to the presentation/application/database logic are largely independent from one another

Increased security, as security can be defined for each service and at each tier

Increased performance, as tasks are shared between tiers

**Shortcomings** of multi-tier architectures:

Increased complexity of development and testing Increased need for load balancing and fault tolerance

# Peer to Peer Networking

# Peer to Peer Networking

On a **peer-to-peer network**, every computer is equal and can communicate with any other computer on the network to which it has been granted access rights.

Essentially, every computer on a peer-to-peer network can function as both a server and a client; any computer on a peer-to-peer network is considered a server if it shares a printer, a folder, a drive, or some other resource with the rest of the network.

This is why you might hear about client and server activities, even when the discussion is about a peer-to-peer network.

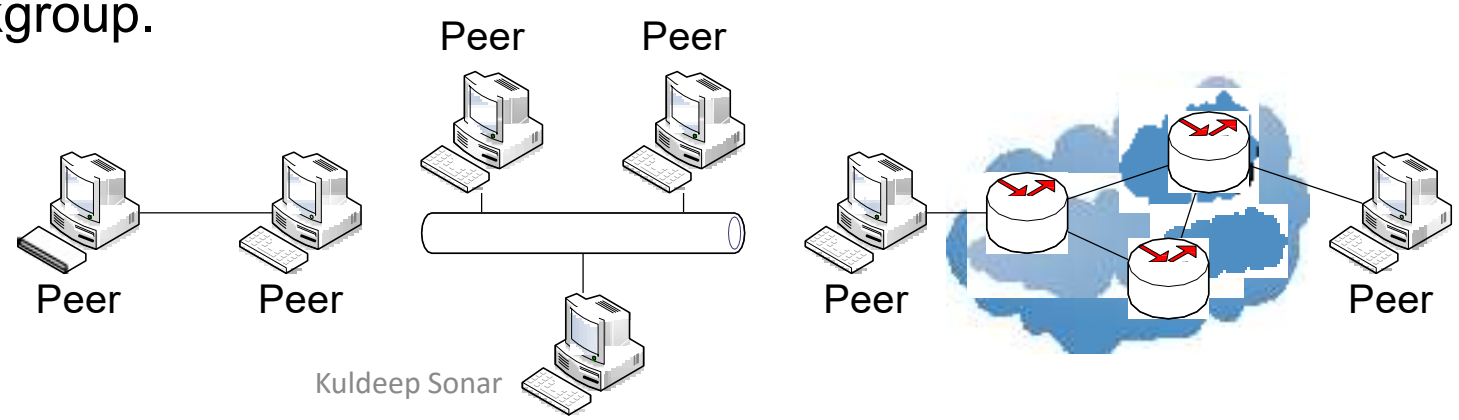
Peer-to-peer networks can be as small as two computers or as large as

hundreds of systems and devices.

Although there is no theoretical limit to the size of a peer-to-peer network, performance, security, and access become a major headache on peer-based networks as the number of computers increases.

# Peer to Peer Networking

- In a peer-to-peer network, a group of computers is connected together so that users can share resources and information.
- There is no central location for authenticating users, storing files, or accessing resources.
- This means that users must remember which computers in the workgroup have the shared resource or information that they want to access.
- It also means that users must log on to each computer to access the shared resources on that computer.
- In most peer-to-peer networks, it is difficult for users to track where information is located because data is generally stored on multiple computers.
- This makes it difficult to back up critical business information, and it often results in small businesses not completing backups. Often, there are multiple versions of the same file on different computers in the workgroup.



# P2P Model

In the P2P model, all end systems have equivalent capabilities and responsibilities and either party can initiate a communication session

The participants share a part of their own hardware resources

E.g., storage capacity, link capacity, CPU power

These shared resources are necessary to provide the service or content offered by the P2P network

Thus, the participants are **both resource providers and resource requestors** and use similar networking programs to connect with each other

In P2P networks, downlink and uplink data flows tend to be (but not necessarily) **symmetric,** This is because each connected host simultaneously operates as both client and server, thus receiving and transmitting on average the same amount of data

The P2P model does not have the notion of clients or servers but only equal **peers** (aka **servents**, servent = SERVer + cliENT) that simultaneously function as both clients and servers, But for any communication session we can still distinguish between requesting peers as ”clients” and responding peers as ”servers”. Again, **this model is relevant to end systems only**, regardless of how the end systems are connected to each other

**Benefits** of P2P:

No need for dedicated application and database servers  Improved scalability and reliability (no single point of  failure)

**Shortcomings** of P2P:

Poor security
Lack of centralized control
Computers with shared resources may suffer from sluggish performance

**P2P networking allows easily to share and download copyrighted files, Is it a benefit or a shortcoming?*

# Pure P2P

In P2P systems, cooperative peers self-organize themselves into overlay networks and store or relay data for each other

**2 types of P2P systems** :

Pure

Hybrid

**Pure P2P system** – a P2P system that has **no central service** of any kind

I.e., the entire communication occurs among connected peers without any assistance from any server

**Pure P2P systems represent the reference type of P2P design**

Examples of pure P2P systems:

Workgroups in Microsoft Windows Network

Gnutella v0.4

Freenet

# Hybrid P2P

The major challenge of P2P systems is how to achieve efficient resource search
Pure P2P systems work well only in a small-scale environment

**Hybrid P2P system** – a P2P system which depends partially on central servers or allocates selected functions to a subset of dedicated peers

**Central servers** act as central directories where either connected users or indexed content can be mapped to the current location
**Dedicated peers** direct control information among other peers

Thus, **client/server and P2P systems are not mutually exclusive**

Examples of hybrid P2P systems:

Usenet
Napster
Gnutella
eDonkey
BitTorrent

# Advantages of a Peer-to-Peer Network

The peer-to-peer network is the easiest type of network to set up. It does not require any software other than the operating system already on your PC, and it does not require the more complex configuration of a client/server network.

- **Simplicity**

A peer-to-peer network is so basic that you don't need anything more than your PC's existing software, a couple of NICs, and some cable. In a wireless network, all the hardware you will need is two wireless NICs.

- **Peer-to-Peer Is Supported in Windows**

Because most personal computers in homes today have some form of Windows, it is very easy to set up a home network..

- **Low Cost**

The cost to build a home network using peer-to-peer technology is lower than that for a client/server network because you do not need any special software or computer.

- **New Technologies for Home Networking Favor Peer-to-Peer**

Network Wiring and Hardware Options such as phone line, powerline, and wireless networking, are built with the understanding that you will likely build a peer-to-peer network. This isn't to say that you can't build a client/server network with these technologies, but with these "no-new-wires" solutions, a peer-to-peer network is extremely simple.

- **What's Mine Is Yours**

A peer-to-peer network allows each PC on the network to access resources on all the other PCs on the network. That DVD on Dad's PC, the laser printer downstairs on Mom's, the PC camera on Billy's PC—they're all part of the network community after you create a peer-to-peer network.

# Disadvantages of a Peer-to-Peer Network

- **Security**

If a PC is on a peer-to-peer network, there is the chance that another PC on the home network will access files that the owner of the PC might not want accessed. Not that this will necessarily happen or that you have something to hide from other users (or do you?), but this is something to think about. However, Windows provides the capability to block access to certain drives, so this shouldn't be a worry for anyone who properly configures her network software.

- **PCs Down on the Network Can Cause the Network to Go Down**

In some instances, a PC might not be working, either because of problems or because it has simply been shut off. In a basic peer-to-peer network that uses direct connections from PC to PC without a hub, this might cause a problem.

- **Network Speed**

In a peer-to-peer network situation in which a hub or switch is not used, it becomes a real possibility that the network can get bogged down when more than one user is using it at the same time.

- **Scalability**

A peer-to-peer network is great for a home network with a handful of users such as three to five PCs. However, if one day you decide to go into business with Brad, your neighbor, and set up a network in your home, when you start to increase the number of users to 10 or beyond, you might want to consider moving to a client/server network.

# Comparison Between Client/ Server & P2P Network

| Item | Client/Server | Peer-to-Peer |
|---|---|---|
| Access control | Via user/group lists of permissions Via user/group lists of permissions to only the resources granted, and different users can be given different levels of access. | Resources are managed by each system with shared resources. Depending on the OS, resources may becontrolled by separate passwords for each shared resource or by a user list stored on each system with shared resources. Some OSs do not use passwords or user/group lists, thus enabling access to shared resources for anyone accessing the network. |
| Security | High; access is controlled by user or by group identity. | Varies; if password protection is employed, anyone who knows the password can access a shared resource. If no passwords are used, anyone who can access the workgroup can access shared resources. However, if user/group names are used, security is comparable to a client/server network. |
| Performance | High; the server is dedicated and doesn't handle other tasks. | Low; servers often act as workstations. |
| Hardware Cost | High; specialized high-performance server hardware with redundancy features. | Low; any workstation can become a server by sharing resources. |
| Software Cost | Higher; license fees per user are part of the cost of the server OS. | Lower; client software is included with OS. |
| Backup | Centralized on the server; managed by network administrator. Backup by device and media only required at server. | Decentralized; managed by users. Backup devices and media are required at each workstation. |
| Redundancy | Yes; duplicate power supplies, hot-swappable drive arrays, and even redundant servers are common; network OS normally is capable of using redundant devices automatically. | No true redundancy among peer "servers" or clients; failures require manual intervention to correct, with a high possibility of data loss. |

# Taxonomy of Computer Systems Architectures