Module 1

Introduction to Computer Networks

**Definition:** A computer network consists of a collection of computers, printers and other equipment that is connected so that they can communicate with each other.

Today’s computer networks are increasingly taking over the functions previously performed by single-use networks. This chapter looks at some typical applications of computer networks and discusses the requirements that a network designer who wishes to support such applications must be aware of.

This chapter does four things. First, it explores the requirements that different applications and different communities of people place on the network. Second, it introduces the idea of a network architecture, which lays the foundation for the rest of the book. Third, it introduces some of the key elements in the implementation of computer networks. Finally, it identifies the key metrics that are used to evaluate the performance of computer networks.

Applications of Computer Networks

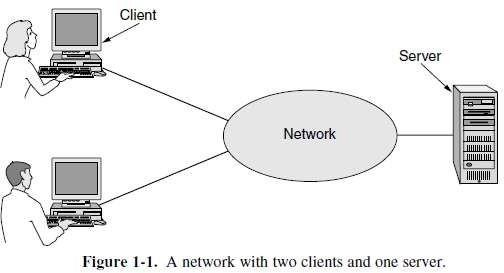
**1.1 APPLICATIONS OF COMPUTER NETWORKS**

Business Applications Home Applications

Most people know the Internet through its applications: the World Wide Web, email, online social networking, streaming audio and video, instant messaging, file-sharing, to name just a few examples. The Applications of computer networks are classified into business applications and home applications.

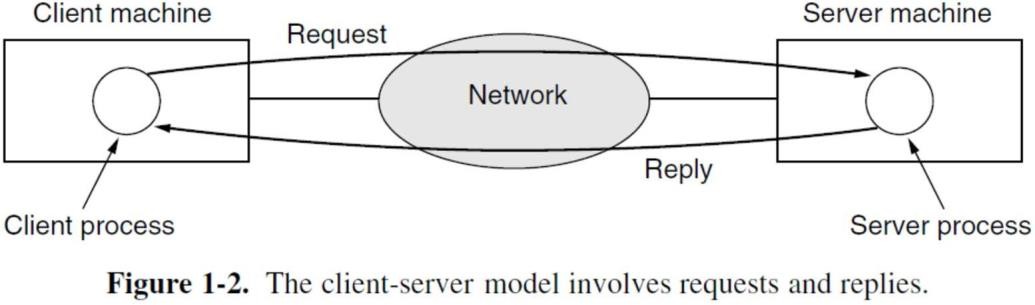
## Business Applications:

* 1. **Resource sharing**. The goal is to make all programs, equipment and especially data available to anyone on the network without regard to the physical location of the resource or the user.
  2. **VPNs** (**Virtual Private Networks**) may be used to join the individual networks at different sites into one extended network. In larger ones, the computers and employees may be scattered over dozens of offices and plants in many countries. Nevertheless, a sales person in New York might sometimes need access to a product inventory database in Singapore.
  3. The **client-server** model, the data are stored on powerful computers called **servers**. Often these are centrally housed and maintained by a system administrator. In contrast, the

employees have simpler machines, called **clients**, on their desks, with which they access remote data. The client and server machines are connected by a network, as illustrated in Fig. 1-1. The client-server model is

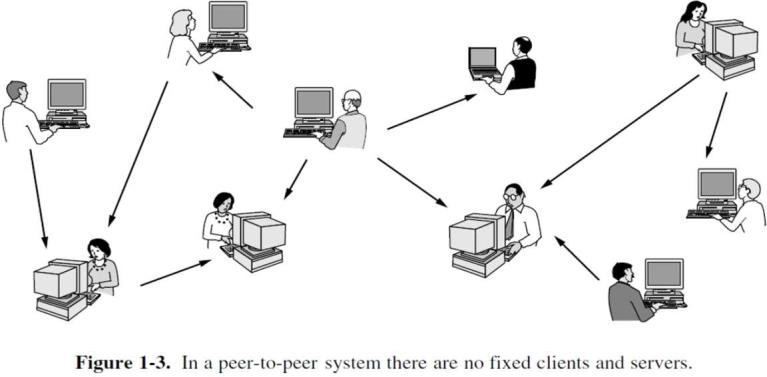
applicable when the client and server are both in the same building.

* 1. The most popular realization is that of a **Web application**, in which the server generates Web pages based on its database in response to client requests. If we look at the client- server model in detail, we see that two processes (i.e., running programs) are involved, one on the client machine and one on the server machine. Communication takes the form of the client process sending a message over the network to the server process. The client process then waits for a reply message. When the server process gets the request, it performs the requested work or looks up the requested data and sends back a reply.



* 1. A computer network can provide a powerful **communication medium** among employees. Virtually every company that has two or more computers now has **email** (**electronic mail**), which employees generally use for a great deal of daily communication.
  2. Telephone calls between employees may be carried by the computer network instead of by the phone company. This technology is called **IP telephony** or **Voice over IP** (**VoIP**) when internet technology is used. The microphone and speaker at each end may belong to a VoIP- enabled phone or the employee’s computer. Companies find this a wonderful way to save on their telephone bills.
  3. **Videoconference** can be one of the important business applications so that employees at distant locations can see and hear each other as they hold a meeting. This technique is a powerful tool for eliminating the cost and time previously devoted to travel. **Desktop sharing** lets remote workers see and interact with a graphical computer screen.
  4. A third goal for many companies is doing business electronically, especially with customers and suppliers. This new model is called **e-commerce** (**electronic commerce**) and it has grown rapidly in recent years. Airlines, bookstores and other retailers have discovered that many customers like the convenience of shopping from home.

## B. Home Applications

1. Internet access provides home users with **connectivity** to remote computers. As with companies, home users can access information, communicate with other people and buy products and services with e-commerce. Access to remote information comes in many forms. It can be surfing the **World Wide Web** for information or just for fun. Information available includes the arts, business, cooking, government, health, history, hobbies, recreation, science, sports, travel, and many others.
2. Many newspapers have gone online and can be personalized. The next step beyond newspapers (plus magazines and scientific journals) is the online digital library. Many professional organizations, such as the ACM (*www.acm.org*) and the IEEE Computer Society (*www.computer.org*), already have all their journals and conference proceedings online. Electronic book readers and **online libraries** may make printed books obsolete.
3. **peer-to-peer** communication. In this form, individuals who form a loose group can communicate with others in the

group, as shown in Fig. 1-3. Every person can, in principle, communicate with one or more other people; there is no fixed division into clients and servers.

Many peer-to-peer systems, such BitTorrent, do not have any central database of content. Instead, each user maintains his own database locally and provides a list of other nearby people who are members of the system. A new user can then go to any existing member to see what he has and get the names of other members to inspect for more content and more names. This lookup process can be repeated indefinitely to build up a large local database of what is out there. Peer-to-peer communication is often used to share music and videos.

1. Our fourth category is entertainment. This has made huge strides in the home in recent years, with the distribution of music, radio and television programs, and movies over the Internet beginning to rival that of traditional mechanisms. Users can find, buy, and download MP3 songs and DVD-quality movies and add them to their personal collection. TV shows now reach many homes via **IPTV** (**IP TeleVision**) systems that are based on IP technology instead of cable TV or radio transmissions. Another form of entertainment is **game playing**. Already we have multi person real-time simulation games, like hide-and- seek in a virtual dungeon and flight simulators with the players on one team trying to shoot down the players on the opposing team.
2. Our last category is **ubiquitous computing**, in which computing is embedded into everyday life. Many homes are already wired with security systems that include door and window sensors and there are many more sensors that can be folded in to a smart home monitor, such as energy consumption. The IoT applications on electricity, gas and water meters could also report usage over the network. A technology called **RFID** (**Radio Frequency IDentification**) will push this idea even further in the future. RFID tags are passive (i.e., have no battery) chips the size of stamps and they can already be affixed to books, passports, pets, credit cards, and other items in the home and out. This lets RFID readers locate and communicate with the items over a distance of up to several meters, depending on the kind of RFID.

**1.2 REQUIREMENTS**

To understand how to build a computer network from the ground up. Our approach to accomplishing this goal will be to start from first principles and then ask the kinds of questions we would naturally ask if building an actual network. It is important to recognize the underlying concepts because networks are constantly changing as the technology evolves and new applications are invented. The following are the essential factors required to build a computer network.

## Perspectives of Network

* **Scalable Connectivity**
* **Cost-Effective Resource Sharing**
* **Support for Common Services**
* **Manageability**

**1.2.1 Design Perspectives of Computer Networks**

As we noted above, ours focused on the perspective of someone who would design networking equipment and protocols. We continue to focus on this perspective. we also want to cover the

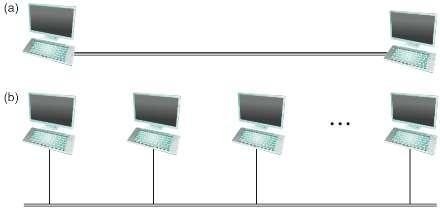
perspectives of two additional groups that are of increasing importance: those who develop networked applications and those who manage or operate networks. Let’s consider how these three groups might list their requirements for a network:

1. An ***application programmer*** would list the services that his or her application needs—for example, a guarantee that each message the application sends will be delivered without error within a certain amount of time or the ability to switch gracefully among different connections to the network as the user moves around.
2. A ***network operator*** would list the characteristics of a system that is easy to administer and manage—for example, in which faults can be easily isolated, new devices can be added to the network and configured correctly, and it is easy to account for usage.
3. A ***network designer*** would list the properties of a cost-effective design—for example, that network resources are efficiently utilized and fairly allocated to different users. Issues of performance are also likely to be important.

This section attempts to distill these different perspectives into a high-level introduction to the major considerations that drive network design.

1.2.2 **Scalable Connectivity**

A network must provide connectivity among a set of computers. Sometimes it is enough to build a limited network that connects only a few select machines. In contrast, other networks are designed to grow in a way that allows them the potential to connect all the computers in the world. A system that is designed to support growth to an arbitrarily large size is said to ***scale***.

To understand the requirements of connectivity more fully, we need to take a closer look at how computers are connected in a network. Connectivity occurs at many different levels. At the lowest level, a network can consist of two or more computers directly connected by some physical medium, such as a coaxial cable or an optical fiber. We call such a physical medium a ***link***, and we often refer to the computers it connects as

***nodes***. Sometimes a node is a more specialized piece of hardware rather than a computer. As illustrated in Figure 1.4, physical links are sometimes limited to a pair of nodes such a link is said to be ***point-to-***

***point***, while in other cases more than two Figure 1.4: Direct Links (a) Point-to Point (b) Multi-access

nodes may share a single physical link such a link is said to be ***multiple-access.*** Wireless links, such as those provided by cellular networks and Wi-Fi networks, are an increasingly important

class of multiple-access links. It is often the case that multiple-access links are limited in size, in terms of both the geographical distance they can cover and the number of nodes they can connect.

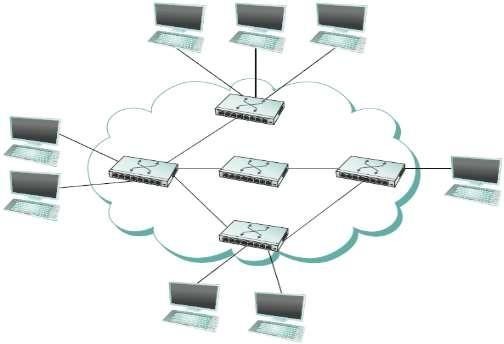


FIGURE 1.5 Switched network.

Fortunately, connectivity between two nodes does not necessarily imply a direct physical connection between them—indirect connectivity may be achieved among a set of cooperating nodes. Consider the following two examples of how a collection of computers can be indirectly connected.

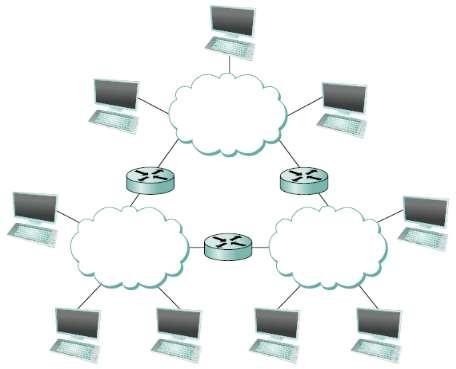
Figure 1.5 shows a set of nodes, each of which is attached to one or more point-to-point links.. Those nodes that are attached to at least two links run software that forwards data received on one link out on another. If organized in a systematic way, these forwarding nodes form a switched network. There are numerous types of switched networks, of which the two most common are ***circuit switched*** and ***packet switched***.

The ***circuit switched networks*** are most notably employed by the telephone system, while

the ***packet switched networks*** are used for most computer networks. Circuit switching is, however, making a bit of a comeback in the optical networking realm, which turns out to be important as demand for network capacity constantly grows.

The important feature of packet-switched networks is that the nodes in such a network send discrete blocks of data to each other. Think of these blocks of data as corresponding to some piece of application data such as a file, a piece of email, or an image. We call each block of data either a ***packet or a message***. Packet-switched networks typically use a strategy called ***store-and-forward***. Each node in a store-and-forward network first receives a complete packet over some link, stores the packet in its internal memory, and then forwards the complete packet to next node. In contrast, a circuit-switched network first establishes a dedicated circuit across a sequence of links and then allows the source node to send a stream of bits across this circuit to a destination node. The major reason for using packet switching rather than circuit switching in a computer network is efficiency.

The cloud in Figure 1.5 distinguishes between the nodes on the inside that implement the network they are commonly called ***switches***, and their primary function is to store and forward packets and the nodes on the outside of the cloud that use the network, they are commonly called ***hosts***, and they support users and run application programs. In general, we use a cloud to denote any type of network, whether it is a single point-to-point link, a multiple-access link, or a switched network.

A second way in which a set of computers can be indirectly connected is shown in Figure 1.6. In this situation, a set of independent networks (clouds) are interconnected to form an internetwork, or internet for short.

A node that is connected to two or more networks is commonly called a router or gateway, and it plays much the same role as a switch—it forwards messages from one network to another. Note that an internet can itself be viewed as another kind of network, which means that an internet can be built from an interconnection

of internets. Thus, we can recursively build arbitrarily large networks by interconnecting clouds to form larger ***clouds***.

Figure 1.6: Interconnection of Networks

Just because a set of hosts are directly or indirectly connected to each other does not mean that we have succeeded in providing host-to-host connectivity. The final requirement is that each node must be able to say which of the other nodes on the network it wants to communicate with. This is done by assigning an ***address*** to each node. An ***address*** is a byte string that identifies a node; that is, the network can use a node’s address to distinguish it from the other nodes connected to the network. When a source node wants the network to deliver a message to a certain destination node, it specifies the address of the destination node. If the sending and receiving nodes are not directly connected, then the switches and routers of the network use this address to decide how to forward the message toward the destination. The process of determining systematically how to forward messages toward the destination node based on its address is called ***routing***.

While routing a packet, if a source node wants to send a message to a single destination node, it is called ***unicast***. While this is the most common scenario, it is also possible that the source node might want to ***broadcast*** a message to all the nodes on the network. Or, a source node might want to send a message to some subset of the other nodes but not all of them, a situation called *multicast*.

**1.2.3 Cost-Effective Resource Sharing**

This section explains the key requirement of computer networks (packet-switched networks)— efficiency—that leads us to packet switching as the strategy of choice. Given a collection of nodes indirectly connected by a nesting of networks, it is possible for any pair of hosts to send messages to each other across a sequence of links and nodes.

To understand how hosts share a network, we need to introduce a fundamental concept, ***multiplexing***, which means that a system resource is shared among multiple users. Multiplexing (or *muxing*) is a way of sending multiple signals or streams of information over a communications

link at the same time in the form of a single, complex signal; the receiver recovers the separate signals, a process called *demultiplexing* (or *demuxing*).

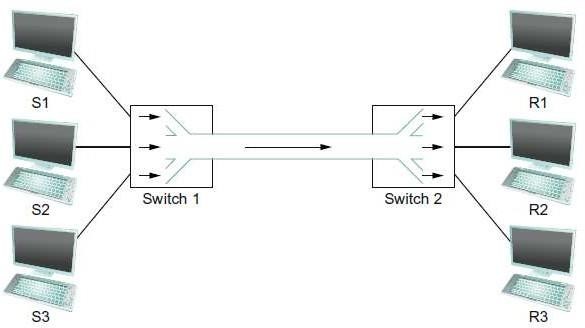
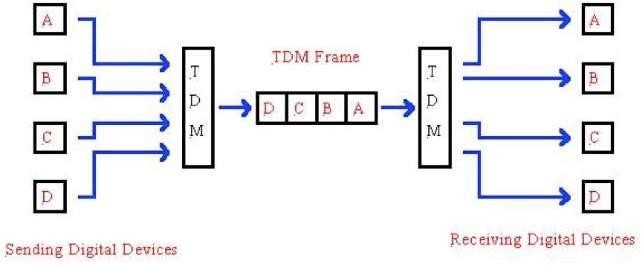
To see how this might work, consider the simple network illustrated in Figure 1.7, where the three hosts on the left side of the network (senders S1– S3) are sending data to the three hosts on the right (receivers R1–R3) by sharing a switched network that contains only one physical link. (For

Figure 1.7: Multiplexing and Demultiplexing simplicity, assume that host S1 is sending data to host R1, and so on.). In this situation, three flows of data—corresponding to the three pairs of hosts—are multiplexed onto a single physical link by switch 1 and then demultiplexed back into separate flows by switch 2.

There are several different methods for multiplexing multiple flows onto one physical link. One common method is ***synchronous time-division multiplexing (STDM)***. The idea of STDM

is to divide time into equal-

sized quanta and, in a round-

Figure 1.8: Synchronous Time Division Multiplexing

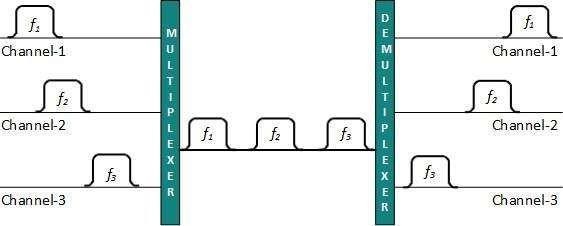
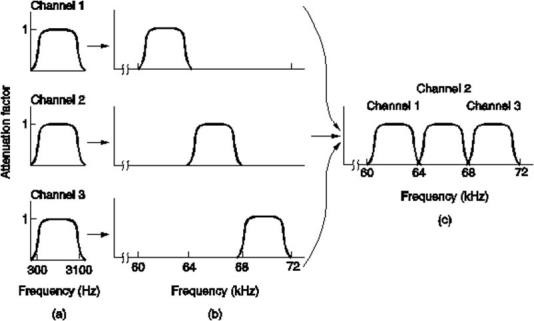
robin fashion, give each flow a chance to send data over the physical link.

1. In synchronous TDM, each device is given same **time slot** to transmit the data over the link, irrespective of the fact that the device has any data to transmit or not. Hence the name Synchronous TDM. Synchronous TDM requires that the total speed of various input lines should not exceed the capacity of path.
2. Each device places its data onto the link when its **time slot** arrives *i.e.* each device is given the possession of line turn by turn.
3. If any device does not have data to send then its time slot remains empty.
4. The various time slots are organized into **time frames** and each time frame consists of one or more time slots dedicated to each sending device.
5. If there are *n* sending devices, there will be *n* slots in a time frame A *i.e.* one slot for each device.

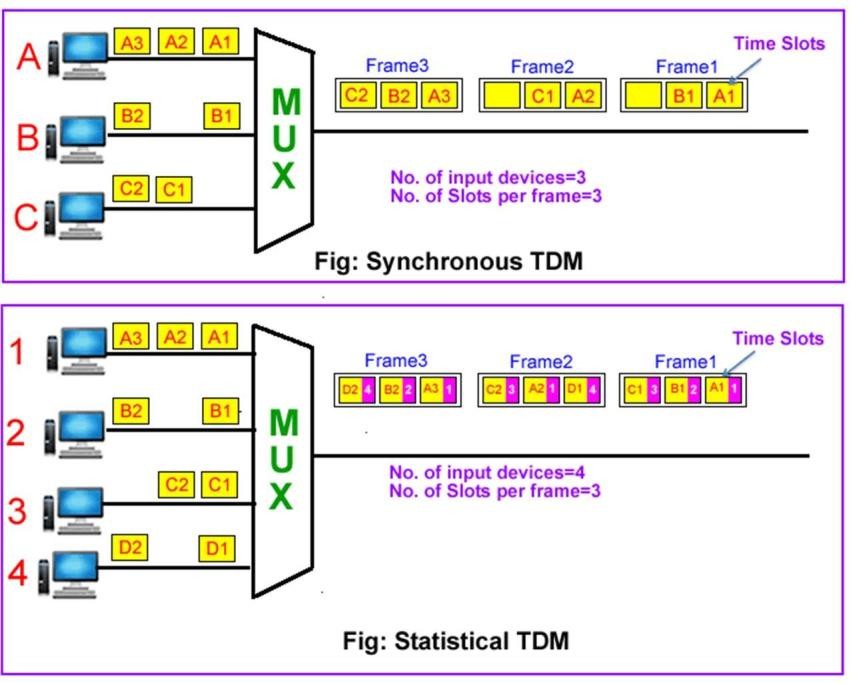
## Disadvantages of Synchronous TDM

1. The channel capacity cannot be fully utilized. Some of the slots go empty in certain time frame.
2. The capacity of single communication line that is used to carry the various transmission should be greater than the total speed of input lines.

Another method is ***frequency-division multiplexing (FDM).*** The idea of FDM is to transmit each flow over the physical link at a different frequency, much the same way that the signals for different TV stations are transmitted at a different frequency over the airwaves or on a coaxial cable TV link.

Although, simple to understaFnidgu, rbeo1t.h9:SFTreDquMenacynDdivFisDioMn Maureltilpilmexiitnegd in two ways. First, if one of the flows (host pairs) does not have any data to send, its share of the physical link— that is, its time quantum or its frequency—remains idle, even if one of the other flows has data to transmit.

***Statistical multiplexing*** dynamically allocates time slot to each channel on an as-needed basis. This is in contrast to time-division multiplexing (TDM) techniques, in which quiet devices use up a portion of the multiplexed data stream, filling it with empty packets.

Statistical multiplexing allocates bandwidth only to channels that are currently transmitting. It packages

the data from the active channels Figure 1.10: STDM Vs Statistical Multiplexing

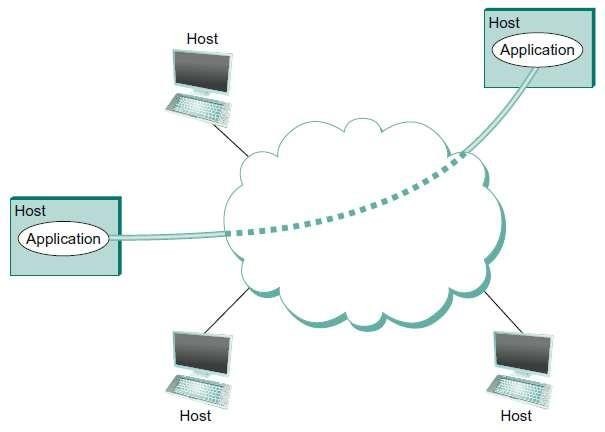
into packets and dynamically feeds them into the output channel.

Statistical multiplexing is facilitated through **packet mode** or **packet- oriented** communication, which among others is utilized in packet switched computer

networks. Each stream is divided into packets that normally are delivered asynchronously in a first- come first-served fashion. In alternative fashion, the packets may be delivered according to some scheduling discipline for fair queuing or differentiated and/or guaranteed quality of service.

**1.2.4 Support for common services**

The next requirement of a computer network is that the application programs running on the hosts connected to the network must be able to communicate in a meaningful way. From the application developer’s perspective, the network needs to make his or her life easier. The challenge for a network designer is to identify the right set of common services. The goal is to hide the complexity of the network from the application without overly constraining the application designer.

Intuitively, we view the network as providing logical channels over which application-level processes can communicate with each other; each channel provides the set of services required by that application. In other words, just as we use a cloud to abstractly represent connectivity among a set of computers, we now think of a channel as connecting one process to

another. Figure 1.11 shows a pair of applicatioFnig-ulerev1e.l11p:rPorcoecsessesessccoommmmuunniciactaintginogveorvaenrabastlroagcitccahlannel channel that is, in turn, implemented on top of a cloud that connects a set of hosts. We can think

of the channel as being like a pipe connecting two applications, so that a sending application can put data in one end and expect that data to be delivered by the network to the application at the other end of the pipe.

**1.2.5 Manageability**

Managing a network includes making changes as the network grows to carry more traffic or reach more users, and troubleshooting the network when things go wrong, or performance isn’t as desired. This requirement is partly related to the issue of scalability discussed above in section

* + 1. —as the Internet has scaled up to support billions of users and at least hundreds of millions of hosts, the challenges of keeping the whole thing running correctly and correctly configuring new devices as they are added have become increasingly problematic.

**1.3 Classification of computer networks**

One way to characterize networks is according to their size.

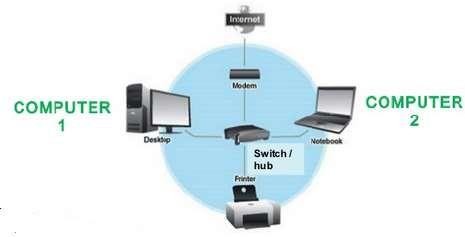
* + - * Networks are frequently classified according to the geographical boundaries spanned by the network itself.
      * PAN, LAN, WAN, MAN and SAN are the basic types of classification, of which LAN and WAN are frequently used.

**1.3.1 PAN (Personal Area Network)**

* PAN devices communicate over the range of a person.
* Used for data transmission among devices such as computers, telephones and personal digital assistants.
* The data cable is an example of PAN.
* This is also a Personal Area Network because that connection is for the user’s personal use. PAN is used for personal use only.
* A wireless personal area network (WPAN) is a personal area network — a network for interconnecting devices centered on an individual person's workspace — in which the connections are wireless.
* **Example**1 Bluetooth uses short-range radio waves over distances up to approximately 10 meters. For example, Bluetooth devices such as a keyboard, pointing devices, audio head sets, and printers may connect to personal digital assistants (PDAs), cell phones, or computers wirelessly.
* **Example**2 Radio-frequency identification (RFID) is the wireless use of electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects.

**1.3.2 LAN (Local Area Network)**

* + LAN is a network that connects computers and devices in a limited geographical area.
  + Example: Home, school computer laboratory, office building or closely positioned group of buildings.
  + The simplest form of LAN is to connect two computers together.



* LANs are inexpensive to install and also provide higher speeds.
* A network which consists of less than 500 interconnected devices across several buildings, is still recognized

as a LAN.

## Advantages of LAN:

Figure 1.12: Local Area Network

* + - Easy to share devices such as printers, scanners etc.,
    - Easy to share data such as pictures
    - Cost of LAN setup is low.

## Disadvantages of LAN:

* + - Power-a good LAN is required to be all the times.
    - Security-each computer and device become another point of entry for undesirables
    - Area covered is limited.

**Example**: IEEE 802.3 popularly called Ethernet.

**1.3.3 MAN (Metropolitan Area Network)**

* It is a high speed network that connects local area networks in a metropolitan area.
* Example: city or town handles the bulk of communication activity across that region.
* Is larger than a LAN, but smaller than a WAN
* Is also used to mean the interconnection of several LANs by bridging them together. This sort of network is also referred to as a campus network.

## Advantages:

Figure 1.13: Metropolitan Area Network

* + Efficiency and shared access.
  + All the computer-owing residents of the area have equal abiolity to go online.
  + MAN can cover a wider area than a LAN.

## Disadvantages:

* + It can be costly.
  + Security problem.
  + As the network consists of many computers over the span of a city, the connection can lag or become quite slow.

**Example:** WiMAX, Cable TV

**1.3.4 WAN (Wide area network):**

* WAN is a network that covers a larger geographic area (such as a city, country, or world) using a communications channel that combines many types of media such as telephone lines, cables and radio waves.
* Also called “enterprise networks” if they are privately owned by a large company.
* To cover great distances, WANs may transmit data over leased high speed phone lines or wireless links such as satellites.

## Types of WAN:

* + Enterprise private network (EPN)
  + Virtual private network (VPN)

An **enterprise private network** is a network build by an enterprise to interconnect carious company sites, eg: production sites, head offices, remote offices, and shops in order to share computer resources.

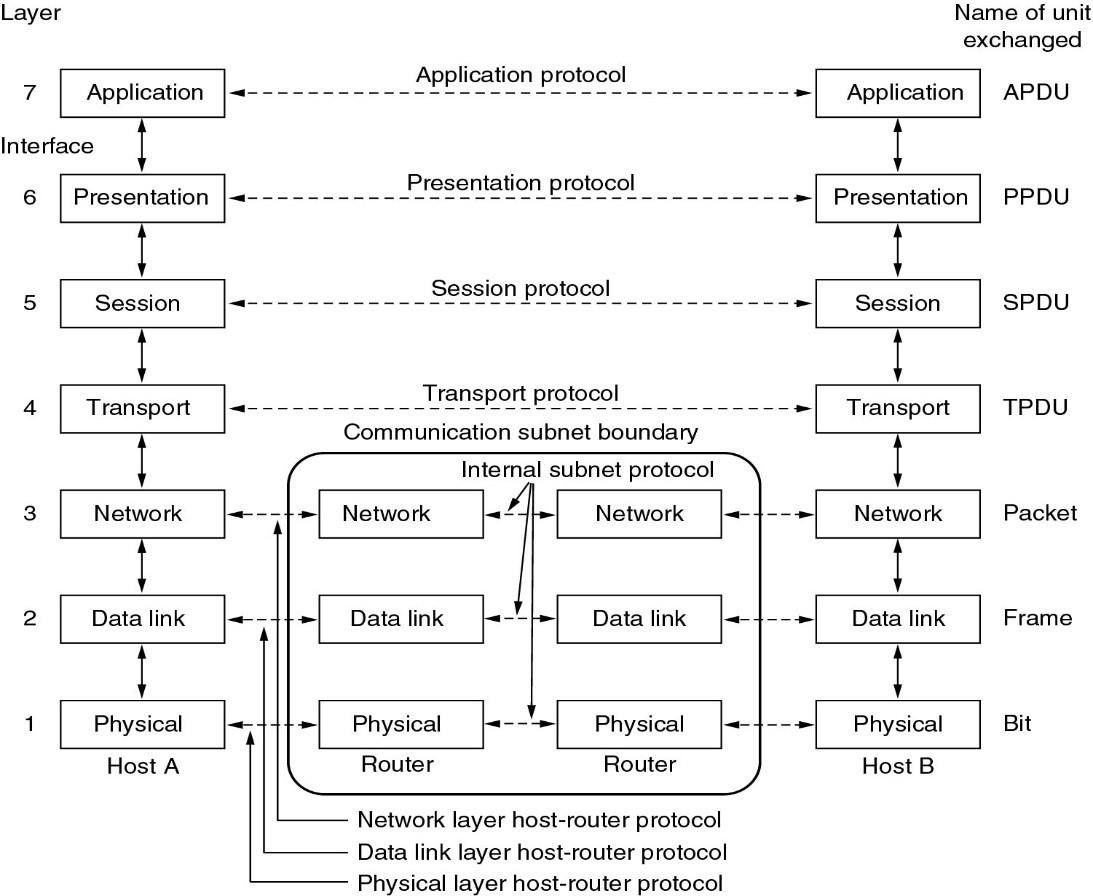
A **virtual private network** is a computer network in which some of the links between nodes are carried by open connections or virtual circuits in some larger network instead of by physical wires.

## Advantages: Disadvantages:

* Increased efficiency
* Ease of communication
* Lowered costs.
* Security problems.
* Training costs
* Maintenance problems.

**Example**: Internet

**1.3.5 Storage Area Network**

A **storage area network** (SAN) is a dedicated high-speed **network** or subnetwork that interconnects and presents shared pools of **storage** devices to multiple servers. A SAN moves **storage** resources off the common user **network** and reorganizes them into an independent, high- performance **network**.

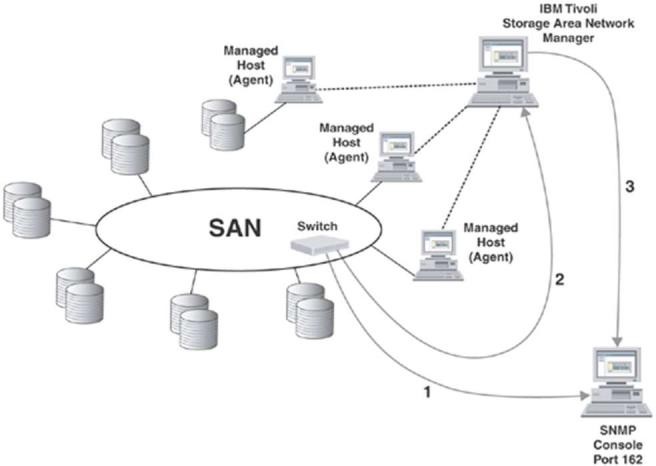


Figure 1.14: Storage Area Network

**1.4 NETWORK ARCHITECTURE**

To help deal with the complexity, network designers have developed general blueprints— usually called network architectures—that guide the design and implementation of networks. It also introduces two of the most widely referenced architectures—the OSI (or 7-layer) architecture and the Internet architecture.

**1.4.1 The 7-Layer Model**

This model group communication functions into seven logical layers. Control is passed from one layer to the next, starting at the application layer in one station, and proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

1. **Physical Layer**: Figure 1.15: 7-Layer OSI Model
   * Deals with the transmission of 0s and 1s over the physical media.
   * Translation of bits into signal and Encode bits into signals
   * Carry data from the higher layers
   * It helps to transmit bits over a medium.
   * Physical examples include Ethernet, FDDI, B8ZS, V.35, V.24, RJ45.
   * The unit of communication at the physical layer is a bit.

## Data link Layer

* + To organize bits into frames and helps in providing hop-to-hop delivery.
  + Create and detect frame boundaries.
  + Handle errors by implementing an acknowledgement and retransmission scheme.
  + Implement flow control.
  + Supports points-to-point as well as broadcast communication.
  + Supports simplex, half-duplex or full-duplex communication.
  + The unit of communication at the data link layer is a frame.

## Network Layer

* + To move packets from source to destination to provide internetworking.
  + Defines the most optimum path the packet should take from the source to the destination
  + Defines logical addressing so that any endpoint can be identified.
  + Handles congestion in the network.
  + The network layer also defines how to fragment a packet into smaller packets to accommodate different media.
  + The unit of communication at the network layer is a datagram

## Transport Layer

* + To provide reliable process-to-process message delivery and error recovery.
  + Purpose of this layer is to provide a reliable mechanism for the exchange of data between two processes in different computers.
  + Ensures that the data units are delivered error free, data units are delivered in sequence, no loss or duplication of data units. Provides connectionless or connection-oriented service.
  + The unit of communication at the transport layer is a segment, user datagram, or a packet, depending on the specific protocol used in this layer.

## Session Layer

* + To establish, manage and terminate sessions.
  + Session layer provides mechanism for controlling the dialogue between the two end systems. It defines how to start, control and end conversations (called sessions) between applications.
  + This layer requests for a logical connection to be established on an end-user’s request.
  + Session layer is also responsible for terminating the connection.

## Presentation Layer

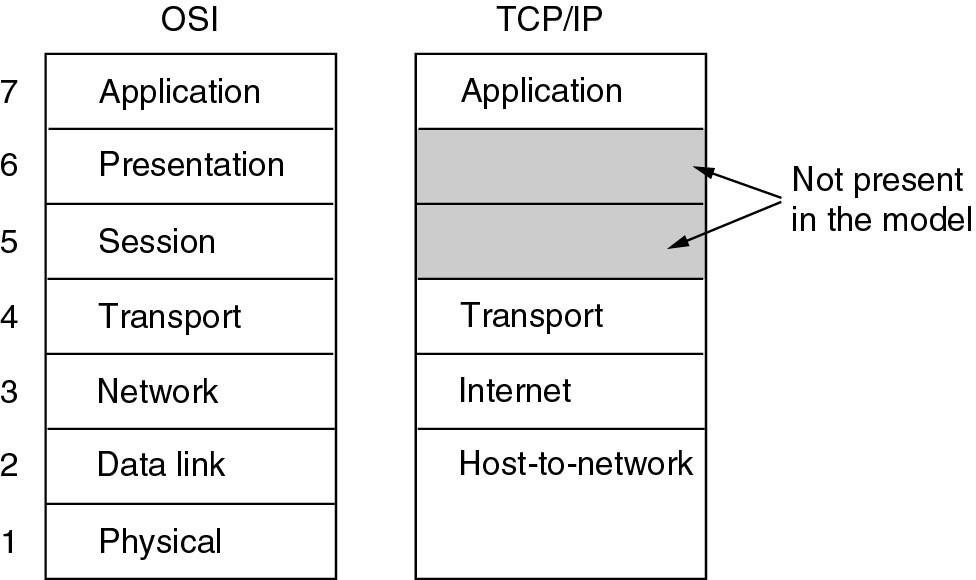
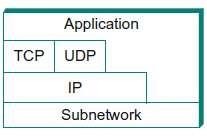
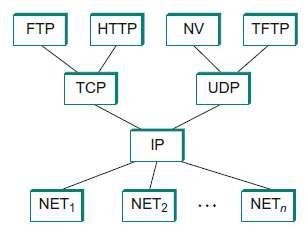
* + To translate, encrypt and compress data.
  + Presentation layer defines the format in which the data is to be exchanged between the two communicating entities.
  + Also handles data compression and data encryption (cryptography).

## Application Layer

* + To allow access to network resources.
  + Application layer interacts with application programs and is the highest level of OSI model.
  + Application layer contains management functions to support distributed applications.
  + Examples of application layer are applications such as file transfer, electronic mail, remote login etc.
  + The unit of communication at the application layer is a message.

**1.4.3 Drawbacks of OSI model**

* + OSI was too loosely defined and proprietary standards were too entrenched.
  + It's not even tangible.
  + The OSI model doesn't do any functions in the networking process.



**1.4.4 Transmission Control Protocol/Internet Protocol (TCP/IP)**

Figure 1.16: Internet (TCP/IP) Model

* It is the basic communication language or protocol of the Internet. TCP/IP provides end-to- end connectivity specifying how data should be formatted, addressed, transmitted, routed and received at the destination. Protocols exist for a variety of different types of communication services between computers.
* The Transmission Control Protocol/Internetworking Protocol (TCP/IP) is a set of protocols, or protocol suite that defines how all the transmissions are exchanged across the Internet.
* At the transport layer TCP/IP defines two protocols: TCP and User Datagram Protocol (UDP).
* At the network layer the main protocol defined by TCP/IP is the Internetworking Protocol (IP).
* At the physical and data link layers, TCP/IP does not define any specific protocol. It supports all the standard and proprietary protocols.
* TCP or UDP creates a data unit that is called either a segment or a user datagram.
* The IP layer creates a data unit called a datagram.
* The movement of the datagram across the Internet is the responsibility of the TCP/IP protocol.

**1.4.5 Function of each layer of Internet Model (TCP/IP)**

**Table 1.1 Function of each layer in Internet Model (TCP/IP)**

|  |  |  |
| --- | --- | --- |
| **Layer** | **Primary function** | **Examples** |
| Application | Do useful work with various network application programs. | HTTP, SMTP, POP, Ping, FTP |
| Transport | *Control* the flow of information between the application program running on the client and the application program running on the server | TCP (reliable), UDP  (unreliable) |
| Internetwork Layer (IP) | Route packets *between* networks (inter-network) | IP |
| Subnetwork Layer (Network access Layer) | Move data *within* a local area network | Ethernet |
| Define the physical characteristics of the communication hardware and medium | radio, twisted pair, fiber |

The **application layer** is where real work gets done. Users typically interact with application programs to retrieve Web pages, transfer files, log on to remote systems, send and read e-mail, conduct teleconferences, etc. In other cases, the "user" might be a computer -- for example, a search engine "spider" automatically downloading pages to index.

**Transport layer** programs do what the name suggest -- they transport information between the application program on the client and the application program on the server. There are two major transport layer protocols.

1. **Transmission control protocol** (**TCP**) is for applications that require a **reliable** connection between the client and server. TCP establishes a temporary

connection between the client and server and controls the transmission of information. It checks for transmission errors, lost packets, and packets arriving out of order, and tries to automatically correct these without "bothering" the application program. It also does **flow control** slowing transmission if it is too fast for the receiver.

1. The **user datagram protocol** (**UDP**), is an **unreliable** transport protocol with no sessions and no flow control. Error checking is optional. UDP is faster than TCP, and is suitable for **isochronous** applications like **voice over IP** (**VoIP**) or **streaming video** where nothing can be done if an error is detected.

The **Internetwork layer(IP)** is responsible for routing packets between networks. The network layer protocol is called **Internet protocol** or **IP** for short. Again, as the name "inter-net" implies, IP moves information *between networks*. Since routing efficiency is critical, IP is simple and fast. The complexity of message integrity is left to TCP.

The **Subnetwork Layer** is used for moving information between two hosts *within* a local area network. The **Subnetwork Layer** Standards at this layer spell out the physical characteristics of the medium, for example radio, optical fiber or twisted pairs of copper wire, and the physical definition of "one" and "zero" bits (the **modulation** method).

Finally, note that nearly all operating systems today -- Windows, Mac, Linux and other versions of UNIX -- include programs for the TCP and IP protocols. They also come with common application programs like Web, FTP, Ping, Telnet, Traceroute, POP and SMTP clients.

**1.4.6 Three features of Internet Architecture**

**First**, as best illustrated by Figure 1.15, the Internet architecture ***does not imply strict layering***. The application is free to bypass the defined transport layers and to directly use IP or one of the underlying networks. In fact, programmers are free to define new channel abstractions or applications that run on top of any of the existing protocols.

**Second,** if you look closely at the protocol graph in Figure 1.16, we will notice an hourglass shape—wide at the top, narrow in the middle, and wide at the bottom. This shape actually reflects the central philosophy of the architecture. That is, IP serves as the focal point for the architecture— it ***defines a common method for exchanging packets among a wide collection of networks***. Above IP there can be arbitrarily many transport protocols, each offering a different channel abstraction to application programs. Thus, the issue of delivering messages from host to host is completely separated from the issue of providing a useful process-to-process communication service. Below

IP, the architecture allows for arbitrarily many different network technologies, ranging from Ethernet to wireless to single point-to-point links.

A **final attribute** of the Internet architecture (or more accurately, of the IETF culture) is that ***in order for a new protocol to be officially included in the architecture***, there must be both a protocol specification and at least one (and preferably two) representative implementations of the specification. The existence of working implementations is required for standards to be adopted by the IETF. This cultural assumption of the design community helps to ensure that the architecture’s protocols can be efficiently implemented.

**1.4.7 A Comparison of the OSI and TCP/IP Reference Models**

**Similarities:** The OSI and TCP/IP reference models have much in common. Both are based on the concept of a stack of independent protocols. Also, the functionality of the layers is roughly similar. In both models the layers up through and including the transport layer are there to provide an end- to-end, network-independent transport service to processes wishing to communicate. These layers form the transport provider. Again in both models, the layers above transport are application- oriented users of the transport service.

**Table 1.2 Difference between OSI & TCP/IP**

|  |  |
| --- | --- |
| OSI Model | TCP / IP |
| 1. It has 7 layers 2. Transport layer guarantees delivery of packets 3. Separate presentation layer, 4. Separate session layer. | 1. Has 4 layers 2. Transport layer does not guarantees delivery of packets 3. No session layer, 4) No presentation layer,   characteristics are provided by appln layer |
| 1. Network layer provides both connectionless and oriented services. 2. It defines the services, interfaces and   protocols very clearly and makes a clear distinction between them. | 1. Network layer provides only connection less services 2. No clear distinguishes between service interface and protocols |
| 1. The protocol is better hidden and can be easily replaced as the technology changes. 2. OSI truly is a general model. 3. It has a problem of protocol filtering into a model | 1. It is not easy to replace the protocols 2. TCP/IP cannot be used for any other application 3. The model does not fit any protocol stack. |

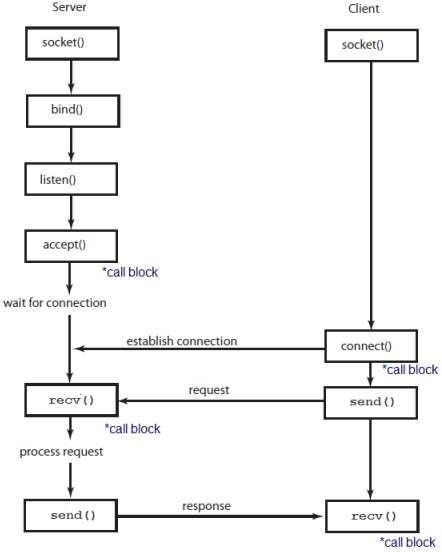
**1.5 IMPLEMENTING NETWORK SOFTWARE**

This section introduces some of the issues involved in implementing a network application on top of the Internet architecture. Typically, such programs are simultaneously an application (i.e., designed to interact with users) and a protocol (i.e., communicates with peers across the network).

**1.5.1 Application Programing Interface**

The place to start when implementing a network application is the interface exported by the network. Since most network protocols are implemented in software (especially those high in the protocol stack), and nearly all computer systems implement their network protocols as part of the operating system, This interface is often called the ***network application programming interface (API)***. Although each operating system is free to define its own network API, over time certain of these APIs have become widely supported; This is called the ***socket interface*** originally provided by the Berkeley distribution of Unix, which is now supported in virtually all popular operating systems. The first step is to create a socket, which is done with the following operation:

int sockfd = socket(address\_family, type, protocol);

This operation takes three arguments is that the socket interface was designed to be general enough to support any underlying protocol suite. Specifically, the ***domain*** argument specifies the protocol *family* that is going to be used: PF\_INET denotes the Internet family, PF\_UNIX denotes the Unix pipe facility, and PF\_PACKET denotes direct access to the network interface (i.e., it bypasses the TCP/IP protocol stack). The ***type*** argument indicates the semantics of the communication. SOCK\_STREAM is used to denote a byte stream. SOCK\_DGRAM is an alternative that denotes a message-oriented service, such as that provided by UDP. The ***protocol*** argument identifies the

specific protocol that is going to be used. In our Figure 1.17: Implementing Network Software using Sockets case, the combination of PF\_INET and SOCK\_STREAM implies TCP. Finally, the return value from socket is a *handle* for the newly created socket—that is, an identifier by which we can refer to the socket in the future.

On a server machine, the application process performs a *passive* open—the server says that it is prepared to accept connections, but it does not actually establish a connection. The server does this by invoking the following three operations:

int bind (int socket, struct sockaddr \*address, int addr\_len) int listen (int socket, int backlog)

int accept (int socket, struct sockaddr \*address,int \*addr\_len)

The ***bind*** operation, as its name suggests, binds the newly created socket to the specified address. This is the network address of the *local* participant—the server. Note that, when used with the Internet protocols, address is a data structure that includes both the IP address of the server

and a TCP port number. The port number is usually some well-known number specific to the service being offered; for example, web servers commonly accept connections on port 80.

The ***listen*** operation then defines how many connections can be pending on the specified socket. Finally, the accept operation carries out the passive open. It is a blocking operation that does not return until a remote participant has established a connection, and when it does complete it returns a *new* socket that corresponds to this just-established connection, and the address argument contains the *remote* participant’s address. Note that when accept returns, the original socket that was given as an argument still exists and still corresponds to the passive open; it is used in future invocations of accept.

On the client machine, the application process performs an *active* open; that is, it says who it wants to communicate with by invoking the following single operation:

int connect (int socket, struct sockaddr \*address,int addr\_len)

This operation does not return until TCP has successfully established a connection, at which time the application is free to begin sending data. In this case, address contains the remote participant’s address.

Once a connection is established, the application processes invoke the following two operations to send and receive data:

int send (int socket, char \*msg, int msg\_len,

int flags) int recv (int socket, char \*buff, int buff\_len,

int flags)

The first operation sends the given message over the specified socket, while the second operation receives a message from the specified socket into the given buffer. Both operations take a set of flags that control certain details of the operation.

**Using TCP/IP sockets, write a client-server program to make the client send the file name and to make the server send back the contents of the requested file if present.**

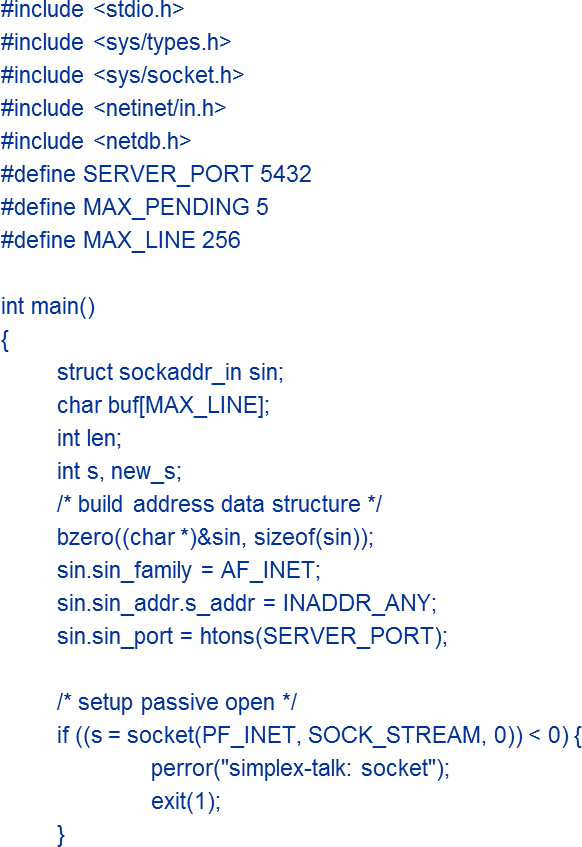
Algorithm (Server Side)

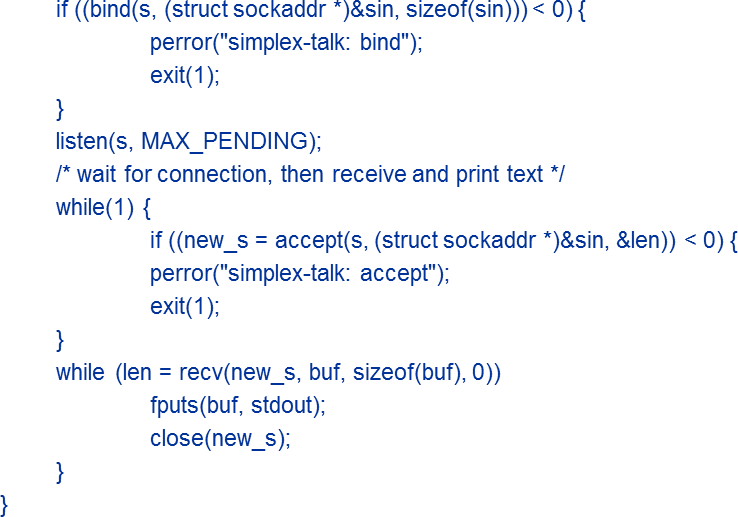
1. Start.
2. Create a socket using socket() system call.
3. Bind the socket to an address using bind() system call.
4. Listen to the connection using listen() system call.
5. accept connection using accept()
6. Receive filename and transfer contents of file with client.
7. Stop.

Algorithm (Client Side)

1. Start.
2. Create a socket using socket() system call.
3. Connect the socket to the address of the server using connect() system call.
4. Send the filename of required file using send() system call.
5. Read the contents of the file sent by server by recv() system call.
6. Stop.

/\*Server.c\*/





## /\*Client.c\*/

**1.6 Performance**

Up to this point, we have focused primarily on the functional aspects of network. Like any computer system, however, computer networks are also expected to perform well. This is because the effectiveness of computations distributed over the network often depends directly on the efficiency with which the network delivers the computation’s data.

Network performance refers to measures of service quality of a network as seen by the customer. There are many different ways to measure the performance of a network, as each network is different in nature and design.

The following measures are often considered important:

* + **Bandwidth** commonly measured in bits/second is the maximum rate that information can be transferred
  + **Throughput** is the actual rate that information is transferred
  + **Latency** the delay between the sender and the receiver decoding it, this is mainly a function of the signals travel time, and processing time at any nodes the information traverses
  + **Jitter** variation in packet delay at the receiver of the information
  + **Error rate** the number of corrupted bits expressed as a percentage or fraction of the total sent

**1.6.1 Bandwidth and Latency**

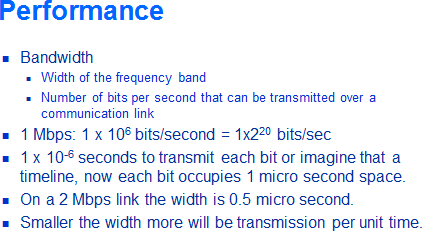
Network performance is measured in two fundamental ways: bandwidth (also called throughput) and latency (also called delay). The bandwidth of a network is given by the number of bits that can be transmitted over the network in a certain period of time. For example, a network might have a bandwidth of 10 million bits/second (Mbps), meaning that it is able to deliver 10 million bits every second. The maximum data rate that is available on the link is called Bandwidth and the throughput to refer to the measured performance of a system. For example, because of inefficiencies of implementation, a pair of nodes connected by a link with a bandwidth of 10 Mbps might achieve a throughput of only 2 Mbps.

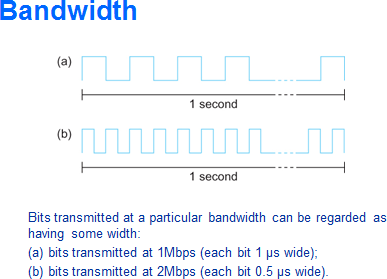
The second performance metric, latency, corresponds to how long it takes a message to travel from one end of a network to the other. Latency is measured strictly in terms of time. For example, a transcontinental network might have a latency of 24 milliseconds (ms); that is, it takes a message 24 ms to travel from one coast of North America to the other. round-trip time (RTT) of the network is the time to send a message from one end of a network to the other and back, rather than the one- way latency.

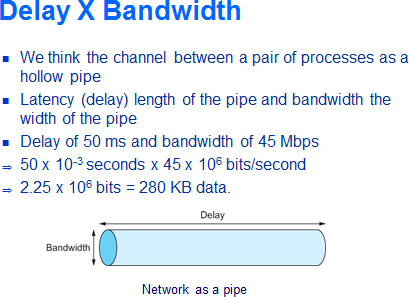
We often think of latency as having three components. First, there is the speed-of-light propagation delay. This delay occurs because nothing, including a bit on a wire, can travel faster than the speed of light. If you know the distance between two points, you can calculate the speed-of light latency, although you have to be careful because light travels across different media at different speeds: It travels at 3.0×108 m/s in a vacuum, 2.3×108 m/s in a copper cable, and 2.0×108 m/s in an optical fiber. Second, there is the amount of time it takes to transmit a unit of data. This is a function of the network bandwidth and the size of the packet in which the data is carried. Third, there may be queuing delays inside the network, since packet switches generally need to store packets for some time before forwarding them on an outbound link.

Latency = Propagation+Transmit+Queue

Propagation = Distance/SpeedOfLight Transmit = Size/Bandwidth







# Relative importance of bandwidth and latency depends on application

* + - * For large file transfer, bandwidth is critical
      * For small messages (HTTP, NFS, etc.), latency is critical
      * Variance in latency (jitter) can also affect some applications (e.g., audio/video conferencing)
      * How many bits the sender must transmit before the first bit arrives at the receiver if the sender keeps the pipe full
      * Takes another one-way latency to receive a response from the receiver
      * If the sender does not fill the pipe—send a whole delay × bandwidth product’s worth of data before it stops to wait for a signal—the sender will not fully utilize the network
    - Infinite bandwidth
      * RTT dominates
      * Throughput = TransferSize / TransferTime
      * TransferTime = RTT + 1/Bandwidth x TransferSize
    - Its all relative
      * 1-MB file to 1-Gbps link looks like a 1-KB packet to 1-Mbps link

