

**SIDDARTHA INSTITUTE OF SCIENCE AND
TECHNOLOGY
(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**



**COMPUTER NETWORKS
(20CS0513)**

COURSE OBJECTIVES

COURSE OBJECTIVES

1. *Build an understanding of the fundamental concepts of computer networking.*
2. *Understand how errors detected and corrected that occur in transmission*
3. *How collisions to be handled when many stations share a single channel*
4. *Know about routing mechanisms and different routing protocols*
5. *Understand transport layer functions*
6. *Know about different application layer protocols*

COURSE OUTCOMES

On successful completion of the course, the student will be able to

1. **Understand** the terminology and concepts of OSI and TCP/IP Reference models and identify different physical media used for data transmission
2. **Explain and implement** the services of Data link layer
3. **Describe** the principles of network layer and **implement** routing algorithms used for data transmission
4. **Demonstrate** the essential services of transport layer
5. **Contrast** the functioning of various protocols of Application layer
6. **Illustrate** the principles of networking

UNIT- I

Introduction: Networks, Network criteria, Physical structures - Reference Models: The OSI Reference Model, The TCP/IP Reference Model- Introduction to physical layer: Data and Signals, Data rate limits, Performance.

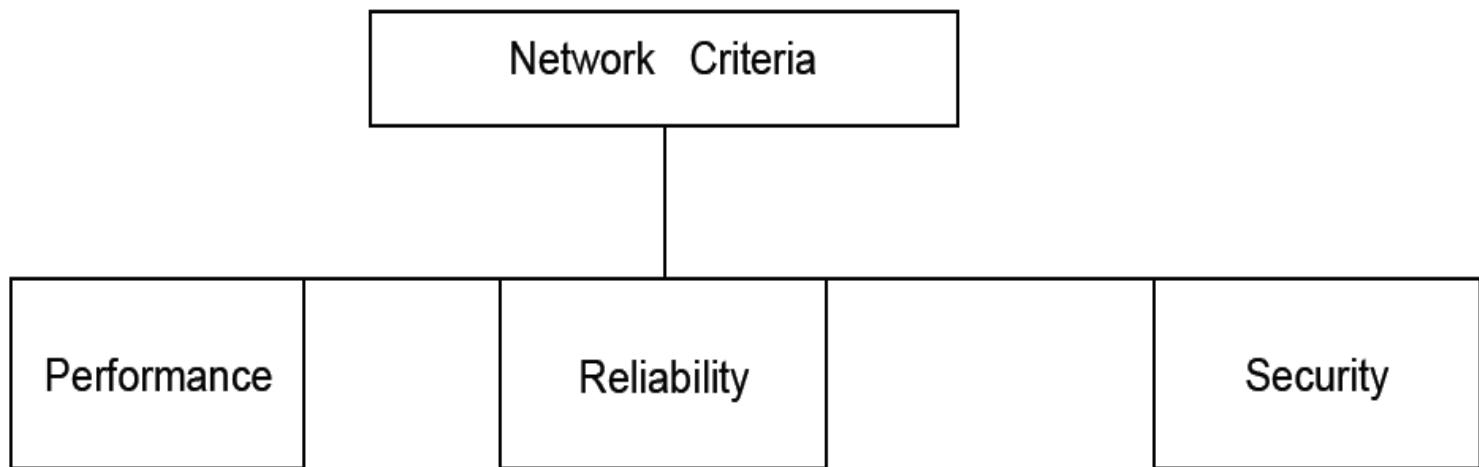
Transmission Media: Introduction,-Guided Media, -Unguided Media.

COMPUTER NETWORK

INTRODUCTION

An interconnected collection of **autonomous** computers is called a computer network. Two computers are said to be interconnected if they are able to exchange the information. If one computer can forcibly start, stop and control another one, the computers are not autonomous. A system with one control unit and many slaves is not a network, nor is a large computer with remote printers and terminals.

Network Criteria



Performance:

The performance can be measured in many ways and depends on number of factors.

Number of users

Type of transmission medium

Hardware

Software

Reliability

This is measured by the following factors

- Frequency of failure
- Recovery time of a network after a failure.
- Catastrophe.

Security

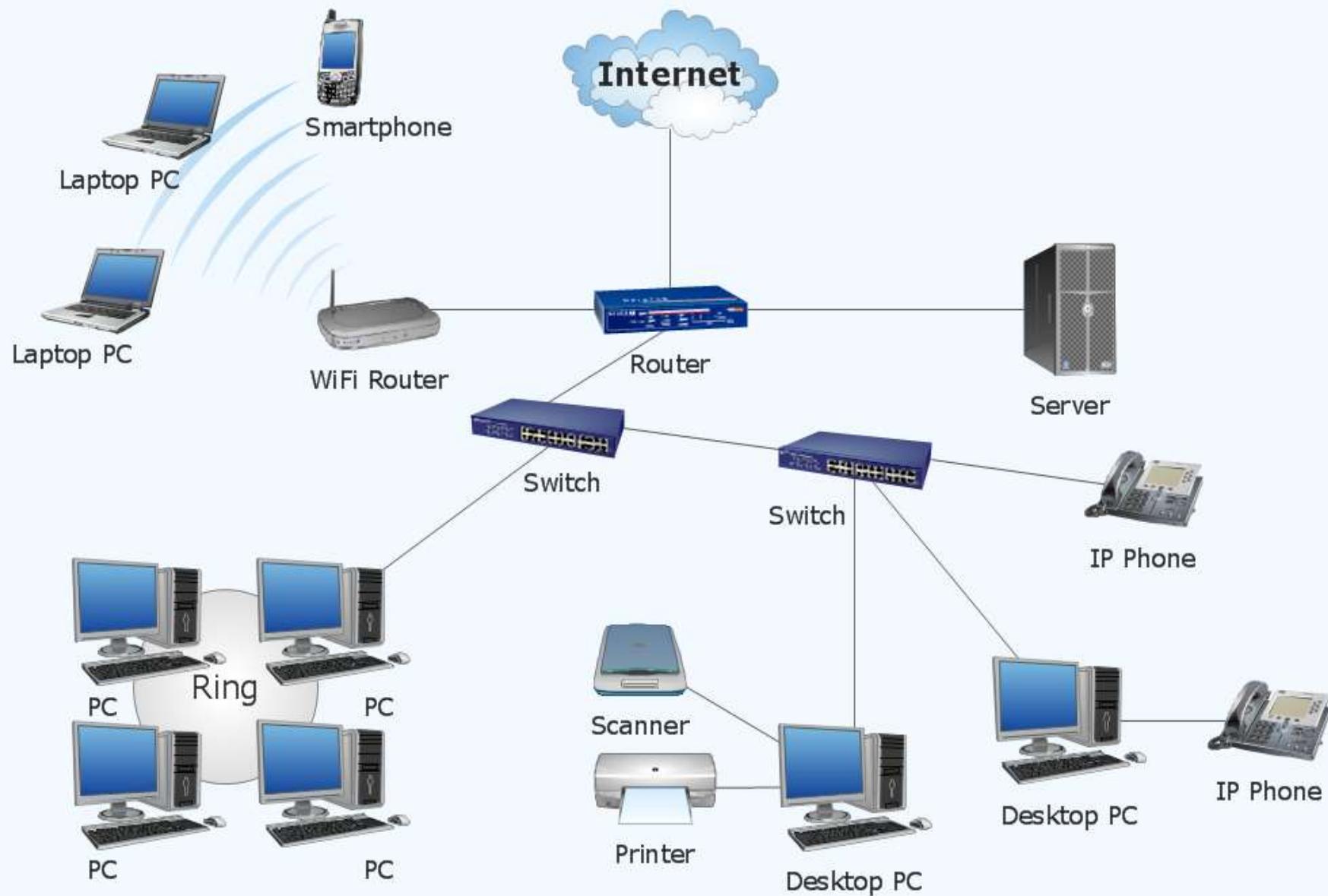
Network security issues include protecting data from the following

- Unauthorized access
- Viruses

Applications

- Accessing Remote databases
- Accessing Remote programs
- Value added communication facility
- Marketing and sales
- Financial services
- Manufacturing
- Electronic message
- Directory services
- Information services
- Teleconferencing
- Cellular telephone
- Cable television

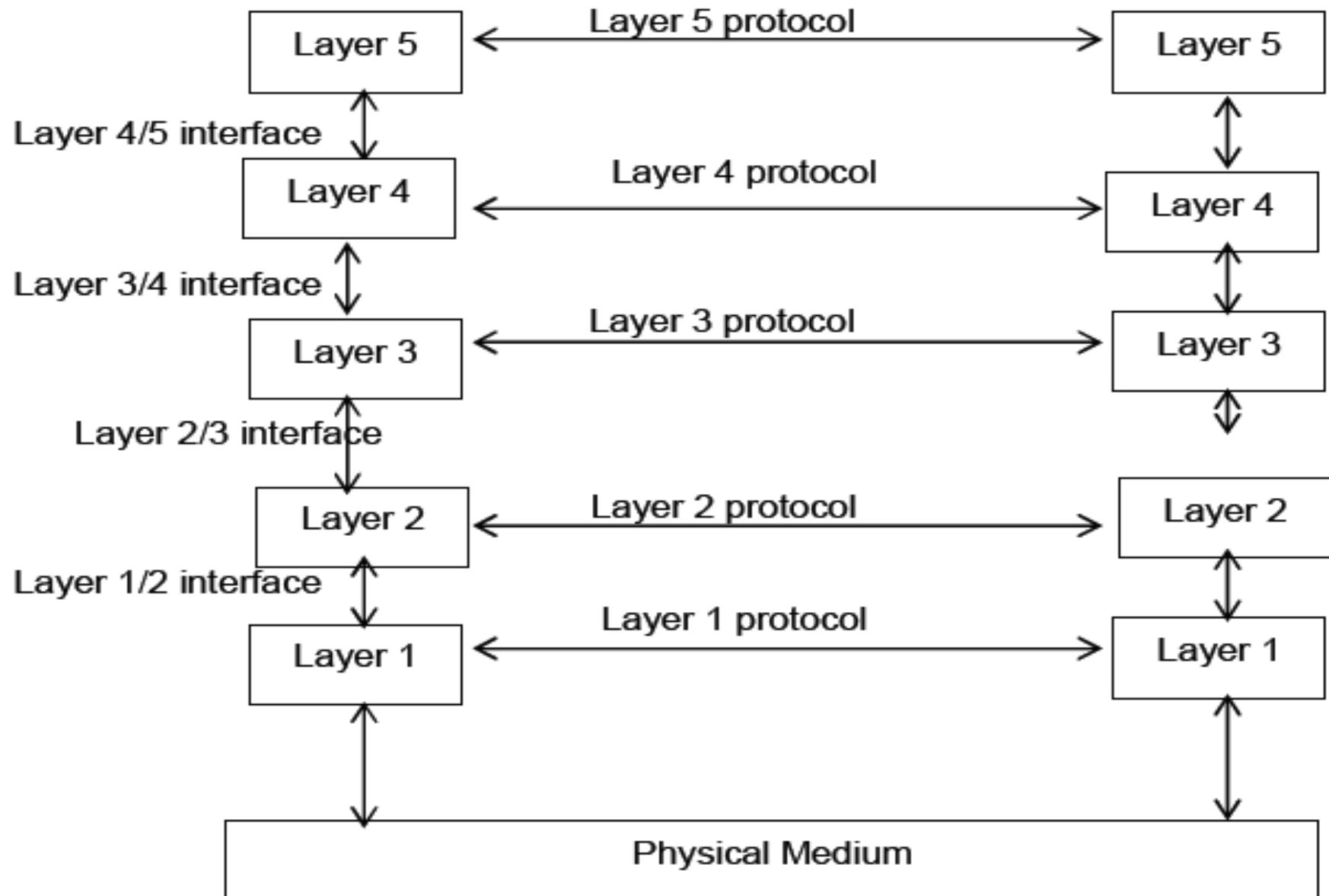
Physical Network Structure



Network Architecture

To reduce the design complexity, most networks are organized as a series of layers or levels, each built upon on the one below it. The number of layers, the name of each layer ,the contents of each layer ,and the function of each layer differ from network to network

However, in all networks the purpose of each layer is to offer certain services to the higher layers ,shielding those layers from the details of how the offered services are actually implemented



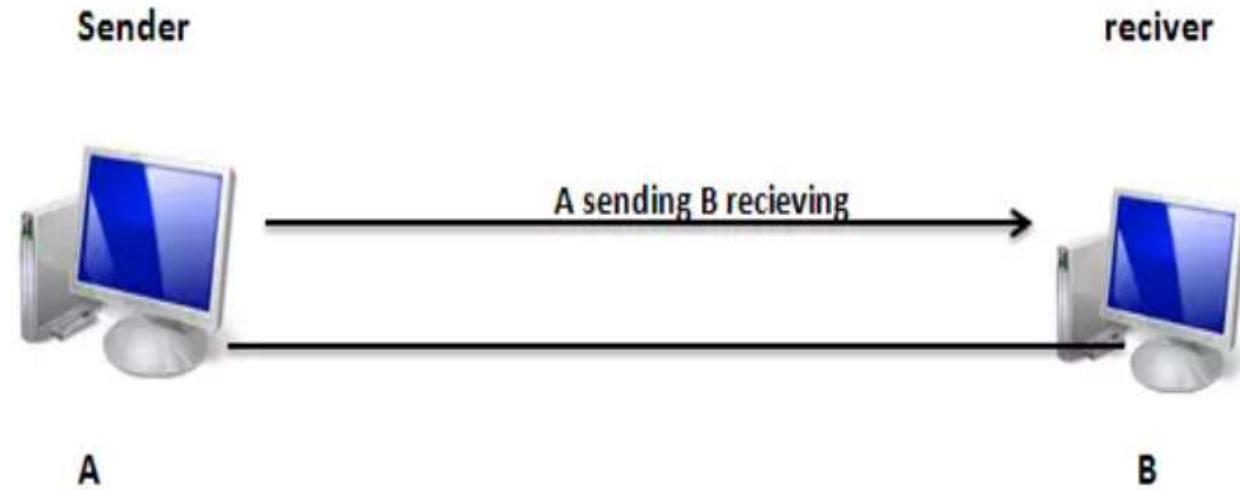
Data transfer methods:

- **Simplex communication:**

Data will be transferred in one direction only.

Example of Simplex Mode

One way traffic road is an example of Simplex mode. In the way the traffic travels in only one way. No vehicle can enter from the opposite side of the road. The all capacity is used by the sender.

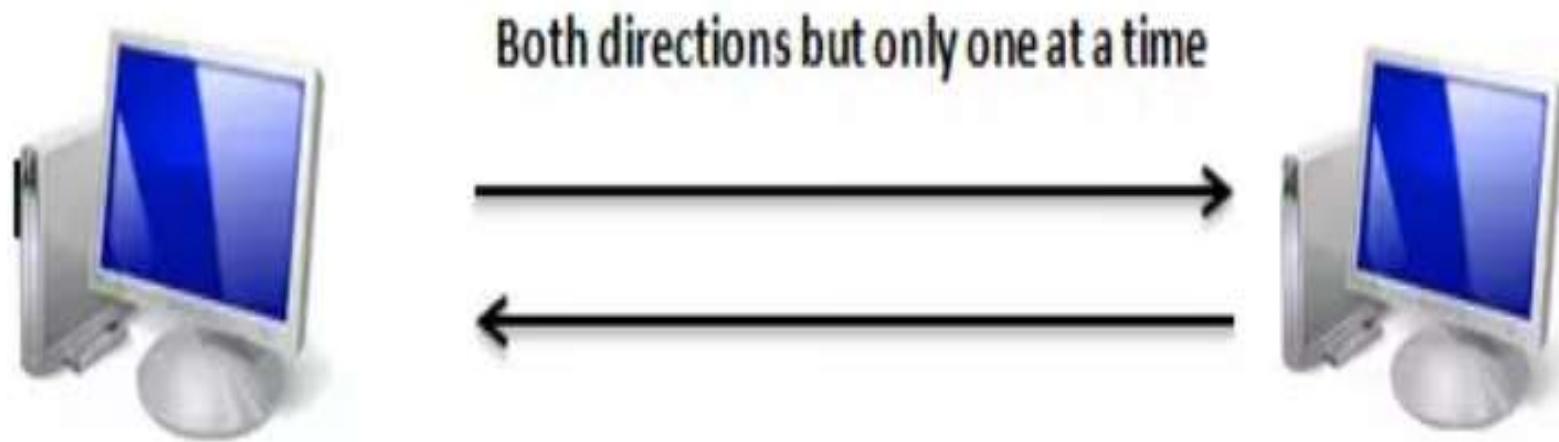


Advantage of Simplex mode

- The full capacity of the transmission medium is used by the sender.
- It is a very simple way of transmission.
- It has no traffic issues.

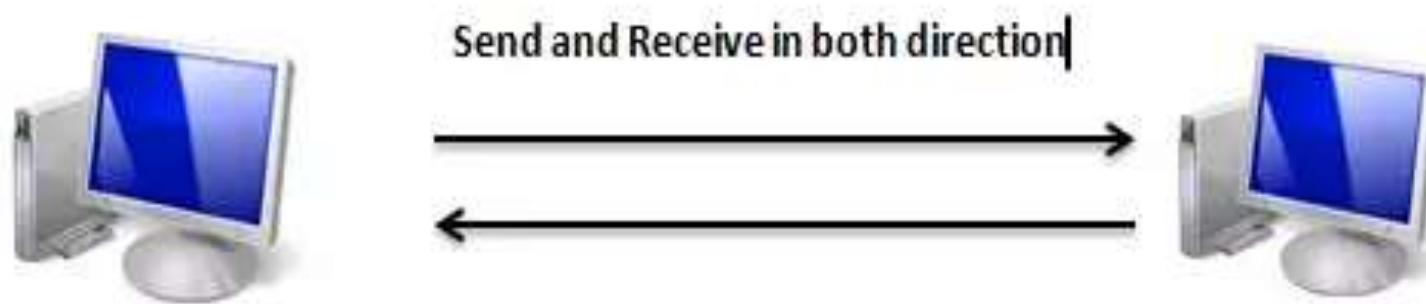
Half -- duplex communication:

In a half-duplex mode, data can be transmitted in both ways. Both devices can send or receive data in both directions but not at the same time. When one device can send signals, the other only receive. The half Duplex mode used in situations where there is no need for transmission on both sides at the same time



Full-Duplex Mode

In full-duplex mode, data is transmitted in both directions at the same time. Sender and receiver can send data at the same time.



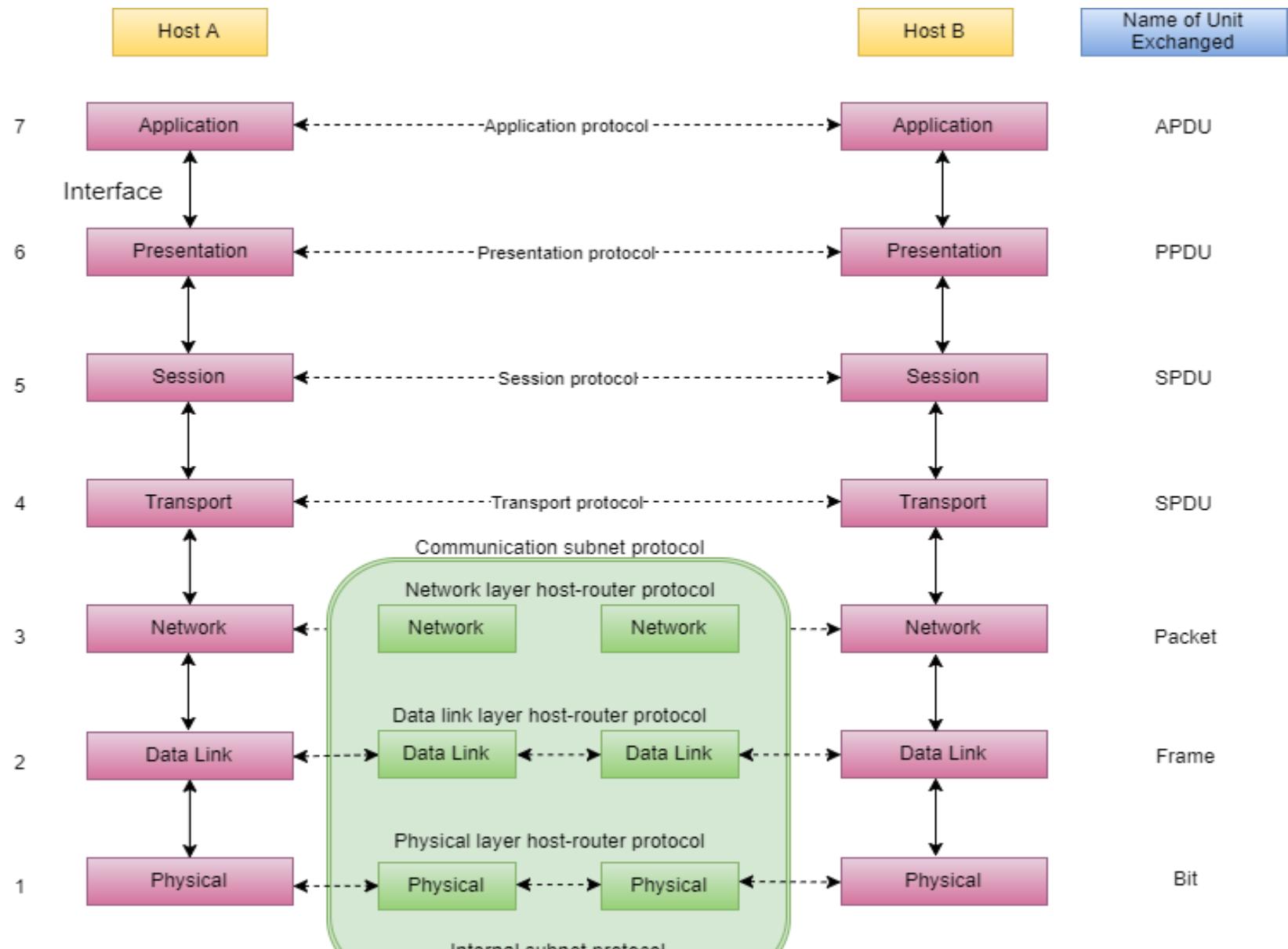
REFERENCE MODELS

The ISO OSI REFERENCE MODEL

In 1947, the International Standards Organization (ISO) proposed a network model that covers all network communications .This model is called Open Systems Interconnection (OSI) model. An open system is a model that allows any two different systems to communicate regardless of their underlying architecture.

The OSI model is built of seven layers: Physical (layer 1), Data link (layer 2), Network (layer 3), Transport (layer 4), Session (layer 5), Presentation (layer 6) and Application layers (layer 7).

OSI REFERENCE MODEL



Functions of the Layers

Physical Layer :

- Physical characteristics of interfaces and media
- Representation of bits.
- Data rate
- Synchronisation of bits
- Line configuration (point to point or multipoint)
- Transmission Mode
- Physical Topology

Data Link Layer :

- Framing
- Physical addressing
- Error control
- Flow control
- Access control

Network Layer :

- Routing
- Congestion control
- Billing

Transport Layer :

- Service – Point addressing
- Segmentation and reassembly
- Flow control
- Error control

Session Layer :

- Dialog control
- Synchronization

Presentation Layer :

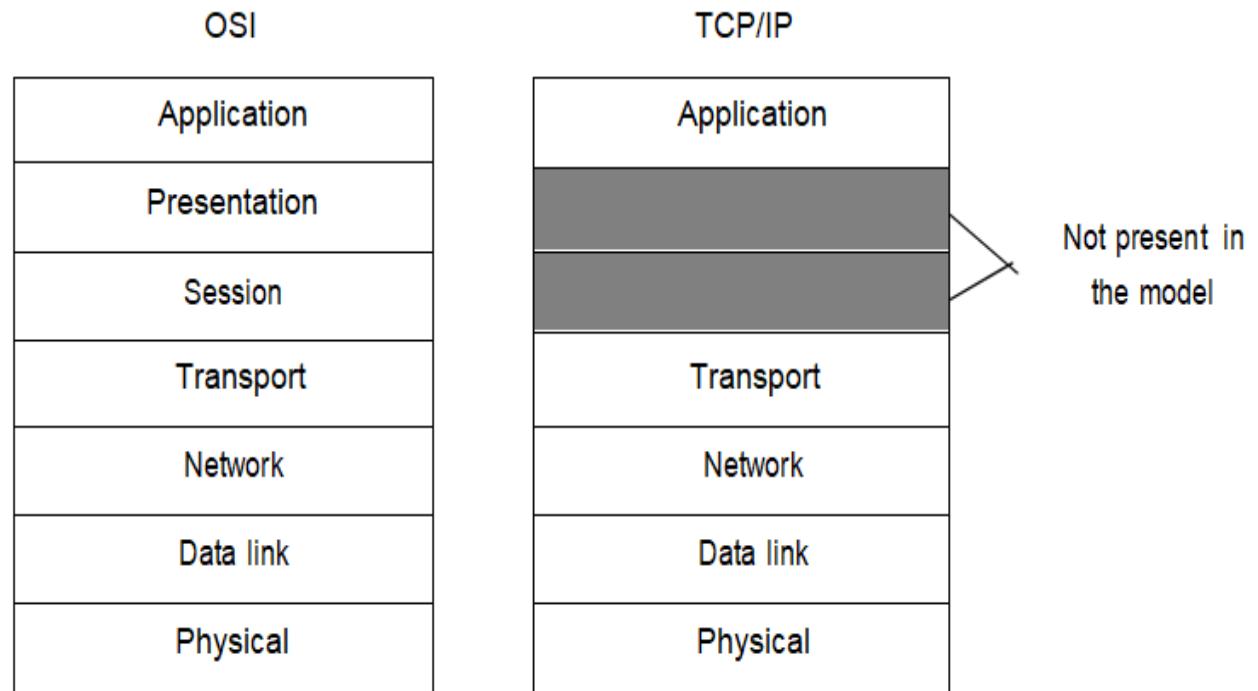
- Data encoding
- Encryption
- Compression

Application Layer :

- File Transfer
- Mail services
- Directory services

TCP/IP reference model

TCP/IP means Transmission Control Protocol and Internet Protocol. It is the network model used in the current Internet architecture as well. **Protocols** are set of rules which govern every possible communication over a network. These protocols describe the movement of data between the source and destination or the internet. They also offer simple naming and addressing schemes.



Layer1: Physical Layer

The following are the main responsibilities of the physical layer:

1. Definition of Hardware Specifications: The details of operation of cables, connectors, wireless radio transceivers, network interface cards and other hardware devices are generally a function of the physical layer (although also partially the data link layer; see below).

2. Encoding and Signaling: The physical layer is responsible for various encoding and signaling functions that transform the data from bits that reside within a computer or other device into signals that can be sent over the network.

3. Data Transmission and Reception: After encoding the data appropriately, the physical layer actually transmits the data, and of course, receives it. Note that this applies equally to wired and wireless networks, even if there is no tangible cable in a wireless network!

4. Topology and Physical Network Design: The physical layer is also considered the domain of many hardware-related network design issues, such as LAN and WAN topology.

Layer 2: Data Link Layer Functions

The following are the key tasks performed at the data link layer:

- **Logical Link Control (LLC):** Logical link control refers to the functions required for the establishment and control of logical links between local devices on a network. As mentioned above, this is usually considered a DLL sublayer; it provides services to the network layer above it and hides the rest of the details of the data link layer to allow different technologies to work seamlessly with the higher layers. Most local area networking technologies use the IEEE 802.2 LLC protocol.
- **Media Access Control (MAC):** This refers to the procedures used by devices to control access to the network medium. Since many networks use a shared medium (such as a single network cable, or a series of cables that are electrically connected into a single virtual medium) it is necessary to have rules for managing the medium to avoid conflicts. For example. Ethernet uses the CSMA/CD method of media access control, while Token Ring uses token passing.

- **Data Framing:** The data link layer is responsible for the final encapsulation of higher-level messages into *frames* that are sent over the network at the physical layer.
- **Addressing:** The data link layer is the lowest layer in the OSI model that is concerned with addressing: labeling information with a particular destination location. Each device on a network has a unique number, usually called a *hardware address* or *MAC address*, that is used by the data link layer protocol to ensure that data intended for a specific machine gets to it properly.
- **Error Detection and Handling:** The data link layer handles errors that occur at the lower levels of the network stack. For example, a cyclic redundancy check (CRC) field is often employed to allow the station receiving data to detect if it was received correctly.

Layer 3: Network layer

- 1.Selection of a packet switching network which is based on a connectionless internetwork layer is called a internet layer.
- 2.It is the layer which holds the whole architecture together.
- 3.It helps the packet to travel independently to the destination.
- 4.Order in which packets are received is different from the way they are sent.
- 5.IP (Internet Protocol) is used in this layer.
- 6.The various functions performed by the Internet Layer are:
 - Delivering IP packets
 - Performing routing
 - Avoiding congestion

Layer 4: Transport Layer

1. It decides if data transmission should be on parallel path or single path.
2. Functions such as multiplexing, segmenting or splitting on the data is done by transport layer.
3. The applications can read and write to the transport layer.
4. Transport layer adds header information to the data.
5. Transport layer breaks the message (data) into small units so that they are handled more efficiently by the network layer.
6. Transport layer also arrange the packets to be sent, in sequence.

Layer 5: Application Layer

The TCP/IP specifications described a lot of applications that were at the top of the protocol stack. Some of them were TELNET, FTP, SMTP, DNS etc.

1.TELNET is a two-way communication protocol which allows connecting to a remote machine and run applications on it.

2.FTP(File Transfer Protocol) is a protocol, that allows File transfer amongst computer users connected over a network. It is reliable, simple and efficient.

3.SMTP(Simple Mail Transport Protocol) is a protocol, which is used to transport electronic mail between a source and destination, directed via a route.

4.DNS(Domain Name Server) resolves an IP address into a textual address for Hosts connected over a network.

5.It allows peer entities to carry conversation.

6.It defines two end-to-end protocols: TCP and UDP

INTRODUCTION TO PHYSICAL LAYER: DATA AND SIGNALS

INTRODUCTION

Physical layer is the lowest layer in the OSI reference model. Information can be transmitted or received using cables. The main purpose of the physical layer is to transmit information (data or text or even animated pictures) from a source to a destination

. For example, if you want to transmit a photograph from a source computer to a destination computer, it is as a stream of bits 0s and 1s. Then this stream is reconverted as energy in the form of electromagnetic signals. Then only these electromagnetic signals can be transmitted over the network, through a transmission medium. The destination computer receives these signals. They are encoded to the original form.

Analog and digital signals:

There are two types of signals, namely analog (also called continuous) and digital (also called discrete). Analog quantities are used for measurement, whereas digital quantities are used for counting. For example, you can go to a provision shop and ask for 1.25 kg of sugar. But, you cannot ask for 1.25 first one is a continuous one and the second one is The measurement. Graphs of these two types of signals are shown below.

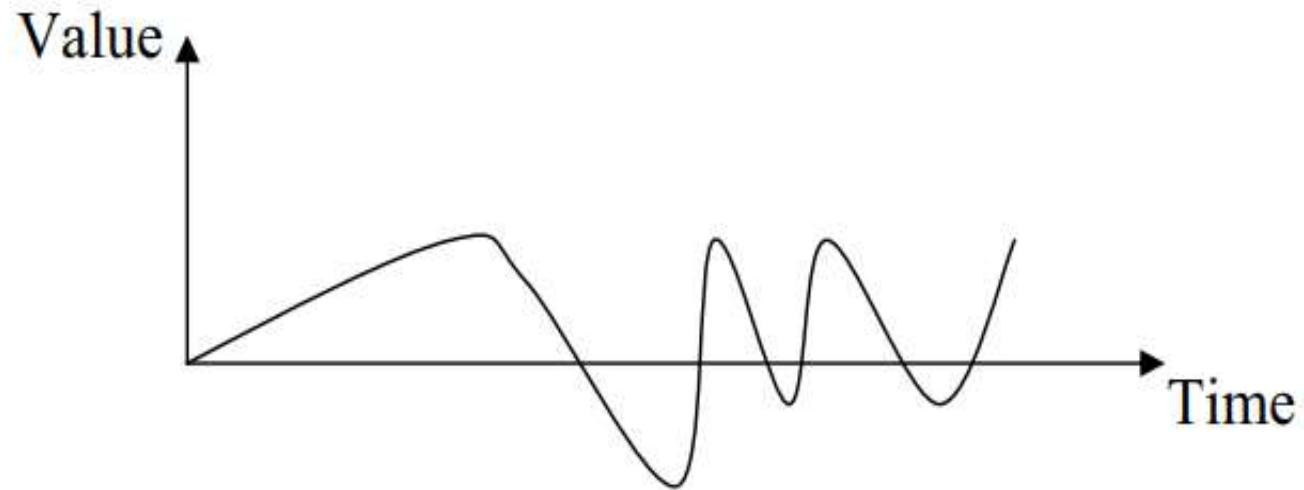


Fig. 2.1 An analog signal.

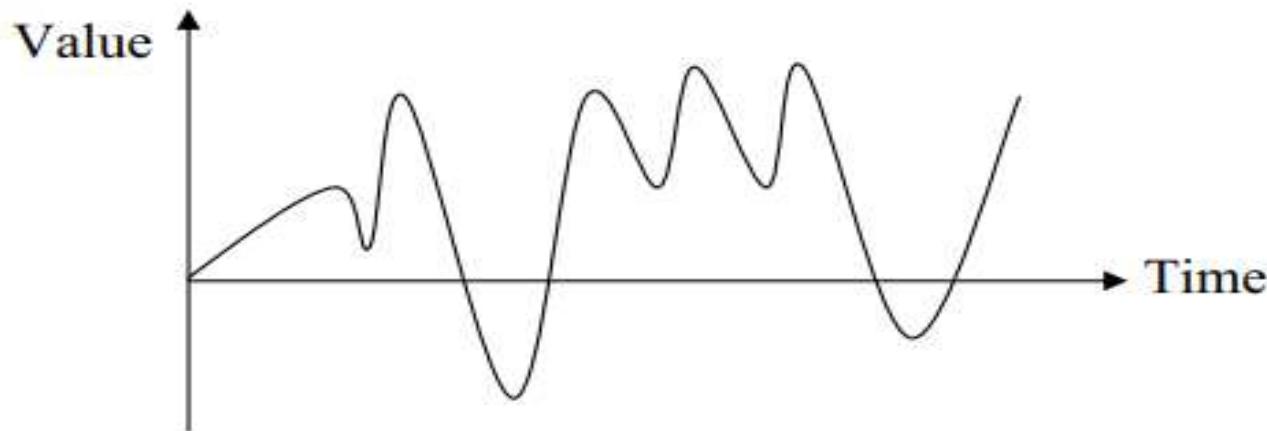


Fig. 2.5 A non-periodic analog signal.

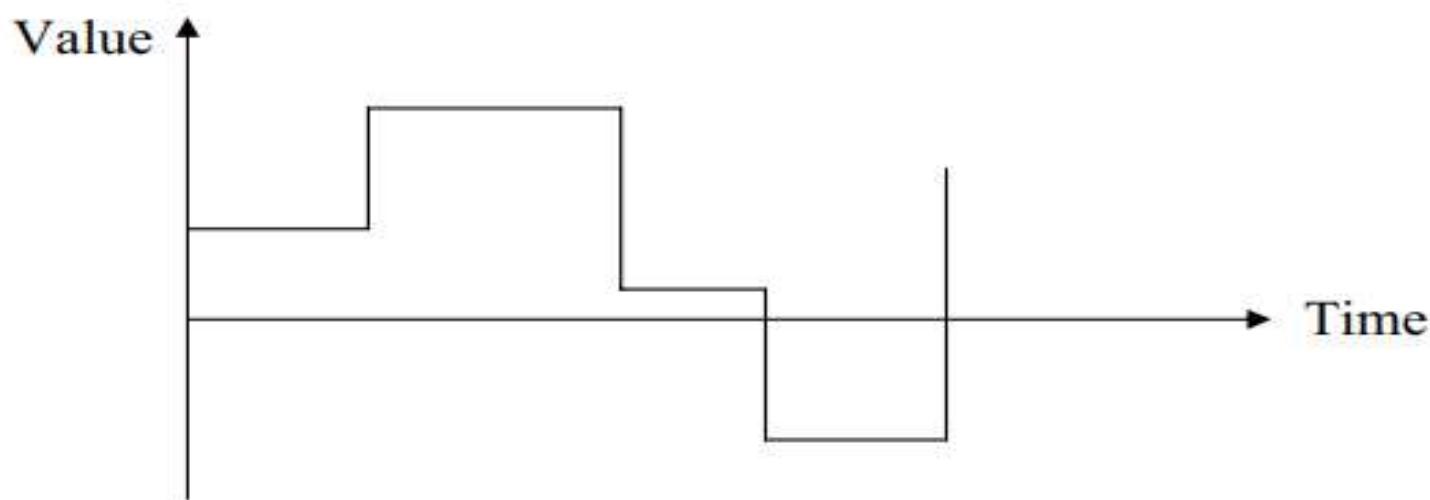


Fig. 2.6 A non-periodic digital signal.

Phase

Phase expresses the position of the waveform relative to time zero and is measured either in degrees or in radians.

The main issues in transmitting data are:

- **Line configuration:** the line configuration is concerned about the mode of connecting devices, about the number of devices that use the transmission line about the problem whether the lines to be shared or limited just to two devices and also whether line is available for transmission or not.
- **Data transmission mode:** the transmission mode whether it is one way or two is to be specified.
- **Topology:** The topology and path used in arranging network devices is to be made clear.
- **Signals:** The type of signals used is to be specified
- **Encoding:** This aspect is about the encoding methods used
- **Interface:** Issues regarding the controls to be imposed on what information and its amount that can be shared and the method of sharing information by two nearby devices have to be resolved.
- **Medium:** The type of the physical medium used is to be decided.

DATA RATE LIMITS

How fast we can send data, in bits per second, over a channel. - Two theoretical formulas were developed to calculate the data rate: Nyquist for a noiseless channel Shannon for a noisy channel.

Noiseless Channel: Nyquist Rate:

For a noiseless channel, the Nyquist bit rate formula defines the theoretical maximum bit rate.

$$\text{Bitrate} = 2 \times \text{bandwidth} \times \log_2^L$$

Ex1: Consider a noiseless channel with a bandwidth of 3000 Hz transmitting a signal with two signal levels. The maximum bit rate can be calculated as

$$\text{Bitrate} = 2 \times 3000 \times \log_2^2 = 6000 \text{ bps}$$

Noisy Channel: Shannon Capacity

In reality, we cannot have a noiseless channel; the channel is always noisy. In 1944, Claude Shannon introduced a formula, called the Shannon capacity, to determine the theoretical highest data rate for a noisy channel:

$$\text{Capacity} = \text{bandwidth} \times \log_2^{(1+SNR)}$$

Consider an extremely noisy channel in which the value of the signal-to-noise ratio is almost zero. In other words, the noise is so strong that the signal is faint. For this channel the capacity C is calculated as

$$C = B \log_2^{(1+SNR)} = B \log_2^{(1+0)} = B \log_2^1 = B \times 0 = 0$$

PERFORMANCE

One important issue in networking is the performance of the network— how good is it?

Bandwidth: One characteristic that measures network performance - can be used in two different contexts with two different measuring values:

- bandwidth in hertz
- bandwidth in bits per second

Throughput: a measure of how fast we can actually send data through a network.

The latency or delay defines how long it takes for an entire message to completely arrive at the destination from the time the first bit is sent out from the source.

We can say that latency is made of four components: propagation time, transmission time, queuing time and processing delay.

Latency= propagation time + transmission Time +queuing Time + Processing Delay.

Example: A network with bandwidth of 10 Mbps can pass only an average of 12,000 frames per minute with each frame carrying an average of 10,000 bits. What is the throughput of this network?

Solution

We can calculate the throughput as

$$\text{Throughput} = (12000 \times 10000)/60 = 2\text{Mbps.}$$

The throughput is almost one-fifth of the bandwidth in this case.

UNIT- II

Introduction to Data Link Layer: Introduction, Link layer addressing- Error detection and Correction: Cyclic codes, Checksum

Data link control: DLC Services- Data link layer protocols, HDLC, Point to Point Protocol- Media Access control:-Random Access -Controlled Access- Channelization

UNIT-2

DATA LINK LAYER

Introduction

The Data Link Layer break the bit stream into discrete frames and compute the checksum for each frame. When a Frame arrives at the destination, the checksum is recomputed.

DATA LINK LAYER FUNCTIONS (SERVICES)

1. Providing services to the network layer:

1 Unacknowledged connectionless service

Appropriate for low error rate and real-time traffic. Ex: Ethernet

2. Acknowledged connectionless service

Useful in unreliable channels, WiFi. Ack/Timer/Resend

3. Acknowledged connection-oriented service

Guarantee frames are received exactly once and in the right order.

Appropriate over long, unreliable links such as a satellite channel or a long distance telephone circuit

2. Framing:

Frames are the streams of bits received from the network layer into manageable data units. This division of stream of bits is done by Data Link Layer.

3. Physical Addressing

The Data Link layer adds a header to the frame in order to define physical address of the sender or receiver of the frame, if the frames are to be distributed to different systems on the network.

4. Flow Control:

A receiving node can receive the frames at a faster rate than it can process the frame. Without flow control, the receiver's buffer can overflow, and frames can get lost.

5. Error Control:

Error control is achieved by adding a trailer at the end of the frame.

Duplication of frames are also prevented by using this mechanism.

Data Link Layers adds mechanism to prevent duplication of frames.

Error detection: Errors can be introduced by signal attenuation and noise.

Data Link Layer protocol provides a mechanism to detect one or more errors. This is achieved by adding error detection bits in the frame and then receiving node can perform an error check.

Error correction: Error correction is similar to the Error detection, except that receiving node not only detects the errors but also determine where the errors have occurred in the frame.

6. Access Control

Protocols of this layer determine which of the devices has control over the link at any given time, when two or more devices are connected to the same link

7. Reliable delivery

Data Link Layer provides a reliable delivery service, i.e., transmits the network layer datagram without any error. A reliable delivery service is accomplished with transmissions and acknowledgements. A data link layer mainly provides the reliable delivery service over the links as they have higher error rates and they can be corrected locally, link at which an error occurs rather than forcing to retransmit the data.

8. Half-Duplex & Full-Duplex

In a Full-Duplex mode, both the nodes can transmit the data at the same time. In a Half-Duplex mode, only one node can transmit the data at the same time.

LINK LAYER ADDRESSING

Address Resolution Protocol

Address Resolution Protocol (ARP) is a communication protocol used to find the MAC (Media Access Control) address of a device from its IP address. This protocol is used when a device wants to communicate with another device on a Local Area Network or Ethernet.

Types of ARP

There are four types of Address Resolution Protocol, which is given below:

Proxy ARP

Gratuitous ARP

Reverse ARP (RARP)

Inverse ARP

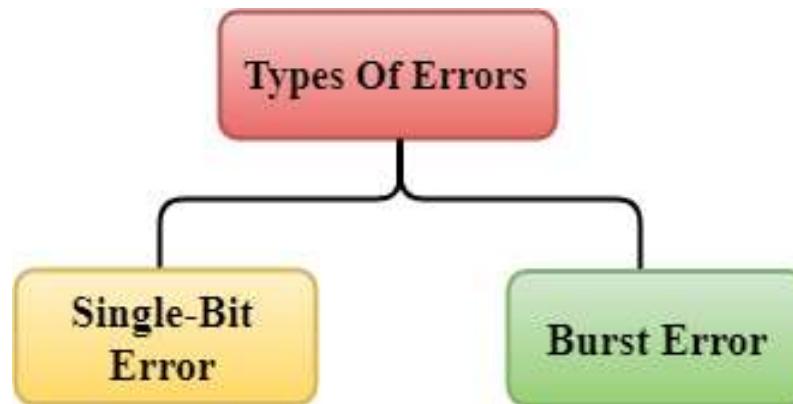
Proxy ARP - Proxy ARP is a method through which a Layer 3 devices may respond to ARP requests for a target that is in a different network from the sender. The Proxy [ARP](#) configured router responds to the ARP and map the MAC address of the router with the target [IP](#) address and fool the sender that it is reached at its destination.

Gratuitous ARP - Gratuitous ARP is an [ARP request](#) of the host that helps to identify the duplicate IP address. It is a broadcast request for the IP address of the router. If an ARP request is sent by a switch or router to get its IP address and no ARP responses are received, so all other nodes cannot use the IP address allocated to that switch or router.

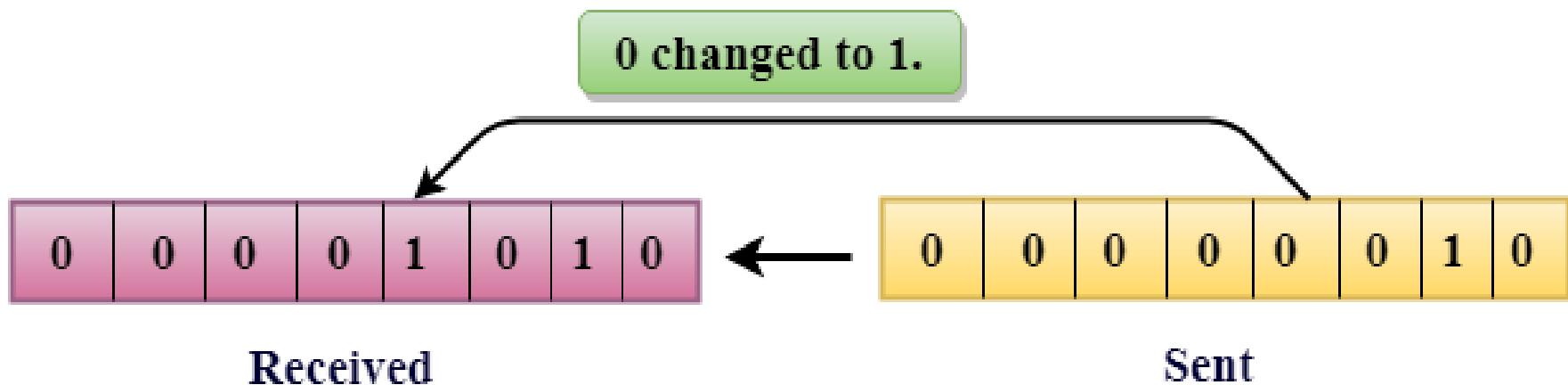
Reverse ARP (RARP) - It is a networking protocol used by the client system in a local area network (LAN) to request its IPv4 address from the ARP gateway router table. A table is created by the network administrator in the gateway-router that is used to find out the MAC address to the corresponding IP address.

Inverse ARP (InARP) - Inverse ARP is inverse of the ARP, and it is used to find the IP addresses of the nodes from the data link layer addresses. These are mainly used for the frame relays, and ATM networks, where Layer 2 virtual circuit addressing are often acquired from Layer 2 signaling. When using these virtual circuits, the relevant Layer 3 addresses are available.

ARP conversions Layer 3 addresses to Layer 2 addresses. However, its opposite address can be defined by InARP. The InARP has a similar packet format as ARP, but operational codes are different.

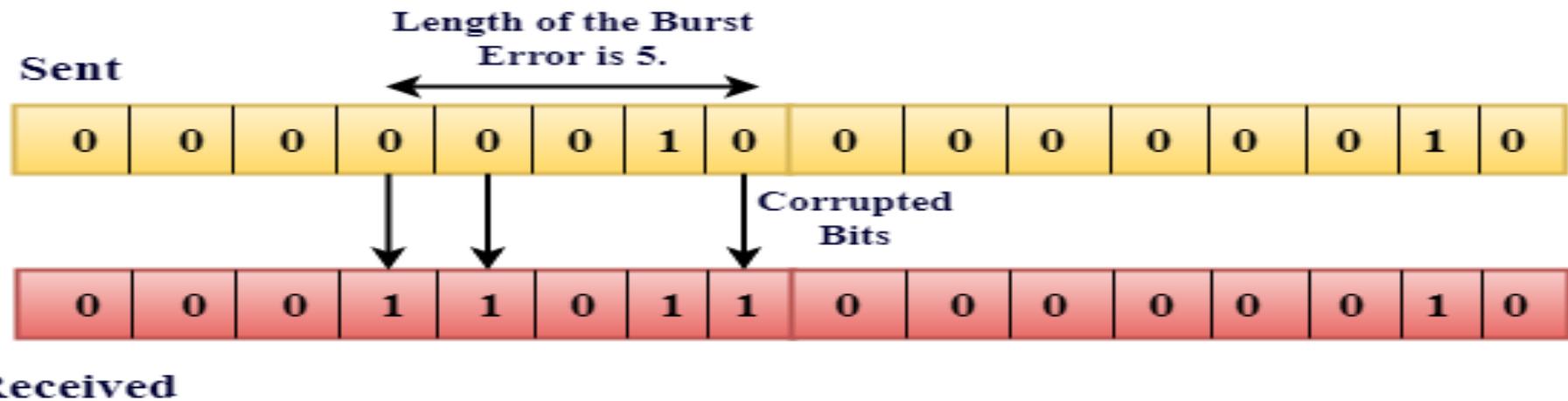


Single-Bit Error does not appear more likely in Serial Data Transmission. For example, Sender sends the data at 10 Mbps, this means that the bit lasts only for 1 ?s and for a single-bit error to occurred, a noise must be more than 1 ?s.



Burst Error:

The two or more bits are changed from 0 to 1 or from 1 to 0 is known as Burst Error. The Burst Error is determined from the first corrupted bit to the last corrupted bit.



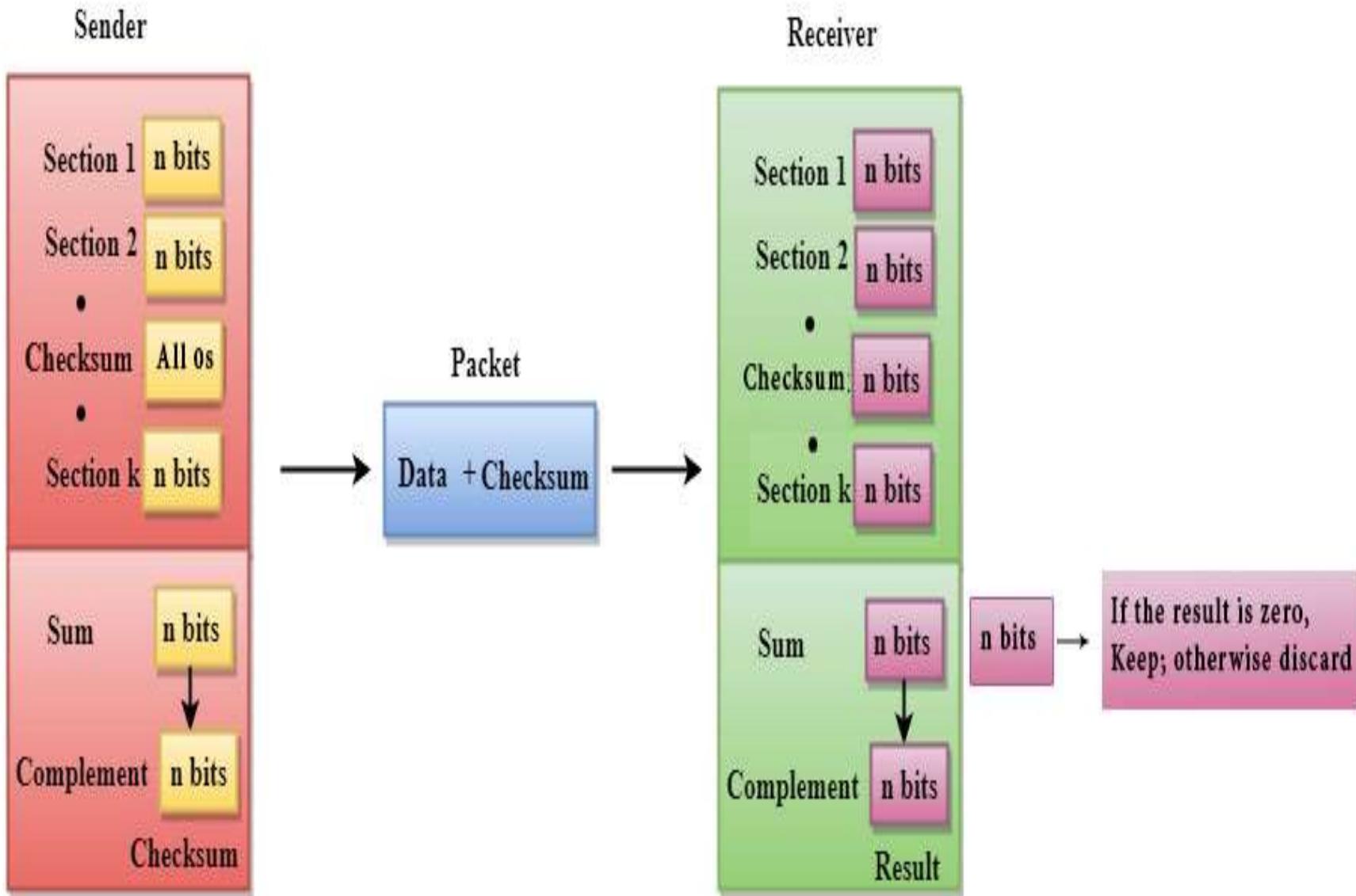
CHECKSUM

A Checksum is an error detection technique based on the concept of redundancy.

It is divided into two parts:

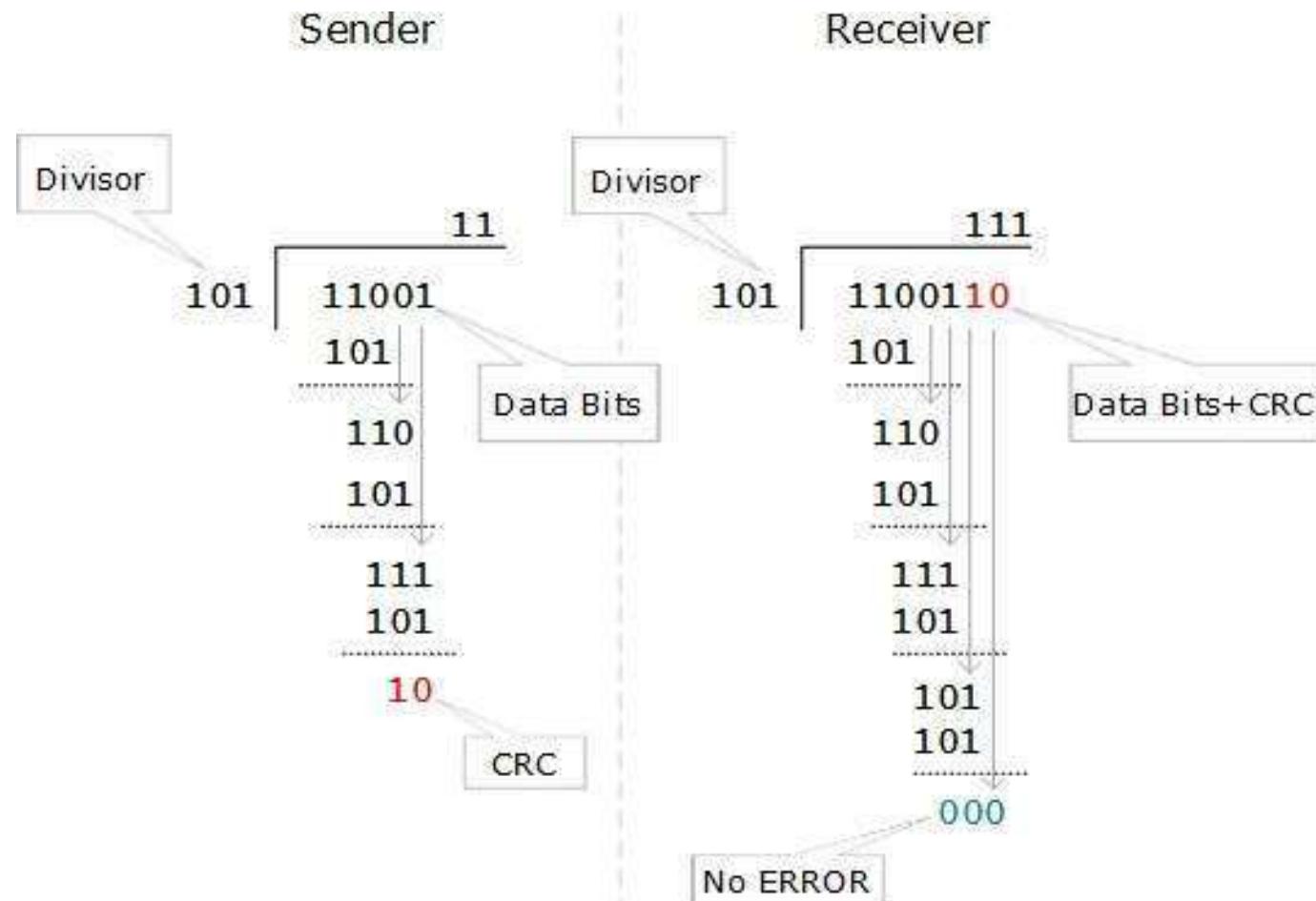
Checksum Generator

A Checksum is generated at the sending side. Checksum generator subdivides the data into equal segments of n bits each, and all these segments are added together by using one's complement arithmetic. The sum is complemented and appended to the original data, known as checksum field. The extended data is transmitted across the network.



Cyclic Redundancy Check (CRC)

CRC is a different approach to detect if the received frame contains valid data. This technique involves binary division of the data bits being sent. The divisor is generated using polynomials. The sender performs a division operation on the bits being sent and calculates the remainder.



Forward Error Correction (FEC)

Forward error correction (FEC) is an error correction technique to detect and correct a limited number of errors in transmitted data without the need for retransmission.

In this method, the sender sends a redundant error-correcting code along with the data frame. The receiver performs necessary checks based upon the additional redundant bits. If it finds that the data is free from errors, it executes error-correcting code that generates the actual frame. It then removes the redundant bits before passing the message to the upper layers.

Error Correction Codes for FEC

Error correcting codes for forward error corrections can be broadly categorized into two types, namely, block codes and convolution codes.

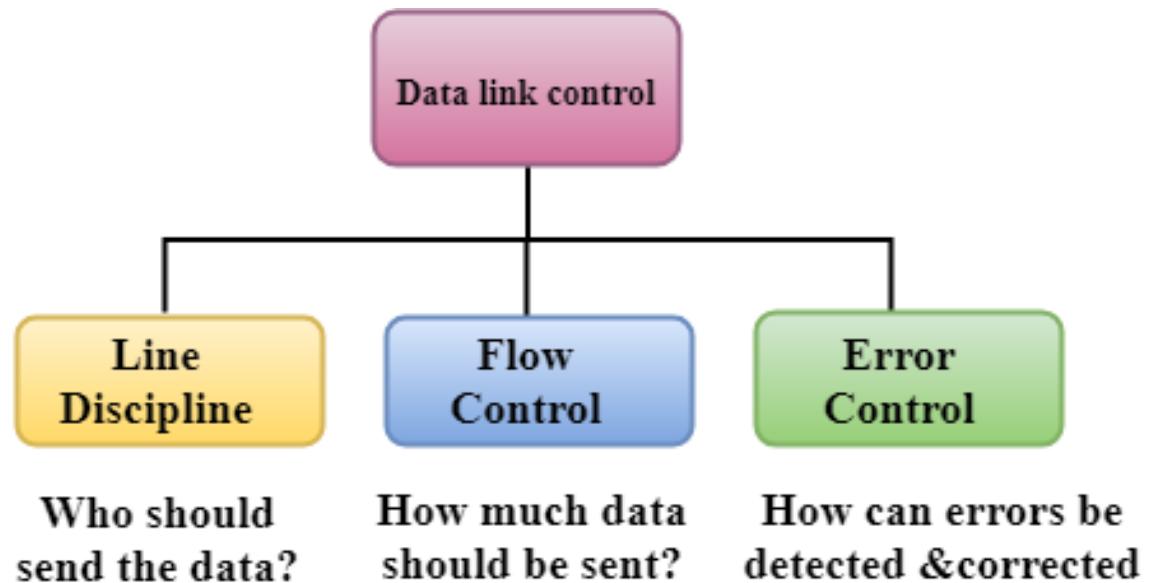
- **Block codes** – The message is divided into fixed-sized blocks of bits to which redundant bits are added for error correction.
- **Convolutional codes** – The message comprises of data streams of arbitrary length and parity symbols are generated by the sliding application of a Boolean function to the data stream.

Data Link Controls

Data Link Control is the service provided by the Data Link Layer to provide reliable data transfer over the physical medium. For example, In the half-duplex transmission mode, one device can only transmit the data at a time. If both the devices at the end of the links transmit the data simultaneously, they will collide and leads to the loss of the information. The Data link layer provides the coordination among the devices so that no collision occurs.

The Data link layer provides three functions:

Line discipline
Flow Control
Error Control



Line Discipline

Line Discipline is a functionality of the Data link layer that provides the coordination among the link systems. It determines which device can send, and when it can send the data.

Line Discipline can be achieved in two ways:

ENQ/ACK

Poll/select

ENq/ACK

ENQ/ACK stands for Enquiry/Acknowledgement is used when there is no wrong receiver available on the link and having a dedicated path between the two devices so that the device capable of receiving the transmission is the intended one.

ENQ/ACK coordinates which device will start the transmission and whether the recipient is ready or not.

Working of ENQ/ACK

The transmitter transmits the frame called an Enquiry (ENQ) asking whether the receiver is available to receive the data or not.

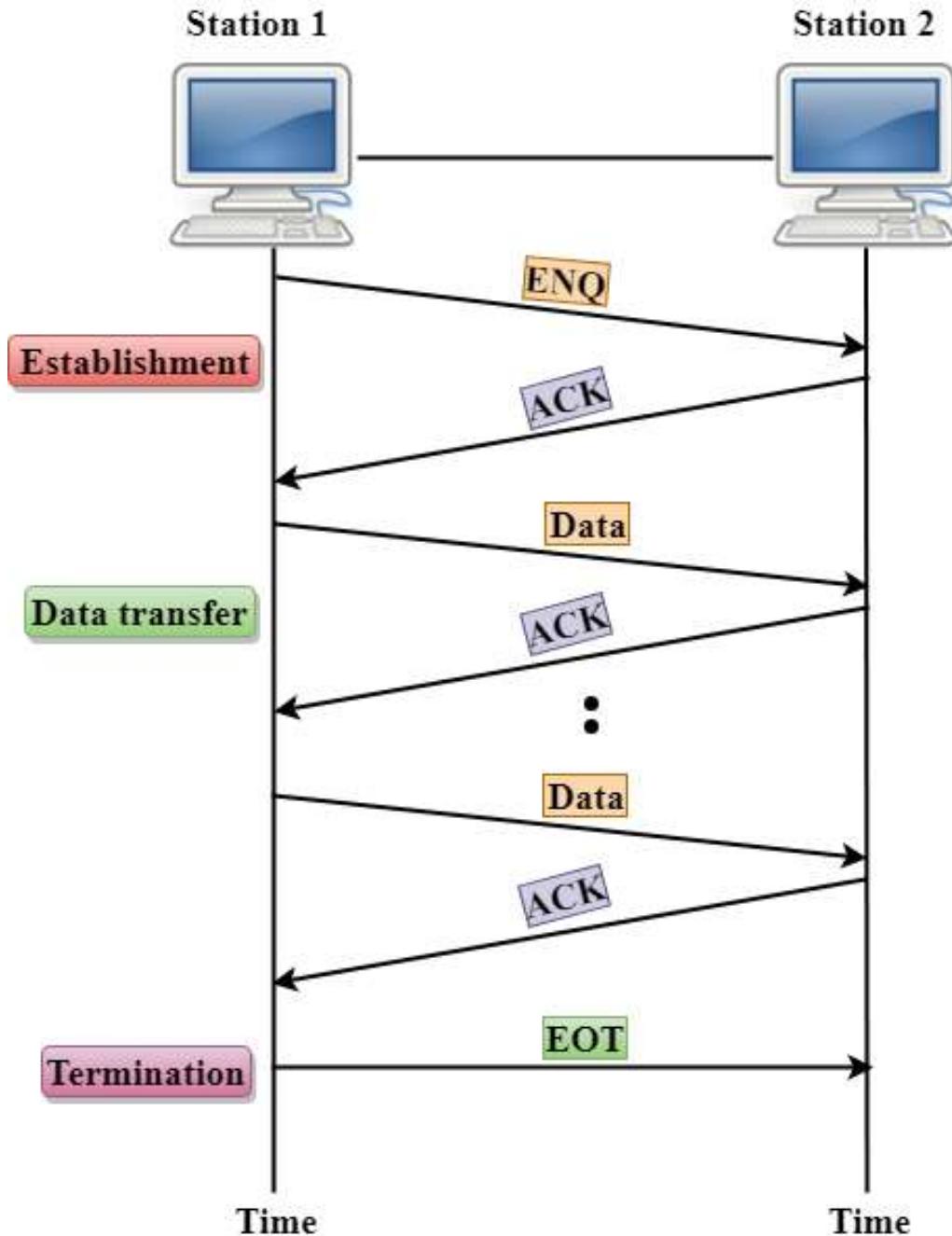
The receiver responses either with the positive acknowledgement(ACK) or with the negative acknowledgement(NACK) where positive acknowledgement means that the receiver is ready to receive the transmission and negative acknowledgement means that the receiver is unable to accept the transmission.

Following are the responses of the receiver:

If the response to the ENQ is positive, the sender will transmit its data, and once all of its data has been transmitted, the device finishes its transmission with an EOT (END-of-Transmission) frame.

If the response to the ENQ is negative, then the sender disconnects and restarts the transmission at another time.

If the response is neither negative nor positive, the sender assumes that the ENQ frame was lost during the transmission and makes three attempts to establish a link before giving up.



Poll/Select

The Poll/Select method of line discipline works with those topologies where one device is designated as a primary station, and other devices are secondary stations.

Working of Poll/Select

In this, the primary device and multiple secondary devices consist of a single transmission line, and all the exchanges are made through the primary device even though the destination is a secondary device.

The primary device has control over the communication link, and the secondary device follows the instructions of the primary device.

The primary device determines which device is allowed to use the communication channel. Therefore, we can say that it is an initiator of the session.

If the primary device wants to receive the data from the secondary device, it asks the secondary device that they anything to send, this process is known as polling.

If the primary device wants to send some data to the secondary device, then it tells the target secondary to get ready to receive the data, this process is known as selecting.

Select

The select mode is used when the primary device has something to send.

When the primary device wants to send some data, then it alerts the secondary device for the upcoming transmission by transmitting a Select (SEL) frame, one field of the frame includes the address of the intended secondary device.

When the secondary device receives the SEL frame, it sends an acknowledgement that indicates the secondary ready status.

If the secondary device is ready to accept the data, then the primary device sends two or more data frames to the intended secondary device. Once the data has been transmitted, the secondary sends an acknowledgement specifies that the data has been received.

.

Primary Device



Secondary A



Secondary B



Secondary C

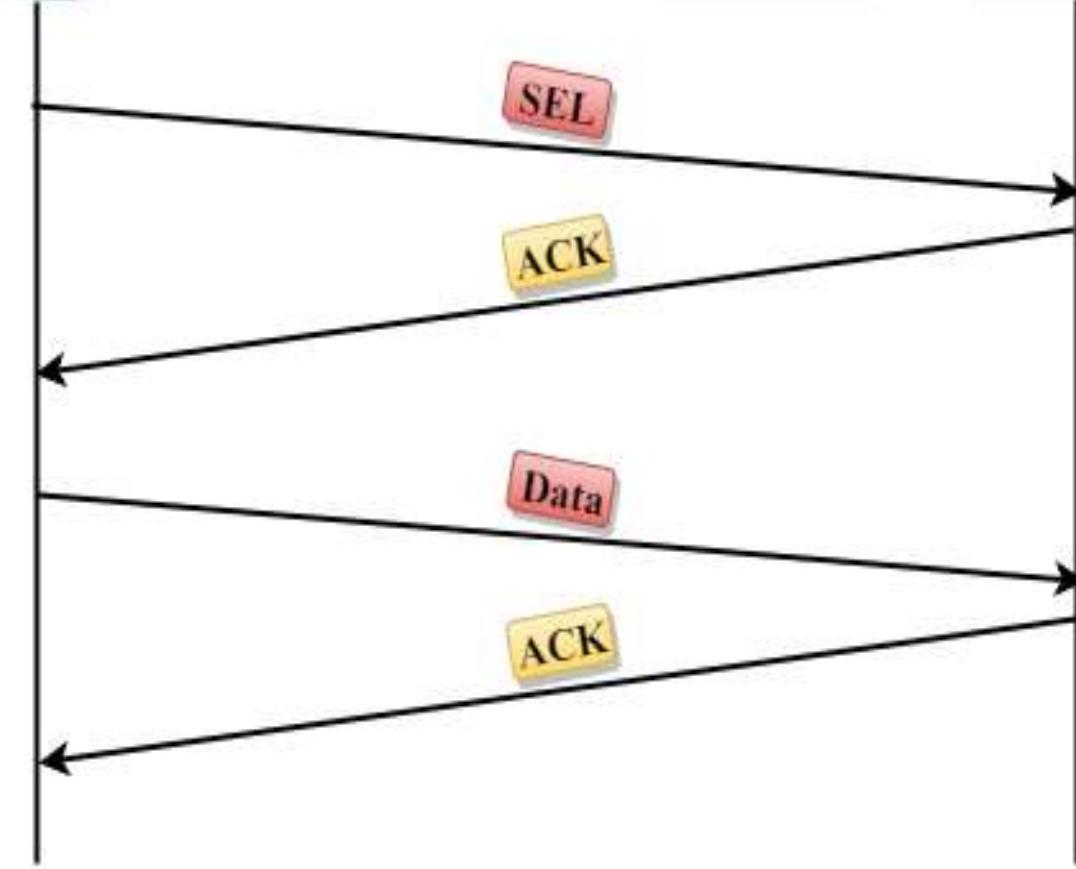


SEL

ACK

Data

ACK

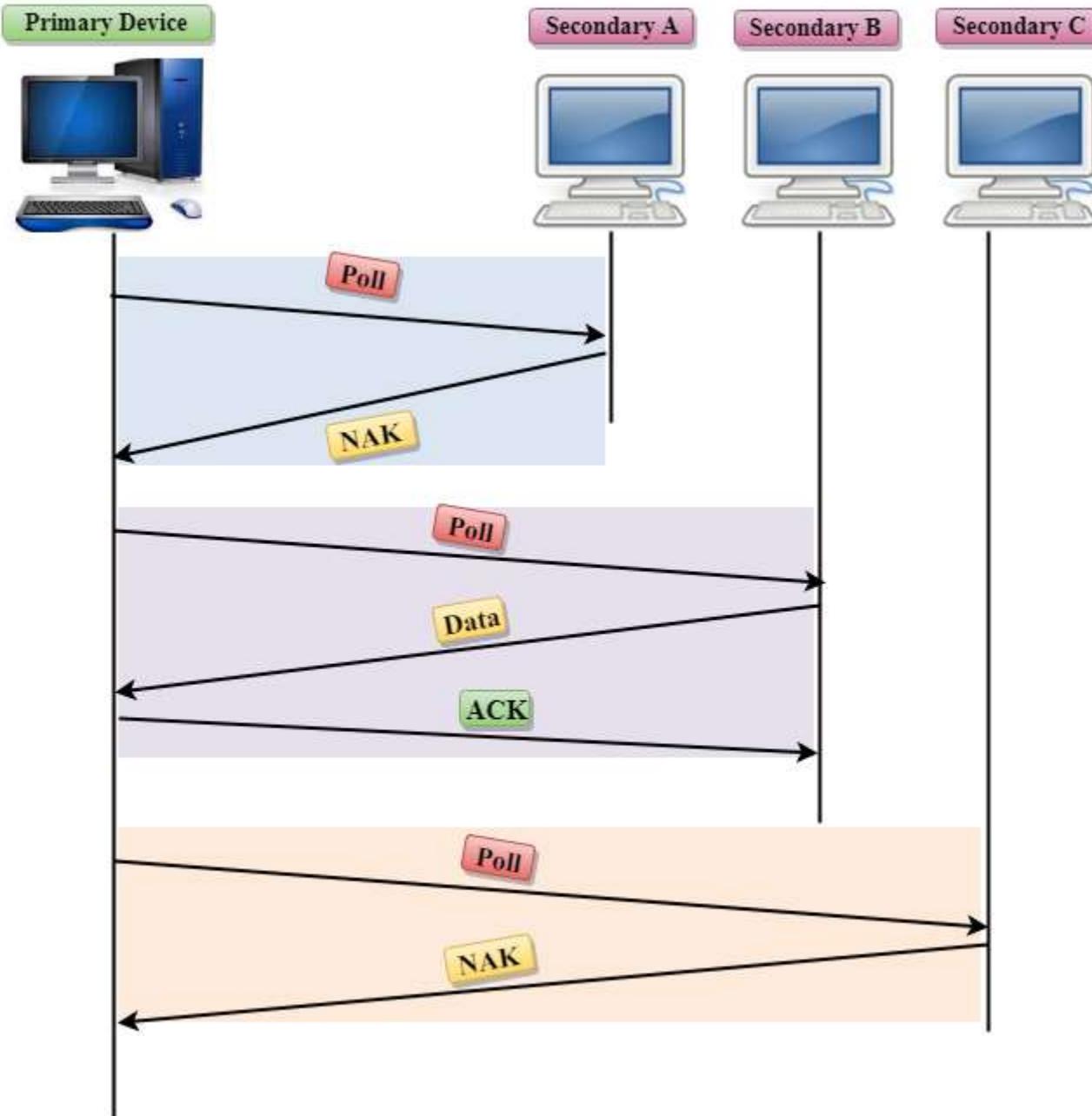


Poll

The Poll mode is used when the primary device wants to receive some data from the secondary device.

When a primary device wants to receive the data, then it asks each device whether it has anything to send.

Firstly, the primary asks (poll) the first secondary device, if it responds with the NACK (Negative Acknowledgement) means that it has nothing to send. Now, it approaches the second secondary device, it responds with the ACK means that it has the data to send. The secondary device can send more than one frame one after another or sometimes it may be required to send ACK before sending each one, depending on the type of the protocol being used.



Flow Control

It is a set of procedures that tells the sender how much data it can transmit before the data overwhelms the receiver.

The receiving device has limited speed and limited memory to store the data. Therefore, the receiving device must be able to inform the sending device to stop the transmission temporarily before the limits are reached.

It requires a buffer, a block of memory for storing the information until they are processed.

Two methods have been developed to control the flow of data:

Stop-and-wait

Sliding window

Stop-and-wait

In the Stop-and-wait method, the sender waits for an acknowledgement after every frame it sends.

When acknowledgement is received, then only next frame is sent.

The process of alternately sending and waiting of a frame continues until the sender transmits the EOT (End of transmission) frame.

Advantage of Stop-and-wait

The Stop-and-wait method is simple as each frame is checked and acknowledged before the next frame is sent.

Disadvantage of Stop-and-wait

Stop-and-wait technique is inefficient to use as each frame must travel across all the way to the receiver, and an acknowledgement travels all the way before the next frame is sent. Each frame sent and received uses the entire time needed to traverse the link.

Sliding Window

The Sliding Window is a method of flow control in which a sender can transmit the several frames before getting an acknowledgement.

In Sliding Window Control, multiple frames can be sent one after the another due to which capacity of the communication channel can be utilized efficiently.

A single ACK acknowledge multiple frames.

Sliding Window refers to imaginary boxes at both the sender and receiver end.

The window can hold the frames at either end, and it provides the upper limit on the number of frames that can be transmitted before the acknowledgement.

Frames can be acknowledged even when the window is not completely filled.

The window has a specific size in which they are numbered as modulo-n means that they are numbered from 0 to n-1. For example, if $n = 8$, the frames are numbered from 0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1.....

The size of the window is represented as $n-1$. Therefore, maximum $n-1$ frames can be sent before acknowledgement. When the receiver sends the ACK, it includes the number of the next frame that it wants to receive. For example, to acknowledge the string of frames ending with frame number 4, the receiver will send the ACK containing the number 5. When the sender sees the ACK with the number 5, it got to know that the frames from 0 through 4 have been received.

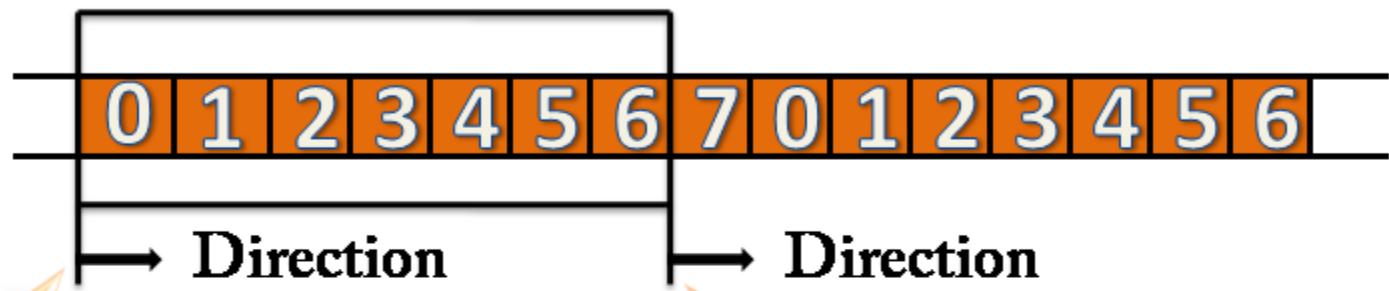
Sender Window

At the beginning of a transmission, the sender window contains $n-1$ frames, and when they are sent out, the left boundary moves inward shrinking the size of the window. For example, if the size of the window is w if three frames are sent out, then the number of frames left out in the sender window is $w-3$.

Once the ACK has arrived, then the sender window expands to the number which will be equal to the number of frames acknowledged by ACK.

For example, the size of the window is 7, and if frames 0 through 4 have been sent out and no acknowledgement has arrived, then the sender window contains only two frames, i.e., 5 and 6. Now, if ACK has arrived with a number 4 which means that 0 through 3 frames have arrived undamaged and the sender window is expanded to include the next four frames. Therefore, the sender window contains six frames (5,6,7,0,1,2).

Senderwindow



This wall moves to the right
When a frame is sent.

This wall moves to the right
When an ACK is received.

Receiver Window

At the beginning of transmission, the receiver window does not contain n frames, but it contains n-1 spaces for frames.

When the new frame arrives, the size of the window shrinks.

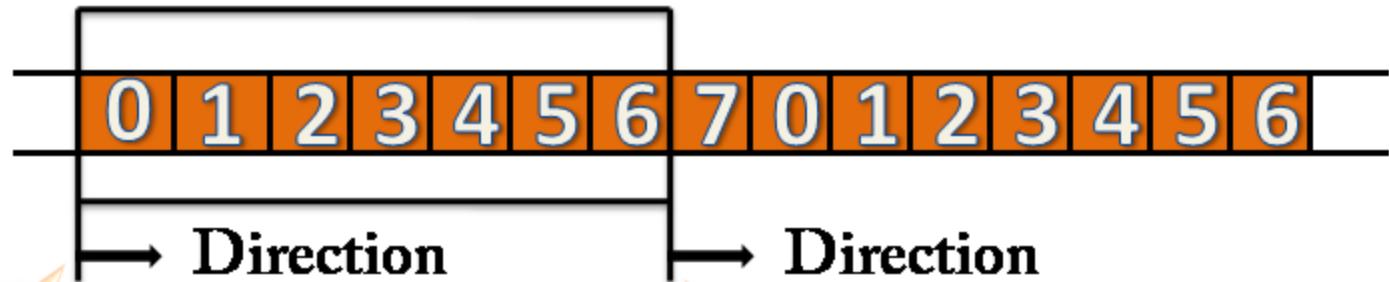
The receiver window does not represent the number of frames received, but it represents the number of frames that can be received before an ACK is sent. For example, the size of the window is w, if three frames are received then the number of spaces available in the window is (w-3).

Once the acknowledgement is sent, the receiver window expands by the number equal to the number of frames acknowledged.

Suppose the size of the window is 7 means that the receiver window contains seven spaces for seven frames.

If the one frame is received, then the receiver window shrinks and moving the boundary from 0 to 1. In this way, window shrinks one by one, so window now contains the six spaces. If frames from 0 through 4 have sent, then the window contains two spaces before an acknowledgement is sent.

Receiver window



This wall moves to the right
When a frame is received.

This wall moves to the right
When an ACK is sent.

Error Control

Error Control is a technique of error detection and retransmission.

Categories of Error Control:



Stop-and-wait ARQ

Stop-and-wait ARQ is a technique used to retransmit the data in case of damaged or lost frames.

This technique works on the principle that the sender will not transmit the next frame until it receives the acknowledgement of the last transmitted frame

Four features are required for the retransmission:

The sending device keeps a copy of the last transmitted frame until the acknowledgement is received. Keeping the copy allows the sender to retransmit the data if the frame is not received correctly.

Both the data frames and the ACK frames are numbered alternately 0 and 1 so that they can be identified individually. Suppose data 1 frame acknowledges the data 0 frame means that the data 0 frame has been arrived correctly and expects to receive data 1 frame.

If an error occurs in the last transmitted frame, then the receiver sends the NAK frame which is not numbered. On receiving the NAK frame, sender retransmits the data.

It works with the timer. If the acknowledgement is not received within the allotted time, then the sender assumes that the frame is lost during the transmission, so it will retransmit the frame.

Two possibilities of the retransmission:

Damaged Frame: When the receiver receives a damaged frame, i.e., the frame contains an error, then it returns the NAK frame. For example, when the data 0 frame is sent, and then the receiver sends the ACK 1 frame means that the data 0 has arrived correctly, and transmits the data 1 frame. The sender transmits the next frame: data 1. It reaches undamaged, and the receiver returns ACK 0. The sender transmits the next frame: data 0. The receiver reports an error and returns the NAK frame. The sender retransmits the data 0 frame.

Lost Frame: Sender is equipped with the timer and starts when the frame is transmitted. Sometimes the frame has not arrived at the receiving end so that it can be acknowledged neither positively nor negatively. The sender waits for acknowledgement until the timer goes off. If the timer goes off, it retransmits the last transmitted frame.

Sliding Window ARQ

SlidingWindow ARQ is a technique used for continuous transmission error control.

Three Features used for retransmission:

In this case, the sender keeps the copies of all the transmitted frames *until they have been acknowledged*. Suppose the frames from 0 through 4 have been transmitted, and the last acknowledgement was for frame 2, the sender has to keep the copies of frames 3 and 4 until they receive correctly.

The receiver can send either NAK or ACK depending on the conditions. The NAK frame tells the sender that the data have been received damaged. Since the sliding window is a continuous transmission mechanism, both ACK and NAK must be numbered for the identification of a frame. The ACK frame consists of a number that represents the next frame which the receiver expects to receive. The NAK frame consists of a number that represents the damaged frame.

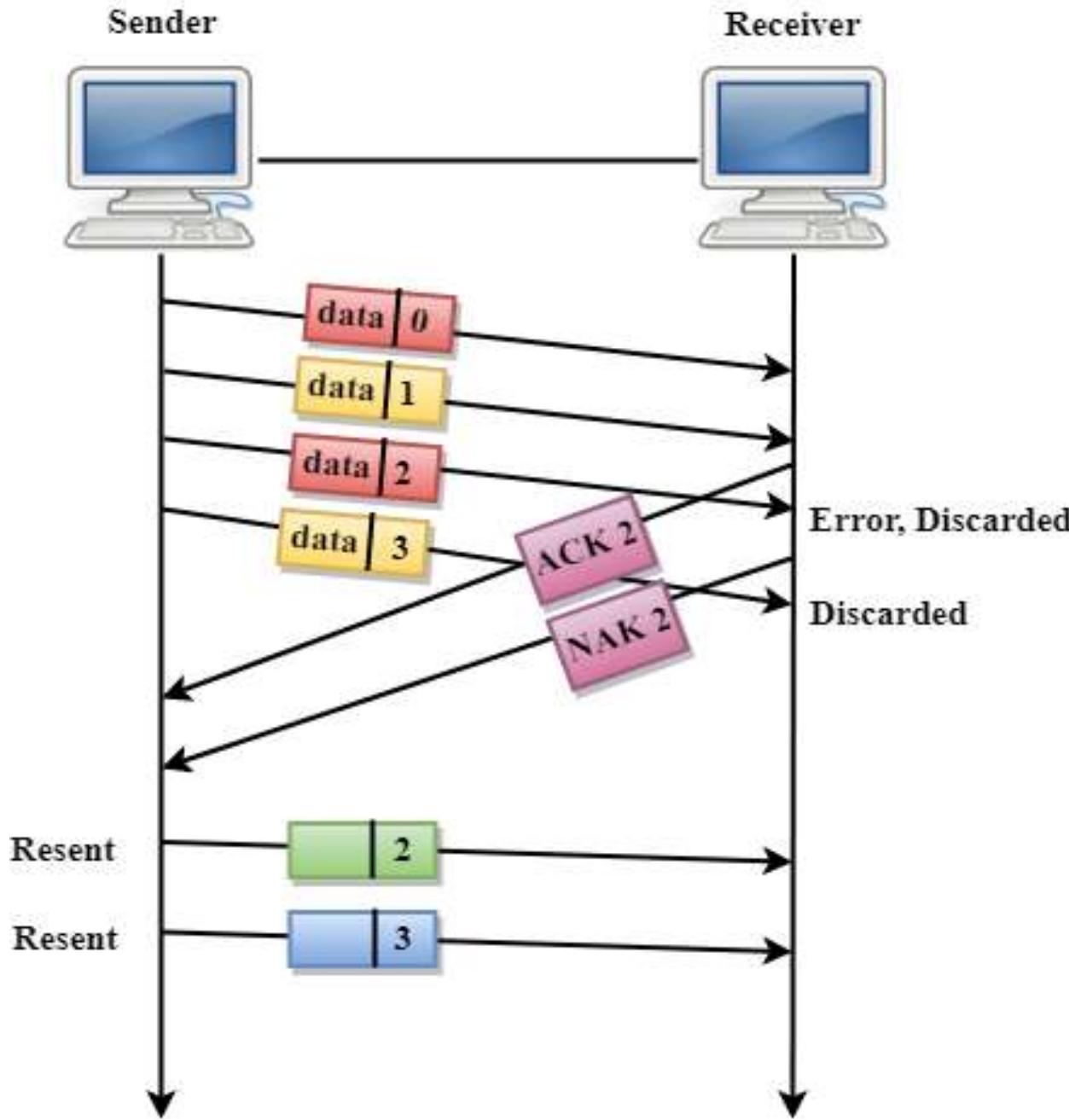
The sliding window ARQ is equipped with **the timer** to handle the lost acknowledgements. Suppose then $n-1$ frames have been sent before receiving any acknowledgement. The sender waits for the acknowledgement, so it starts the timer and waits before sending any more. If the allotted time runs out, the sender retransmits one or all the frames depending upon the protocol used.

Two protocols used in sliding window ARQ:

Go-Back-n ARQ: In Go-Back-N ARQ protocol, if one frame is lost or damaged, then it retransmits all the frames after which it does not receive the positive ACK.

Three possibilities can occur for retransmission:

Damaged Frame: When the frame is damaged, then the receiver sends a NAK frame.



In the above figure, three frames have been transmitted before an error discovered in the third frame. In this case, ACK 2 has been returned telling that the frames 0,1 have been received successfully without any error. The receiver discovers the error in data 2 frame, so it returns the NAK 2 frame. The frame 3 is also discarded as it is transmitted after the damaged frame. Therefore, the sender retransmits the frames 2,3.

Lost Data Frame: In Sliding window protocols, data frames are sent sequentially. If any of the frames is lost, then the next frame arrive at the receiver is out of sequence. The receiver checks the sequence number of each of the frame, discovers the frame that has been skipped, and returns the NAK for the missing frame. The sending device retransmits the frame indicated by NAK as well as the frames transmitted after the lost frame.

Lost Acknowledgement: The sender can send as many frames as the windows allow before waiting for any acknowledgement. Once the limit of the window is reached, the sender has no more frames to send; it must wait for the acknowledgement. If the acknowledgement is lost, then the sender could wait forever. To avoid such situation, the sender is equipped with the timer that starts counting whenever the window capacity is reached. If the acknowledgement has not been received within the time limit, then the sender retransmits the frame since the last ACK.

Selective-Reject ARQ

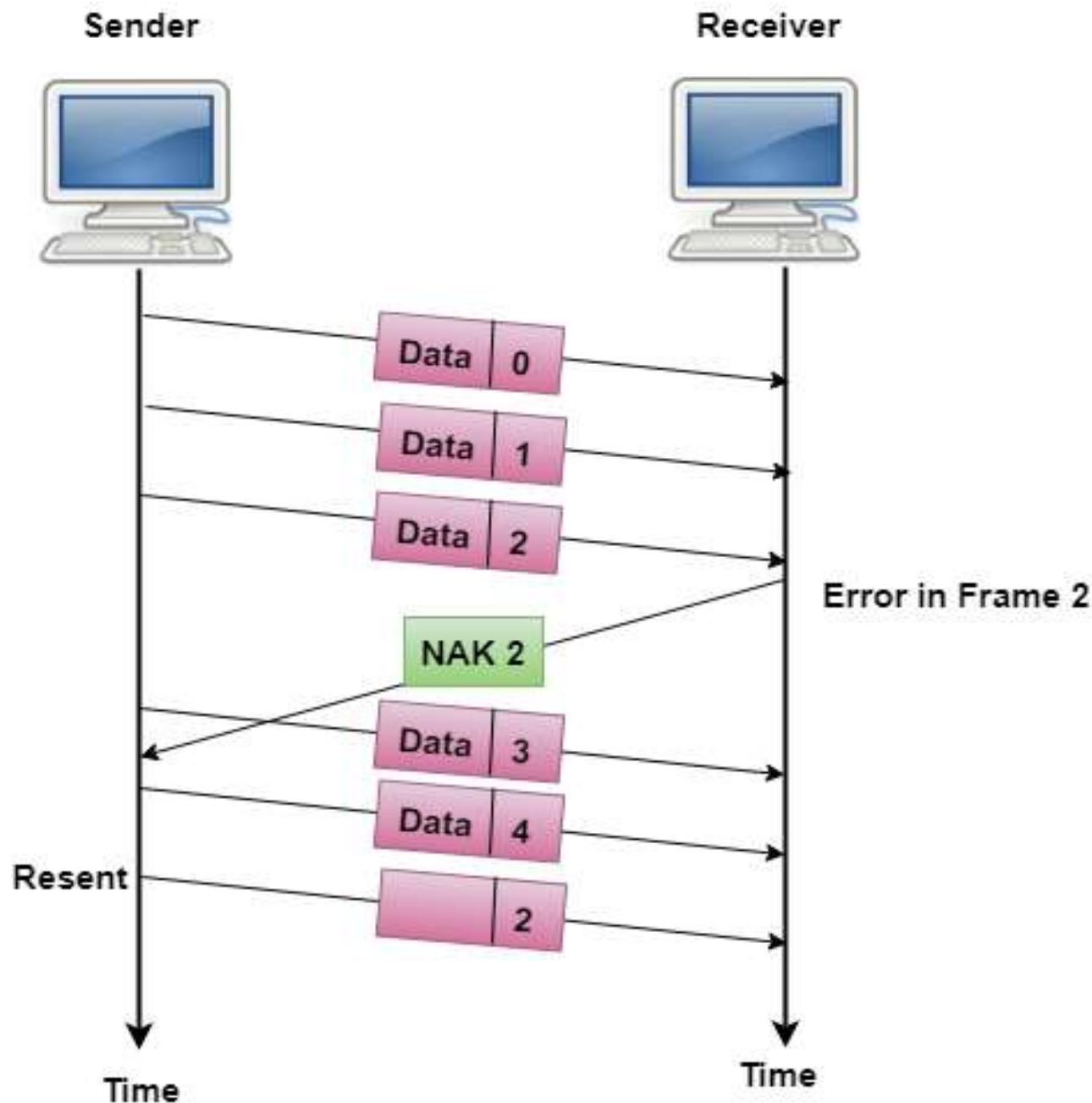
Selective-Reject ARQ technique is more efficient than Go-Back-n ARQ.

In this technique, **only those frames are retransmitted** for which negative acknowledgement (NAK) has been received.

The receiver storage buffer keeps all the damaged frames on hold until the frame in error is correctly received.

The receiver must have an appropriate logic for reinserting the frames in a correct order.

The sender must consist of a searching mechanism that selects only the requested frame for retransmission.



High-level Data Link Control (HDLC)

High-level Data Link Control (HDLC) is a group of communication protocols of the data link layer for transmitting data between network points or nodes. Since it is a data link protocol, data is organized into frames.

Transfer Modes

HDLC supports two types of transfer modes, normal response mode and asynchronous balanced mode.

Normal Response Mode (NRM) – Here, two types of stations are there, a primary station that send commands and secondary station that can respond to received commands. It is used for both point - to - point and multipoint communications.

Asynchronous Balanced Mode (ABM) – Here, the configuration is balanced, i.e. each station can both send commands and respond to commands. It is used for only point - to - point communications.

Normal Response Mode

Primary Station



Command

Response

Secondary Station



Point – to – point communication

Primary Station



Command

Secondary Stations



Response

Response

Multipoint communication

Asynchronous Balanced Mode

Station



Station



Command/ Response



Command/ Response



HDLC Frame

HDLC is a bit - oriented protocol where each frame contains up to six fields. The structure varies according to the type of frame. The fields of a HDLC frame are –

- **Flag** – It is an 8-bit sequence that marks the beginning and the end of the frame. The bit pattern of the flag is 01111110.
- **Address** – It contains the address of the receiver. If the frame is sent by the primary station, it contains the address(es) of the secondary station(s). If it is sent by the secondary station, it contains the address of the primary station. The address field may be from 1 byte to several bytes.
- **Control** – It is 1 or 2 bytes containing flow and error control information.
- **Payload** – This carries the data from the network layer. Its length may vary from one network to another.
- **FCS** – It is a 2 byte or 4 bytes frame check sequence for error detection. The standard code used is CRC (cyclic redundancy code)

HDLC Frame



Types of HDLC Frames

There are three types of HDLC frames. The type of frame is determined by the control field of the frame –

- **I-frame** – I-frames or Information frames carry user data from the network layer. They also include flow and error control information that is piggybacked on user data. The first bit of control field of I-frame is 0.
- **S-frame** – S-frames or Supervisory frames do not contain information field. They are used for flow and error control when piggybacking is not required. The first two bits of control field of S-frame is 10.
- **U-frame** – U-frames or Un-numbered frames are used for myriad miscellaneous functions, like link management. It may contain an information field, if required. The first two bits of control field of U-frame is 11.

HDLC Frame

I – Frame



S – Frame



U – Frame



Point-to-Point Protocol (PPP)

Point - to - Point Protocol (PPP) is a communication protocol of the data link layer that is used to transmit multiprotocol data between two directly connected (point-to-point) computers. It is a byte - oriented protocol that is widely used in broadband communications having heavy loads and high speeds.

Services Provided by PPP

The main services provided by Point - to - Point Protocol are –

- Defining the frame format of the data to be transmitted.
- Defining the procedure of establishing link between two points and exchange of data.
- Stating the method of encapsulation of network layer data in the frame.
- Stating authentication rules of the communicating devices.
- Providing address for network communication.
- Providing connections over multiple links.
- Supporting a variety of network layer protocols by providing a range of services.

Components of PPP

Point - to - Point Protocol is a layered protocol having three components –

- **Encapsulation Component** – It encapsulates the datagram so that it can be transmitted over the specified physical layer.
- **Link Control Protocol (LCP)** – It is responsible for establishing, configuring, testing, maintaining and terminating links for transmission. It also imparts negotiation for set up of options and use of features by the two endpoints of the links.
- **Authentication Protocols (AP)** – These protocols authenticate endpoints for use of services. The two authentication protocols of PPP are –
 - Password Authentication Protocol (PAP)
 - Challenge Handshake Authentication Protocol (CHAP)

- **Network Control Protocols (NCPs)** – These protocols are used for negotiating the parameters and facilities for the network layer. For every higher-layer protocol supported by PPP, one NCP is there. Some of the NCPs of PPP are –
 - Internet Protocol Control Protocol (IPCP)
 - OSI Network Layer Control Protocol (OSINLCP)
 - Internetwork Packet Exchange Control Protocol (IPXCP)
 - DECnet Phase IV Control Protocol (DNCP)
 - NetBIOS Frames Control Protocol (NBFCP)
 - IPv6 Control Protocol (IPV6CP)

NETWORK LAYER

Data from Networking Protocols

DATA LINK LAYER

NCP	AP	LCP
IPCP	PAP	
OSINLCP	CHAP	
IPXCP		
....		

PHYSICAL LAYER

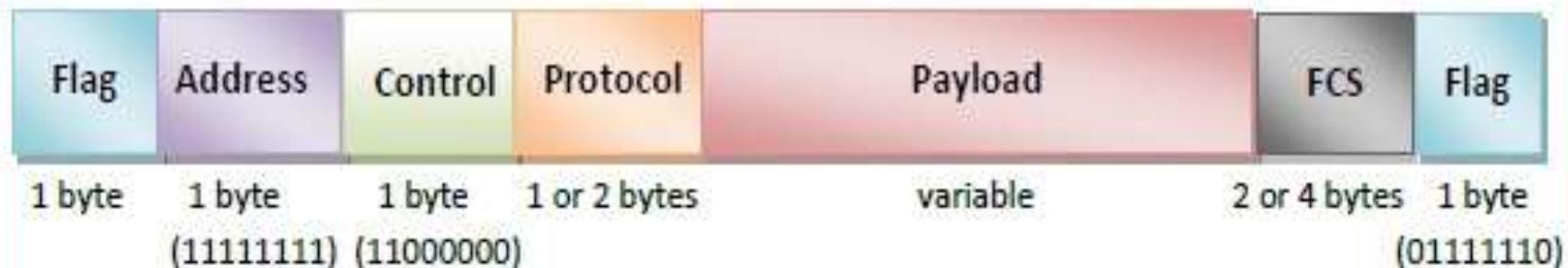
Physical Layer Protocols

PPP Frame

PPP is a byte - oriented protocol where each field of the frame is composed of one or more bytes. The fields of a PPP frame are –

- **Flag** – 1 byte that marks the beginning and the end of the frame. The bit pattern of the flag is 01111110.
- **Address** – 1 byte which is set to 11111111 in case of broadcast.
- **Control** – 1 byte set to a constant value of 11000000.
- **Protocol** – 1 or 2 bytes that define the type of data contained in the payload field.
- **Payload** – This carries the data from the network layer. The maximum length of the payload field is 1500 bytes. However, this may be negotiated between the endpoints of communication.
- **FCS** – It is a 2 byte or 4 bytes frame check sequence for error detection. The standard code used is CRC (cyclic redundancy code)

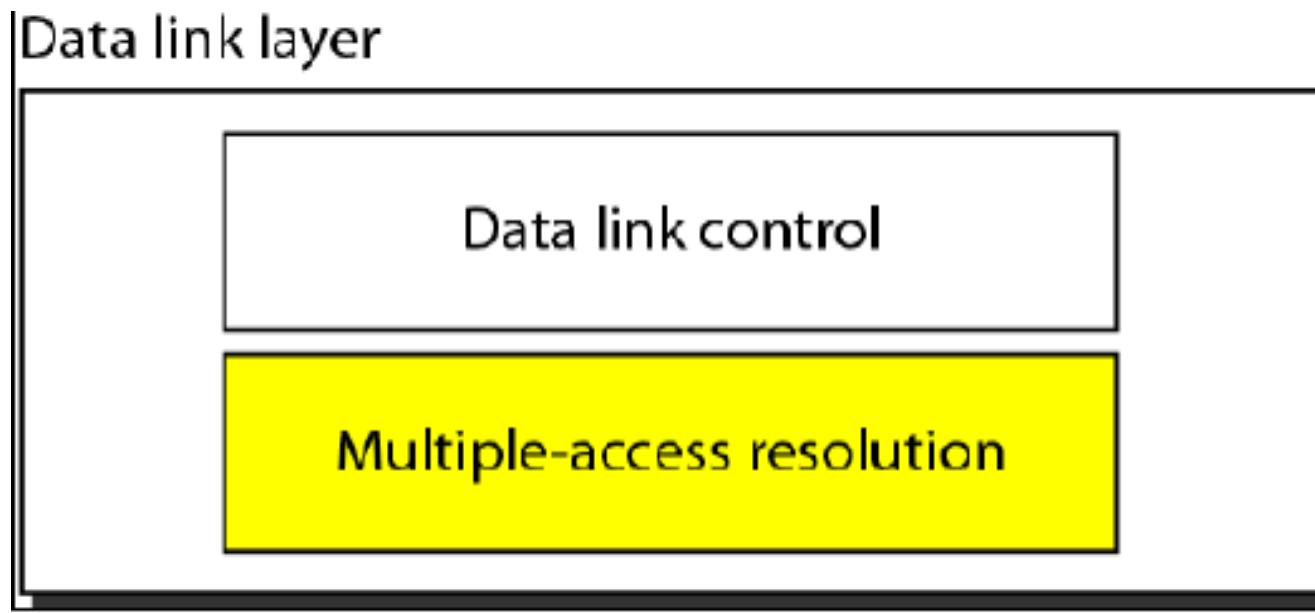
PPP Frame

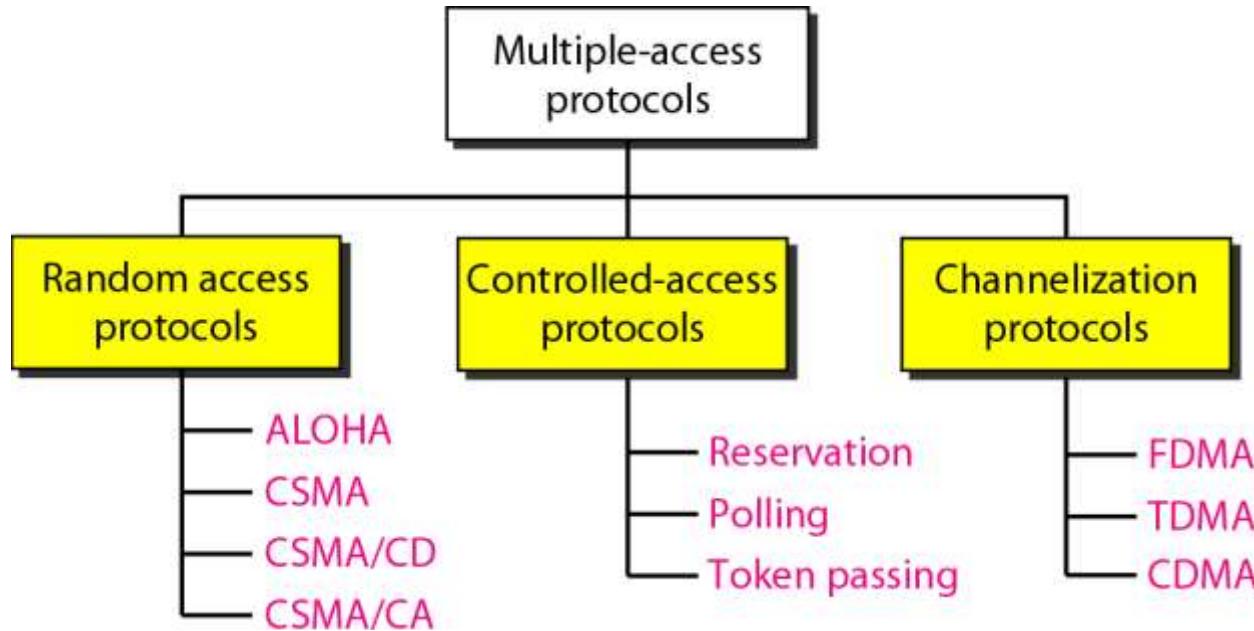


MEDIA ACCESS CONTROL

RANDOM ACCESS PROTOCOLS

We can consider the data link layer as two sub layers. The upper sub layer is responsible for data link control, and the lower sub layer is responsible for resolving access to the shared media





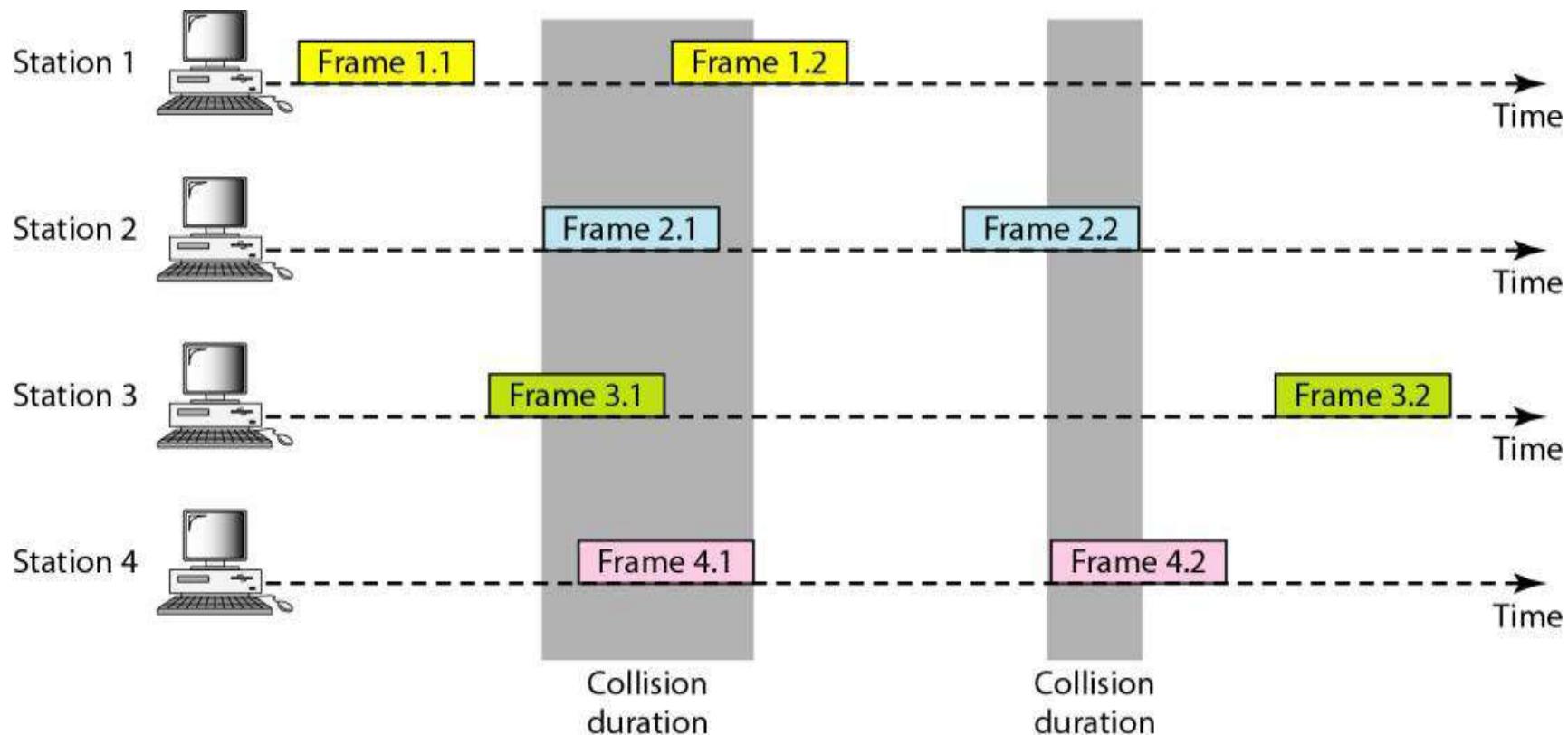
RANDOM ACCESS

In random access or contention methods, no station is superior to another station and none is assigned the control over another. Two features give this method its name. First, there is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called *random access*. Second, no rules specify which station should send next.

ALOHA

1 Pure ALOHA

The original ALOHA protocol is called pure ALOHA. This is a simple, but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send. However, since there is only one channel to share, there is the possibility of collision between frames from different stations.



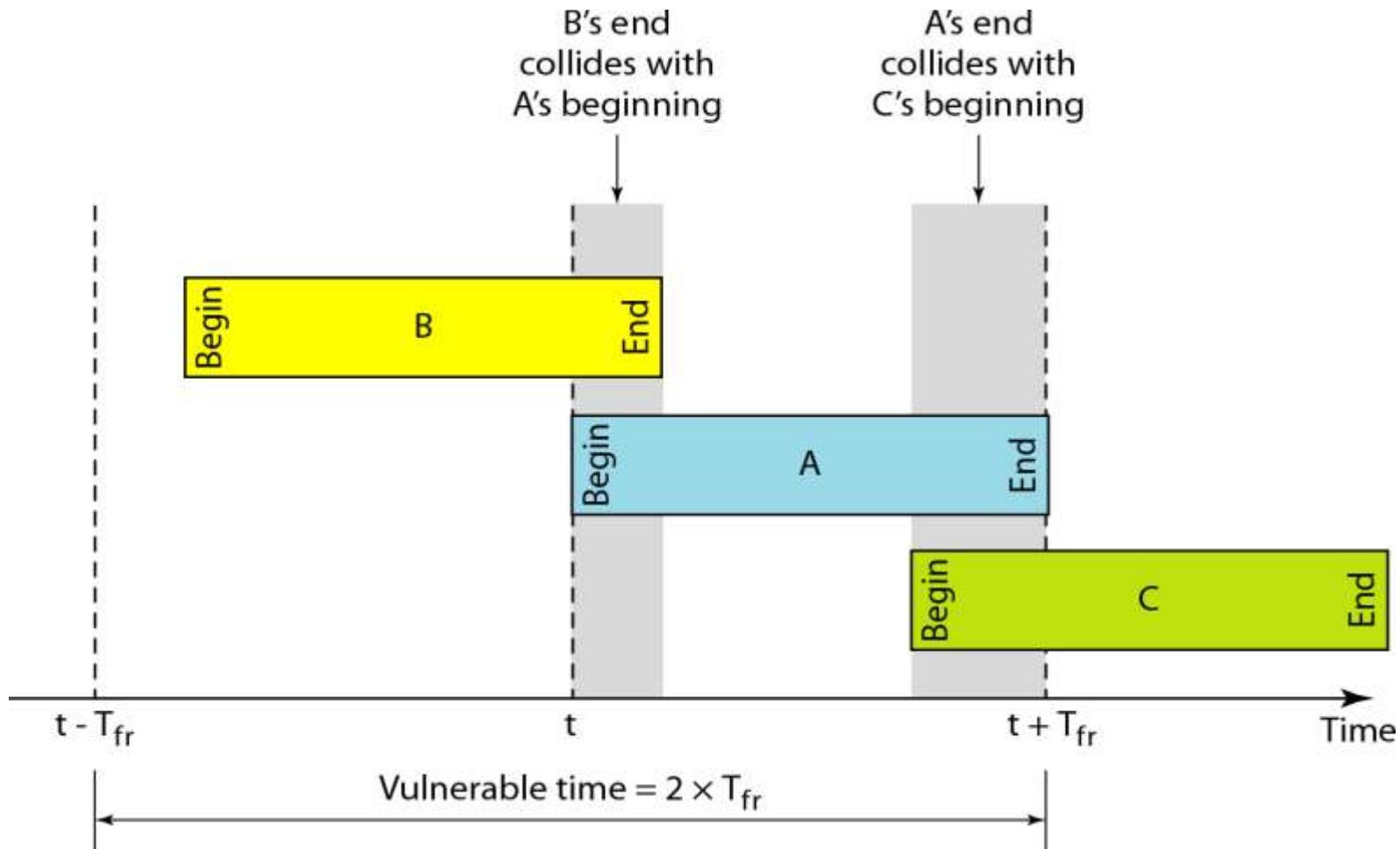
In pure ALOHA,

- the stations transmit frames whenever they have data to send.
- When two or more stations transmit simultaneously, there is collision and the frames are destroyed.
- In pure ALOHA, whenever any station transmits a frame, it expects the acknowledgement from the receiver.
- If acknowledgement is not received within specified time, the station assumes that the frame (or acknowledgement) has been destroyed.
- If the frame is destroyed because of collision the station waits for a random amount of time and sends it again. This waiting time must be random otherwise same frames will collide again and again.

Vulnerable time

Let us find the length of time, the **vulnerable time**, in which there is a possibility of collision. We assume that the stations send fixed length frames with each frame taking T_{fr}

S to send. Below Figure shows the vulnerable time for station A



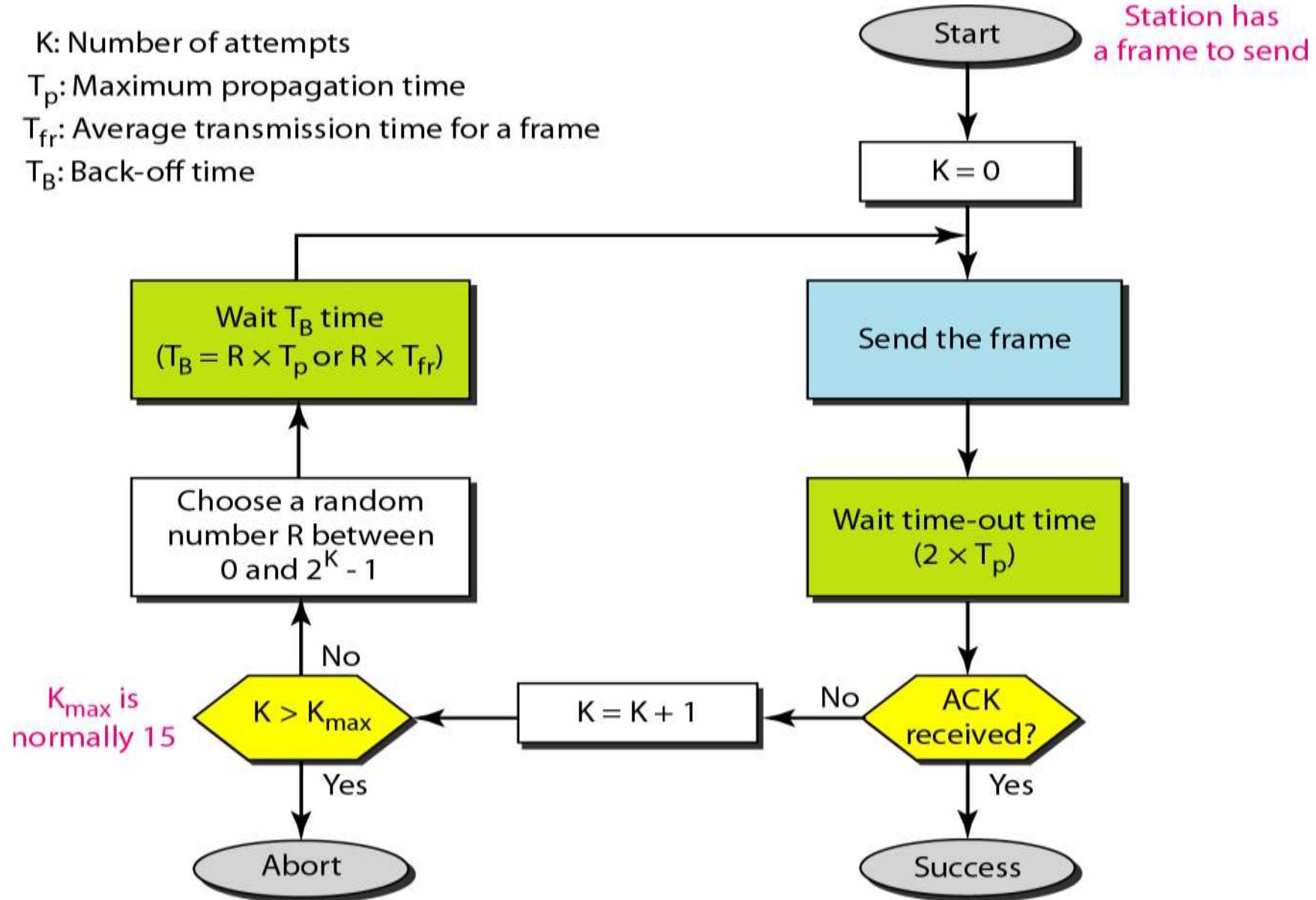
Procedure for pure ALOHA protocol

K: Number of attempts

T_p : Maximum propagation time

T_{fr} : Average transmission time for a frame

T_B : Back-off time

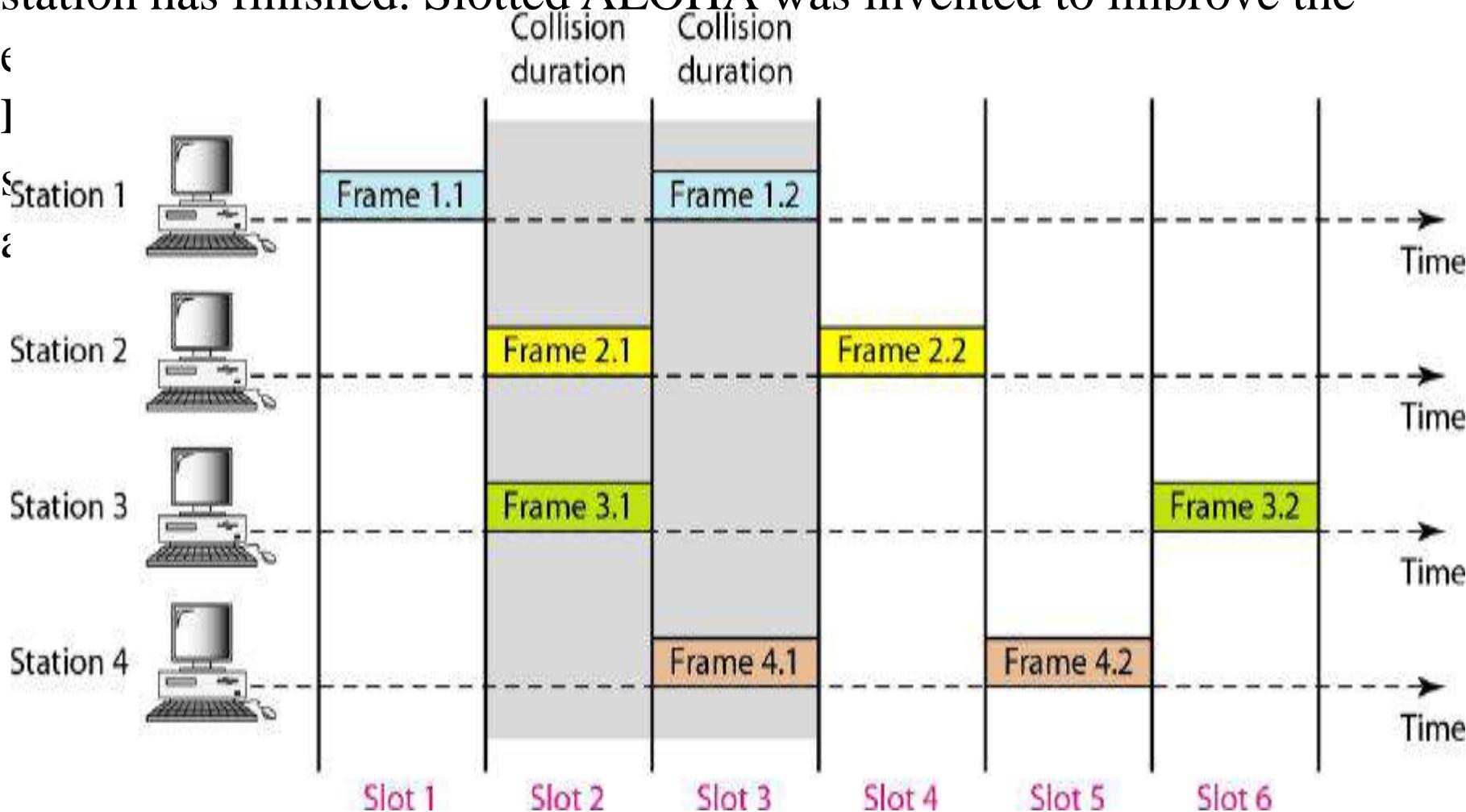


Station has
a frame to send

K_{max} is
normally 15

2 Slotted ALOHA

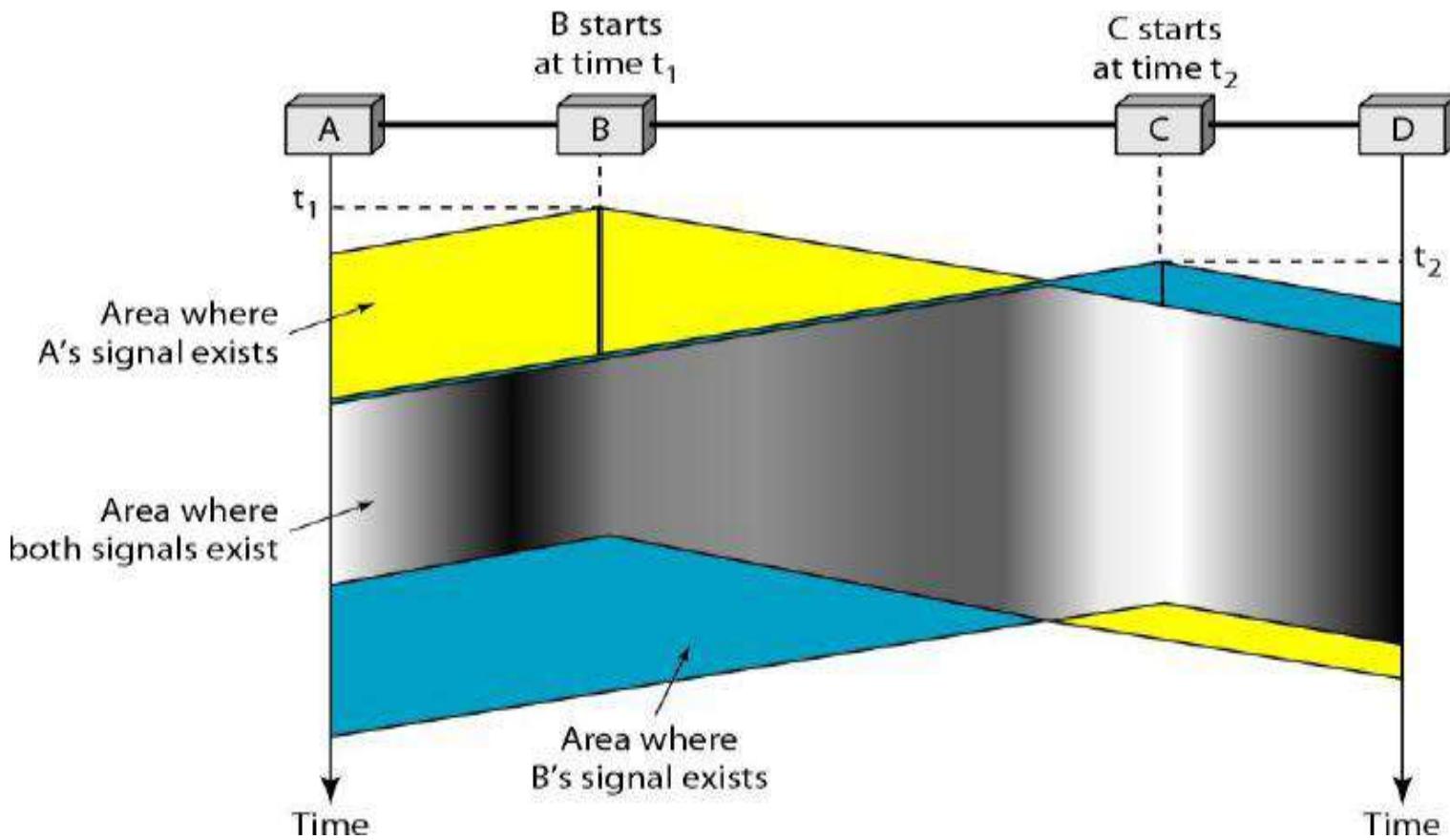
Pure ALOHA has a vulnerable time of $2 \times T_{fr}$. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or soon before another station has finished. Slotted ALOHA was invented to improve the



Carrier Sense Multiple Access (CSMA)

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending

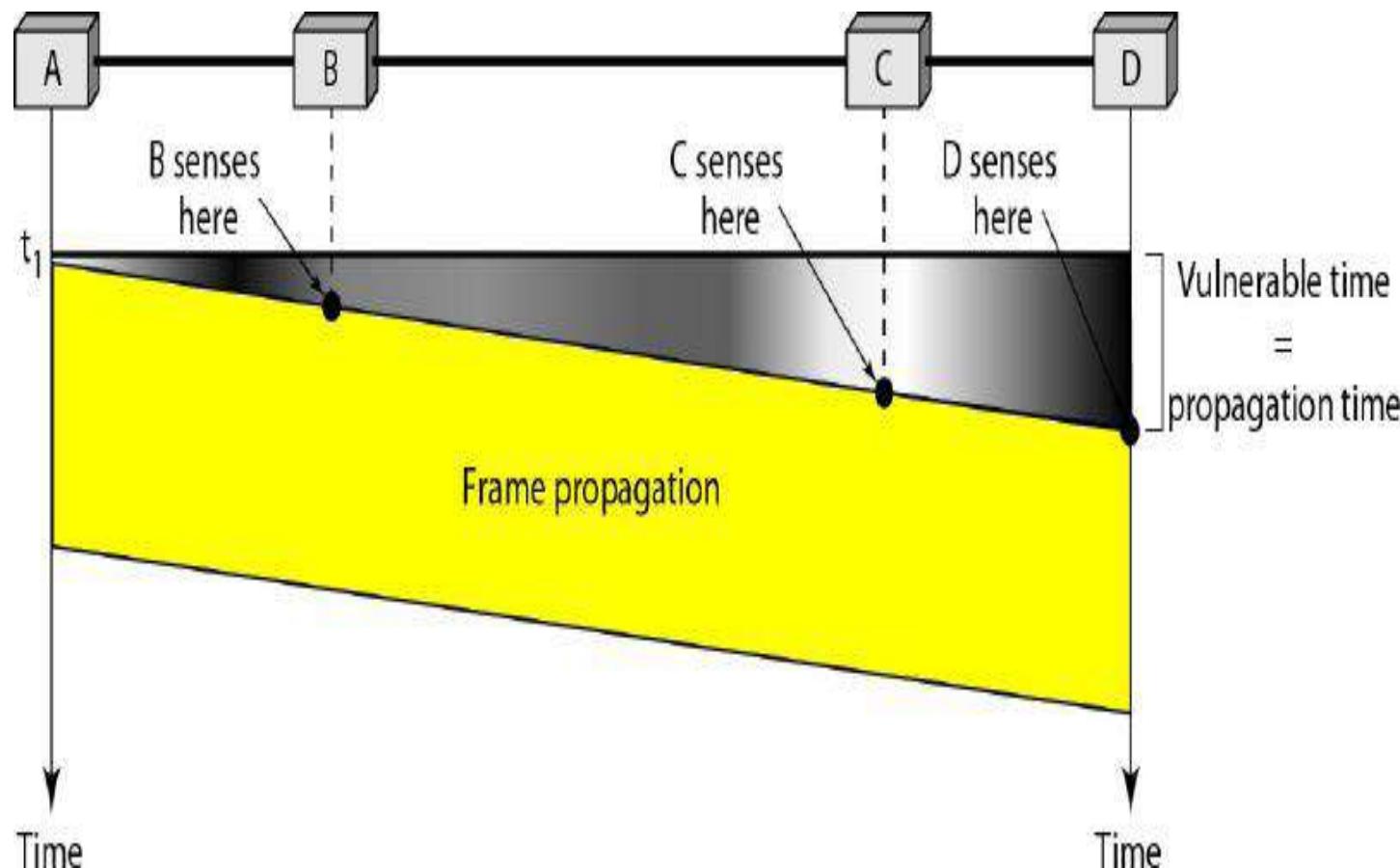
CSMA can reduce the possibility of collision, but it cannot eliminate it



Space/time model of the collision in CSMA

Vulnerable Time

The vulnerable time for CSMA is the propagation time T_p . This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame, and any other station tries to send a frame during this time, a collision will result.



Persistence Methods

What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: the 1-persistent method, the non-persistent method, and the p-persistent method

1-Persistent:

In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.

Non-persistent:

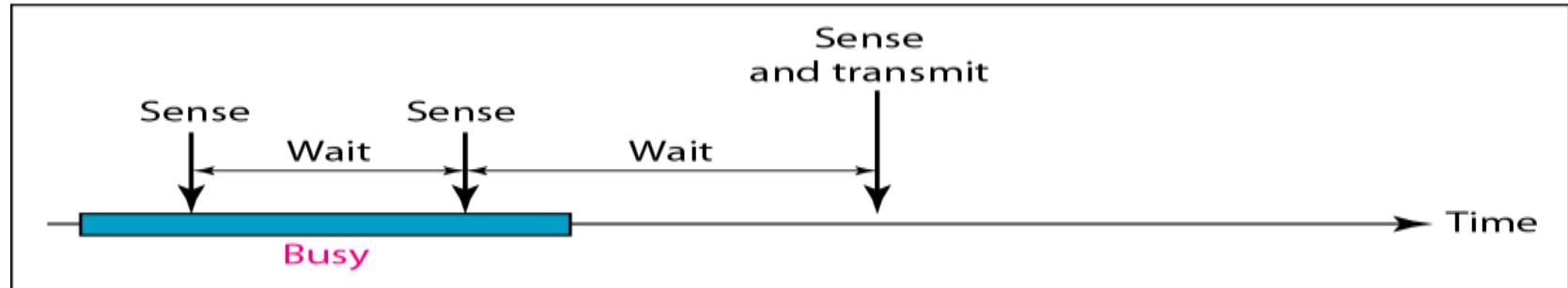
a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. This approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously.

p-Persistent:

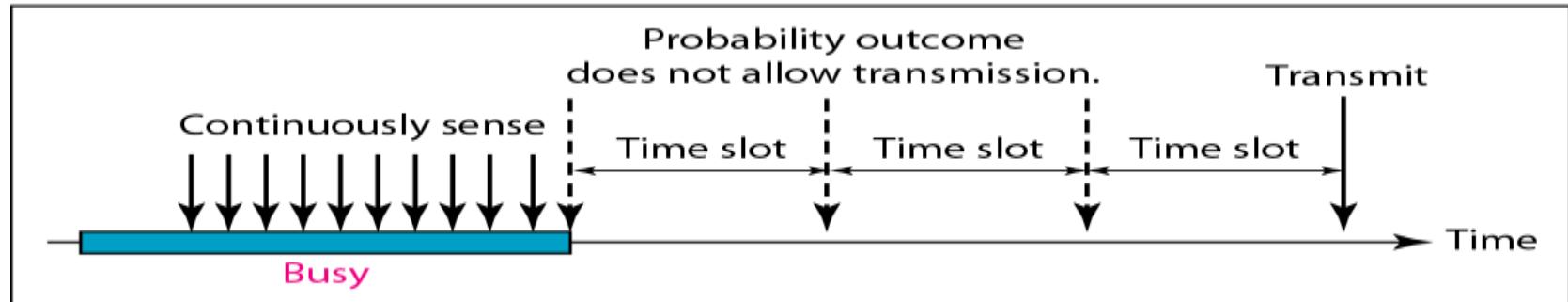
This is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The p-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency.



a. 1-persistent



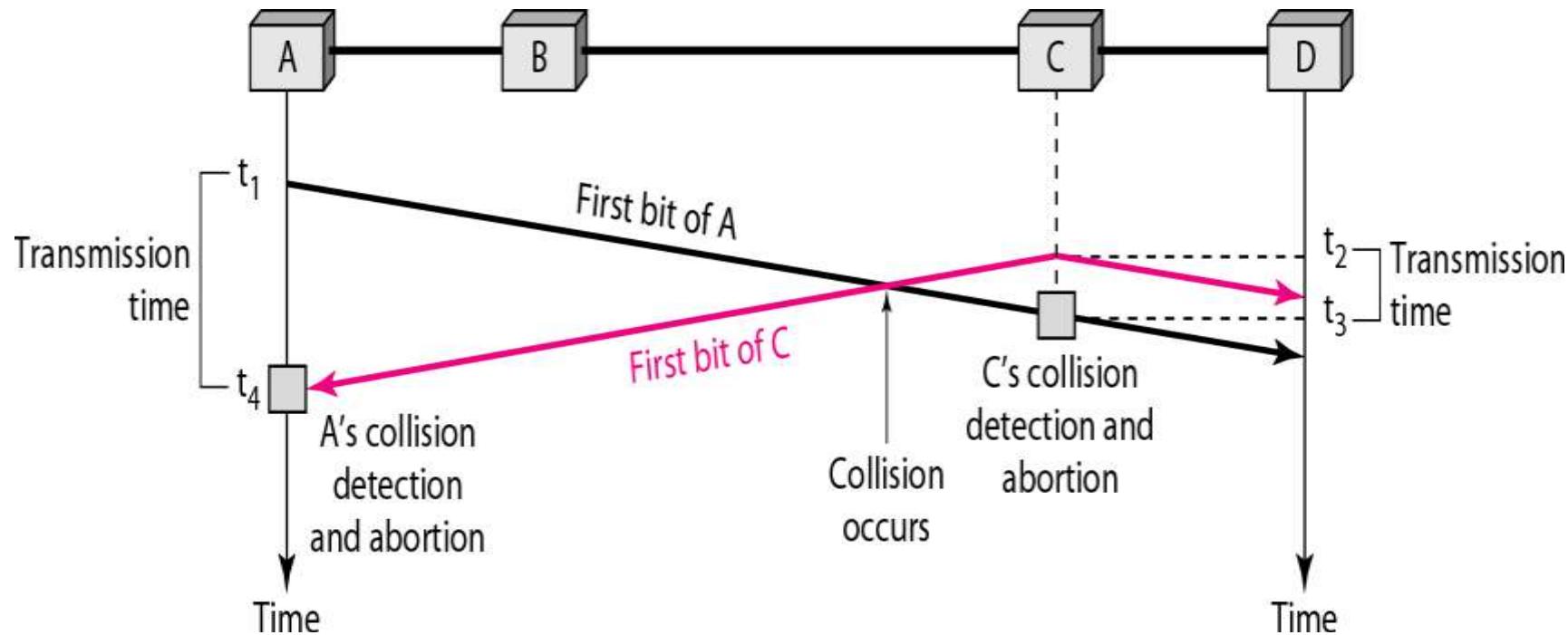
b. Nonpersistent



c. p-persistent

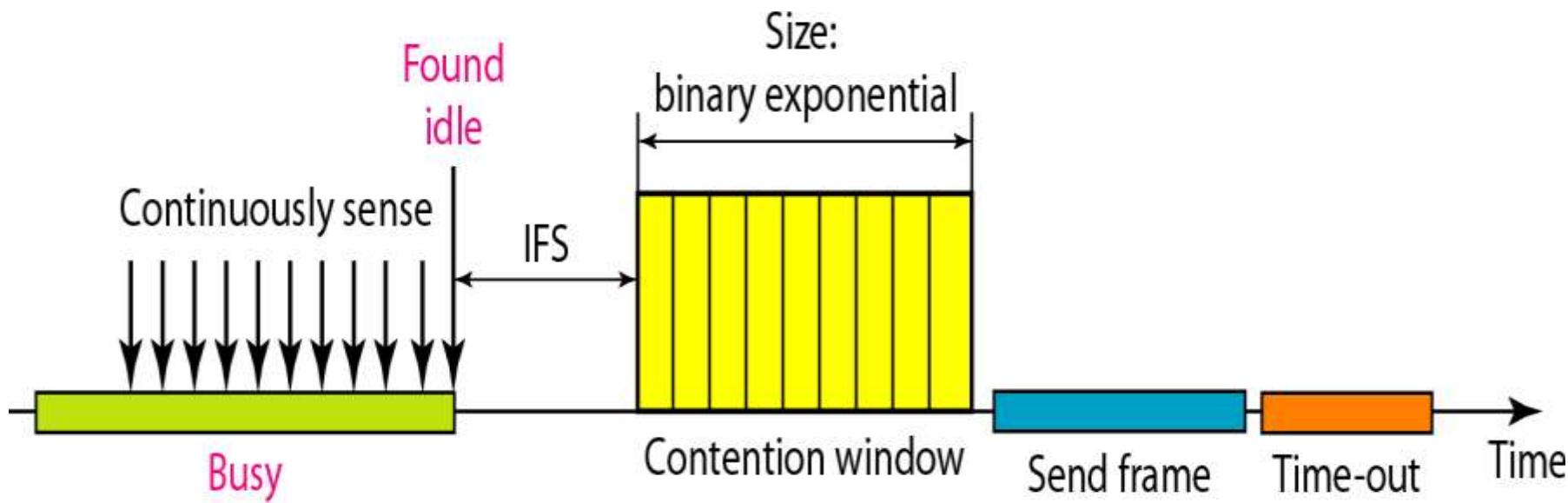
Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

The CSMA method does not specify the procedure following a collision. Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision. In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.



Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance (*CSMA/CA*) was invented for wirelessss network. Collisions are avoided through the use of CSMA/CA's three strategies: the inter frame space, the contention window, and acknowledgments, as shown in Figure



Controlled Access Protocols

In controlled access, the stations seek information from one another to find which station has the right to send. It allows only one node to send at a time, to avoid collision of messages on shared medium. The three controlled-access methods are:

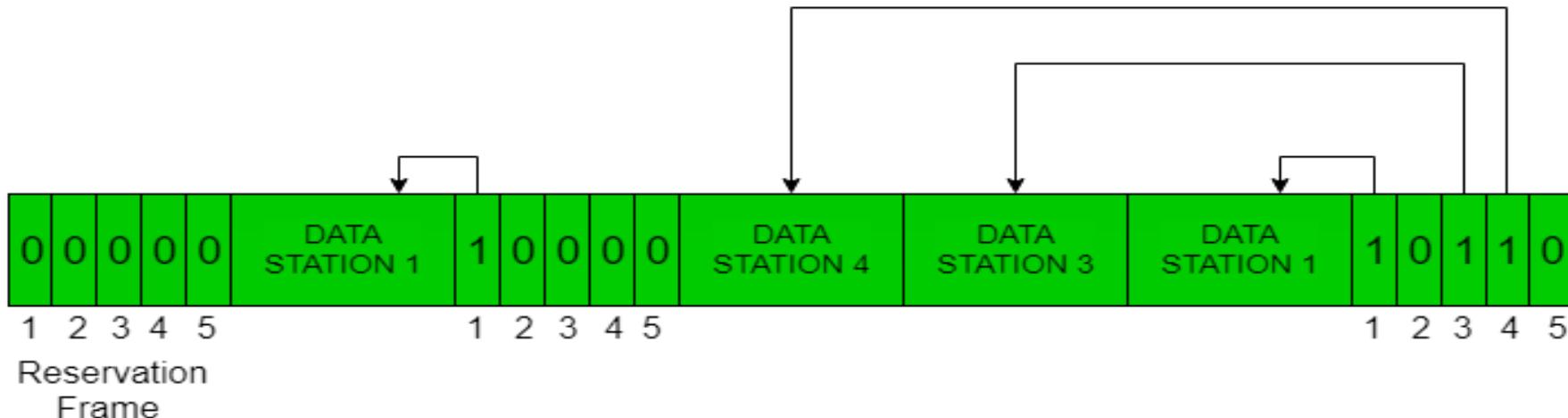
- Reservation 2. Polling 3. Token Passing

Reservation

In the reservation method, a station needs to make a reservation before sending data.

The time line has two kinds of periods:

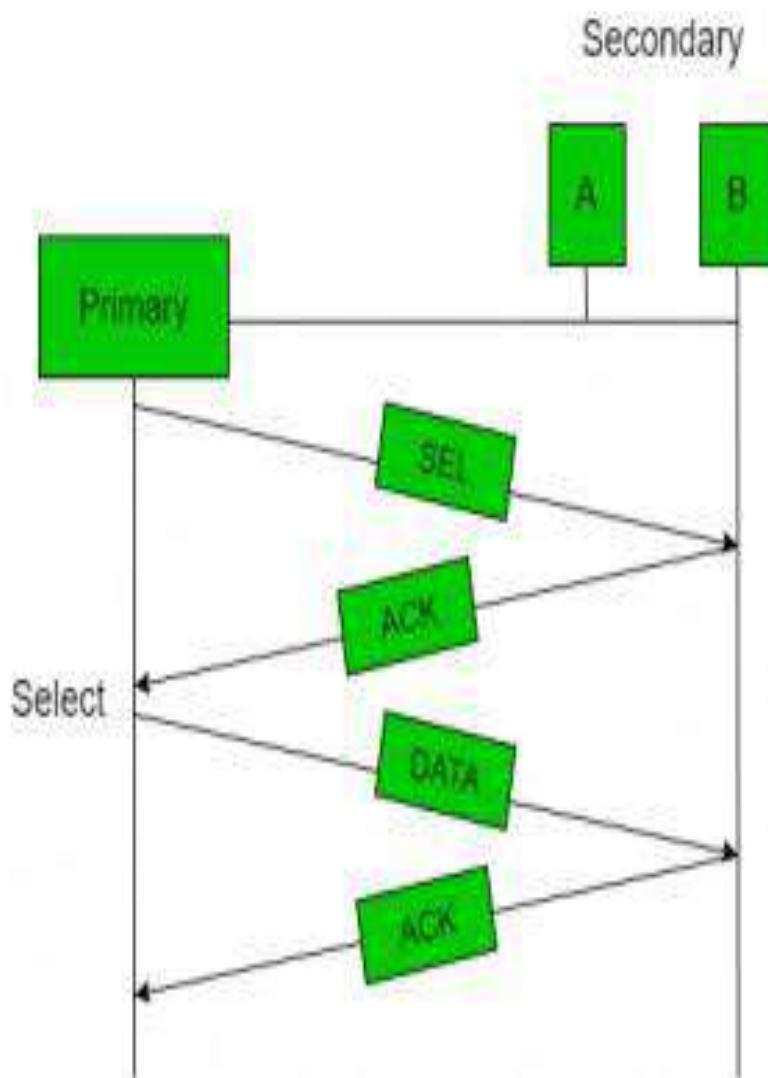
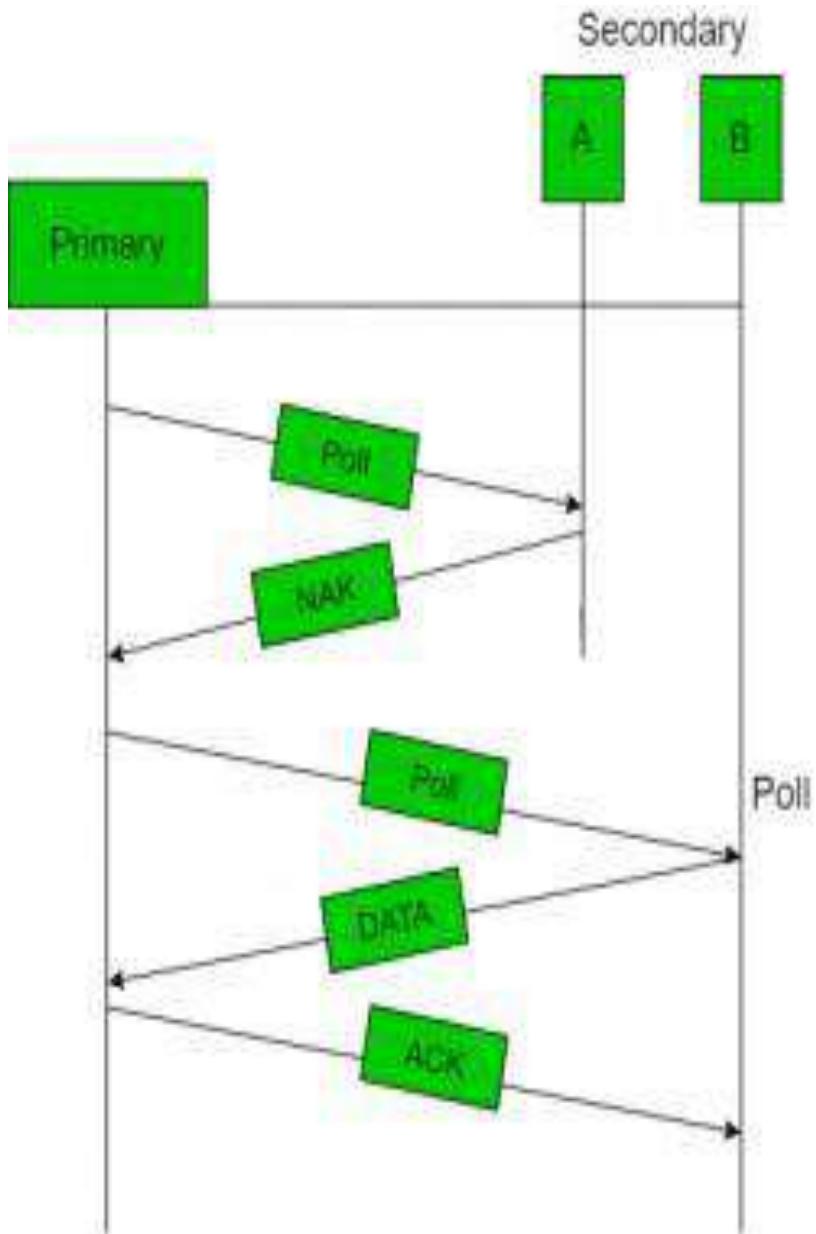
1. Reservation interval of fixed time length
2. Data transmission period of variable frames.



Polling

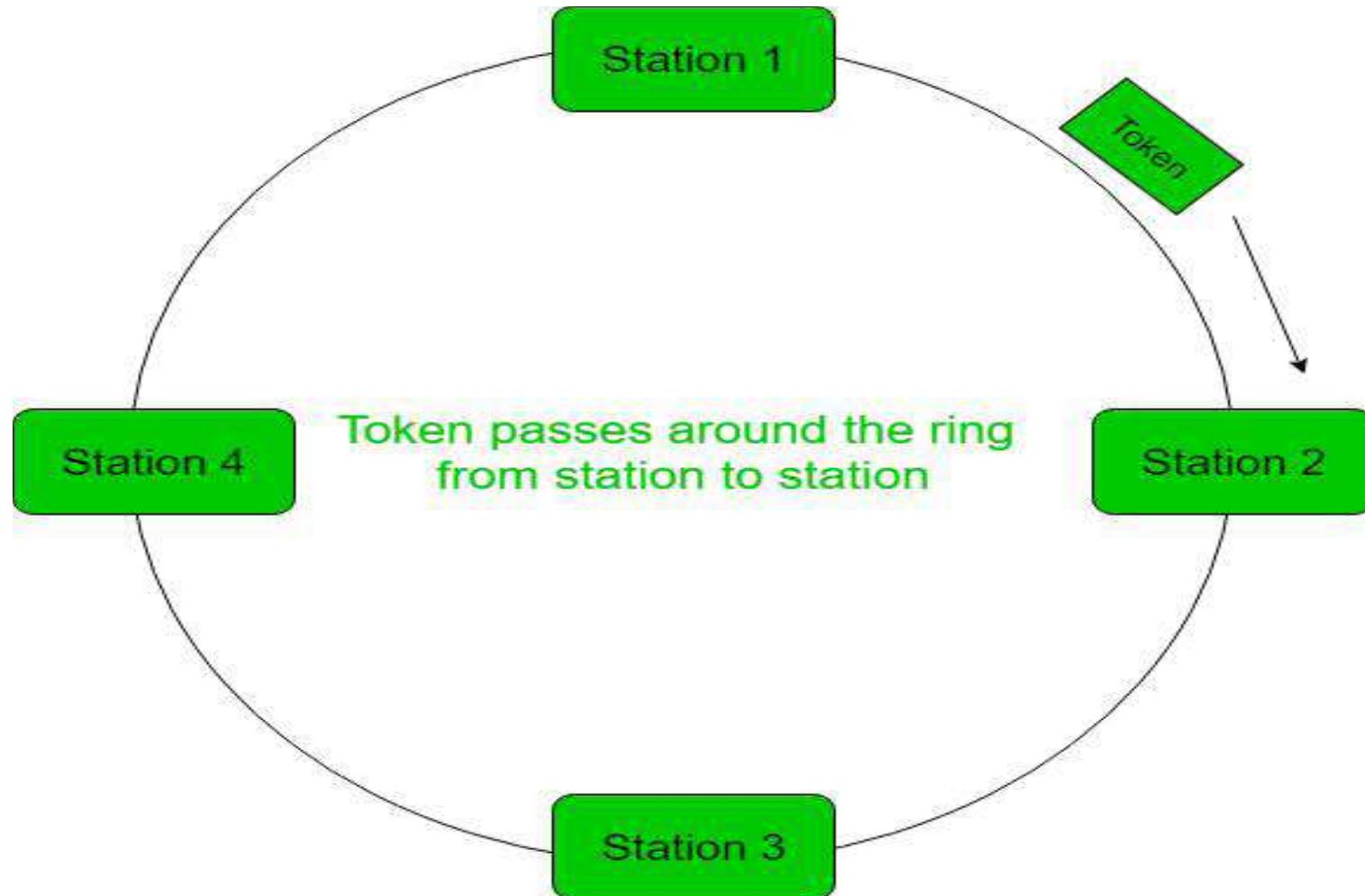
- Polling process is similar to the roll-call performed in class. Just like the teacher, a controller sends a message to each node in turn.
- In this, one acts as a primary station(controller) and the others are secondary stations. All data exchanges must be made through the controller.
- The message sent by the controller contains the address of the node being selected for granting access.
- Although all nodes receive the message but the addressed one responds to it and sends data, if any.

If there is no data, usually a “poll reject”(NAK) Problems include high overhead of the polling messages and high dependence on the reliability of the controller



Token Passing

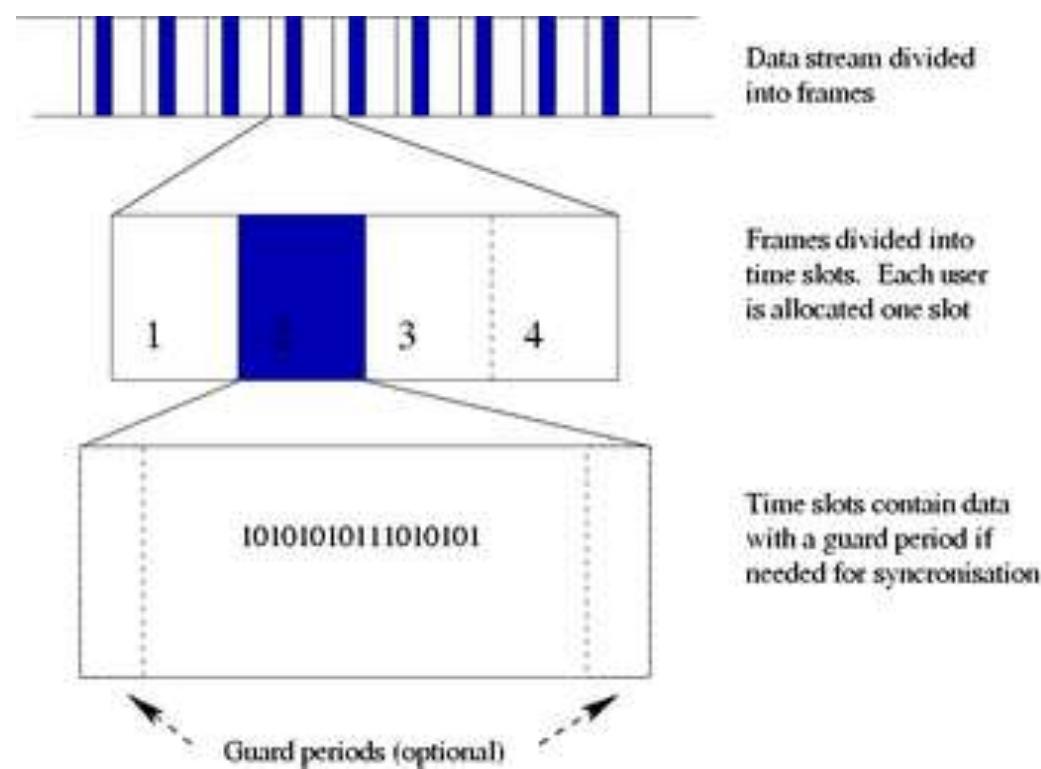
- In token passing scheme, the stations are connected logically to each other in form of ring and access of stations is governed by tokens.
- A token is a special bit pattern or a small message, which circulate from one station to the next in some predefined order.
- In Token ring, token is passed from one station to another adjacent station in the ring whereas in case of Token bus, each station uses the bus to send the token to the next station in some predefined order.
- In both cases, token represents permission to send. If a station has a frame queued for transmission when it receives the token, it can send that frame before it passes the token to the next station. If it has no queued frame, it passes the token simply.
- After sending a frame, each station must wait for all N stations (including itself) to send the token to their neighbors and the other $N - 1$ stations to send a frame, if they have one.



CHANNELIZATION

In this, the available bandwidth of the link is shared in time, frequency and code to multiple stations to access channel simultaneously.

Time Division Multiple Access (TDMA) : a digital wireless telephony transmission technique. TDMA allocates each user a different time slot on a given frequency. TDMA divides each cellular channel into three time slots in order to increase the amount of data that can be carried.

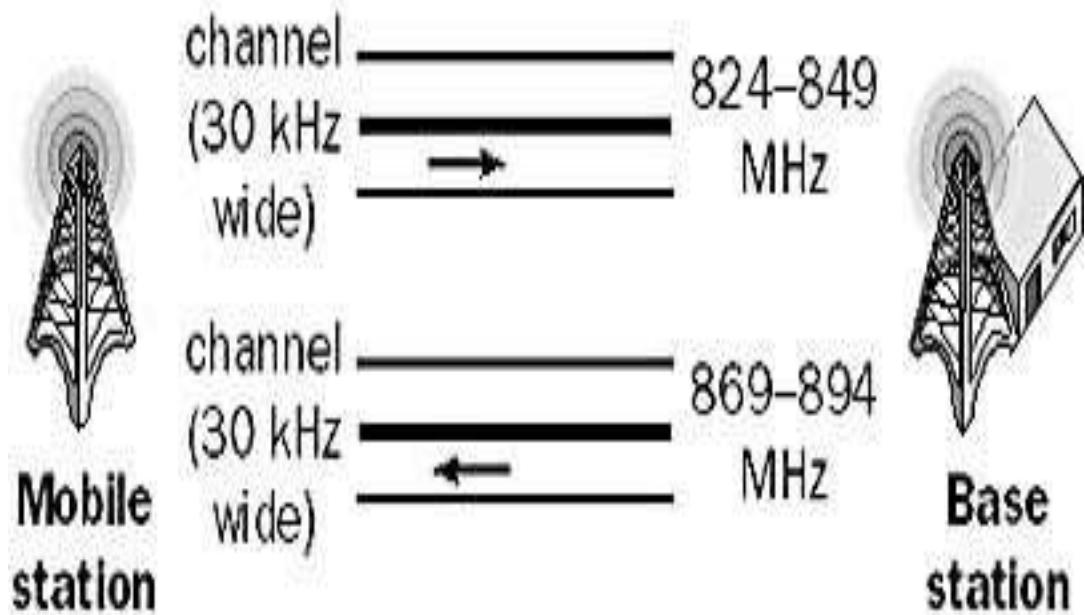


CDMA

Code Division Multiple Access (CDMA): a digital wireless technology that uses spread-spectrum techniques. CDMA does not assign a specific frequency to each user. Instead, every channel uses the full available spectrum. Individual conversations are encoded with a pseudo-random digital sequence.

FDMA

Frequency Division Multiple Access (FDMA) is one of the most common analogue multiple access methods. The frequency band is divided into channels of equal bandwidth so that each conversation is carried on a different frequency



UNIT- III

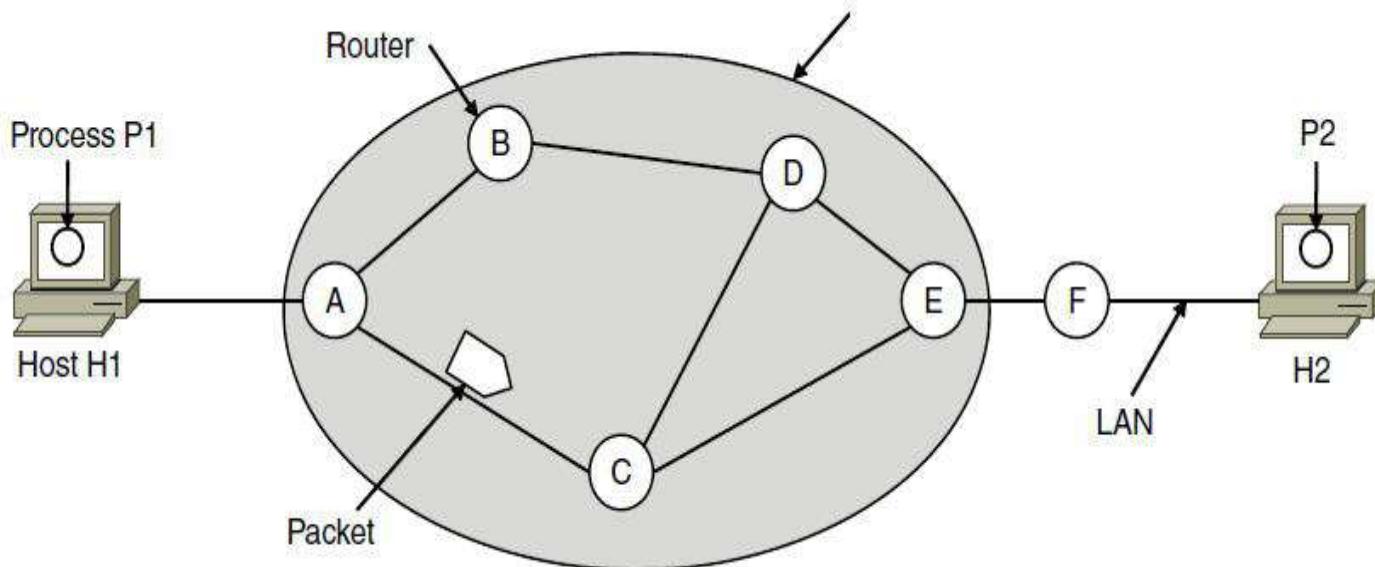
The Network Layer: Network layer design issues- Routing algorithms - Congestion control algorithms - Quality of service- Internetworking- The network layer in the Internet: IPV4 -IPV6, Internet Control protocols- OSPF- BGP.

Unit-3

The Network Layer

Network layer design issues

1. Store-and-forward packet switching
2. Services provided to transport layer
3. Implementation of connectionless service
4. Implementation of connection-oriented service
5. Comparison of virtual-circuit and datagram networks



1 Store-and-forward packet switching

A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP.

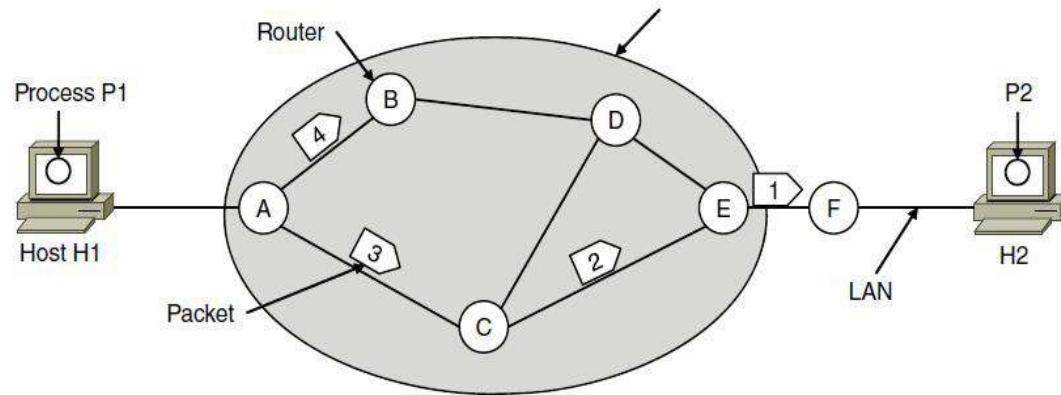
The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching

2. Services provided to transport layer The network layer provides services to the transport layer at the network layer/transport layer interface. The services need to be carefully designed with the following goals in mind:

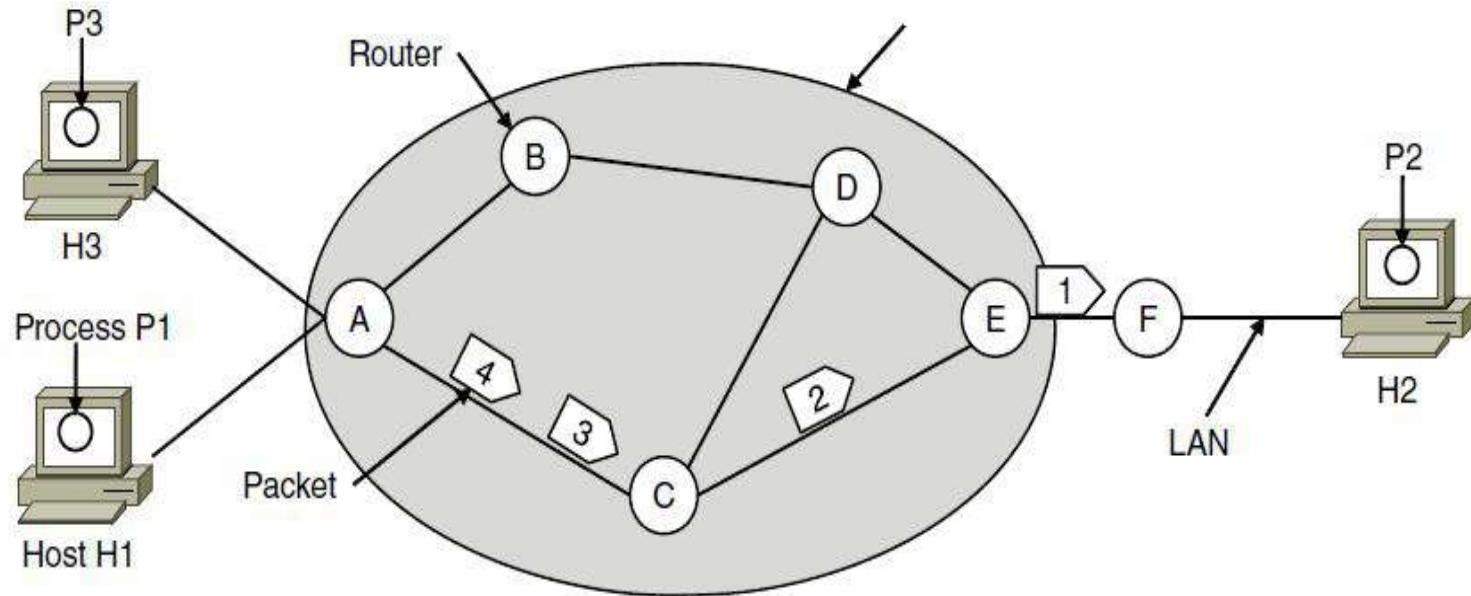
1. Services independent of router technology.
2. Transport layer shielded from number, type, topology of routers.
3. Network addresses available to transport layer use uniform numbering plan even across LANs and WANs

3 .Implementation of connectionless service

If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** (in analogy with telegrams) and the network is called a **datagram network**



4 Implementation of connection oriented service:



A's table

H1	1
H3	1
C	1
C	2

In Out

C's Table

A	1
A	2
E	1
E	2

E's Table

C	1
C	2
F	1
F	2

5 Comparison of virtual-circuit and datagram networks

Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

ROUTING ALGORITHMS

The main function of NL (Network Layer) is routing packets from the source machine to the destination machine. There are two processes inside router:

- a) One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing table. This process is forwarding.
- b) The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play. This process is routing.

Routing algorithms can be grouped into two major classes:

- 1) nonadaptive (Static Routing)
- 2) adaptive. (Dynamic Routing)

Nonadaptive algorithm

do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J is computed in advance, off line, and downloaded to the routers when the network is booted. This procedure is sometimes called static routing.

Adaptive algorithm, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in

- 1) Where they get their information (e.g., locally, from adjacent routers, or from all routers),
- 2) When they change the routes (e.g., every ΔT sec, when the load changes or when the topology changes), and
- 3) What metric is used for optimization (e.g., distance, number of hops, or estimated transit time). This procedure is called dynamic routing.

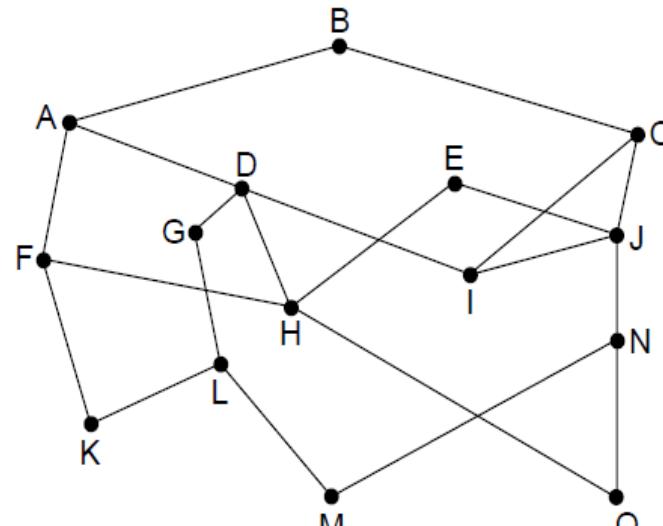
Different Routing Algorithms

- Optimality principle
- Shortest path algorithm
- Flooding
- Distance vector routing
- Link state routing
- Hierarchical Routing

The Optimality Principle

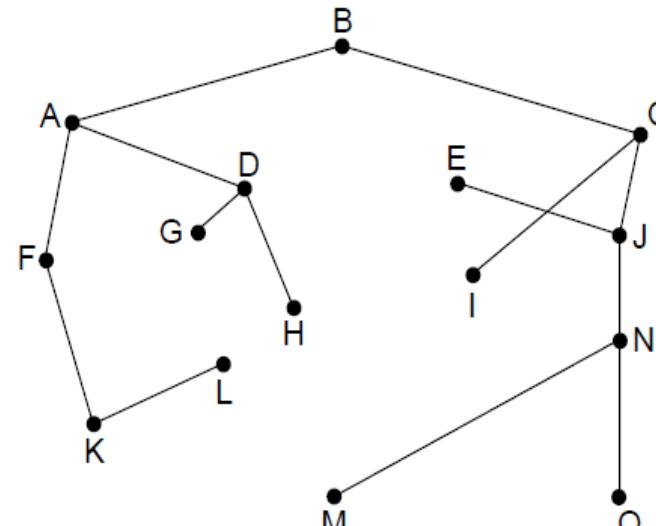
It states that if router J is on the optimal path from router I to router K, then the optimal path

from J to K also falls along the same. As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a **sink tree**. The goal of all routing algorithms is to discover and use the sink trees for all routers.



(a)

(a) A network.



(b)

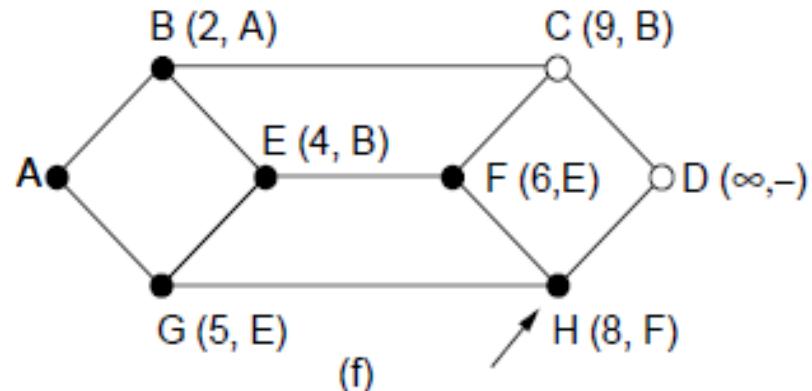
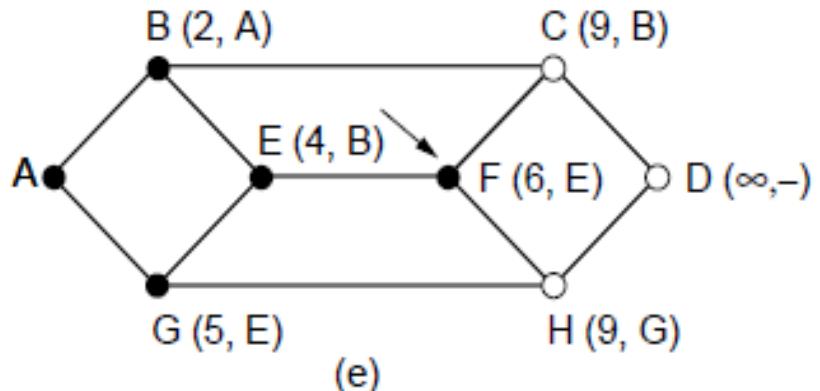
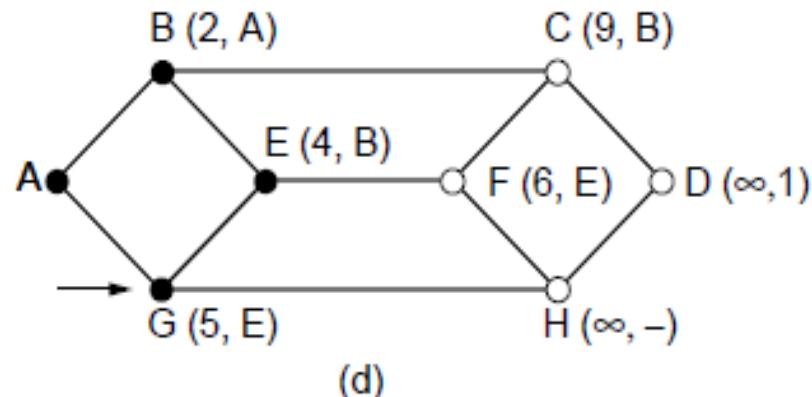
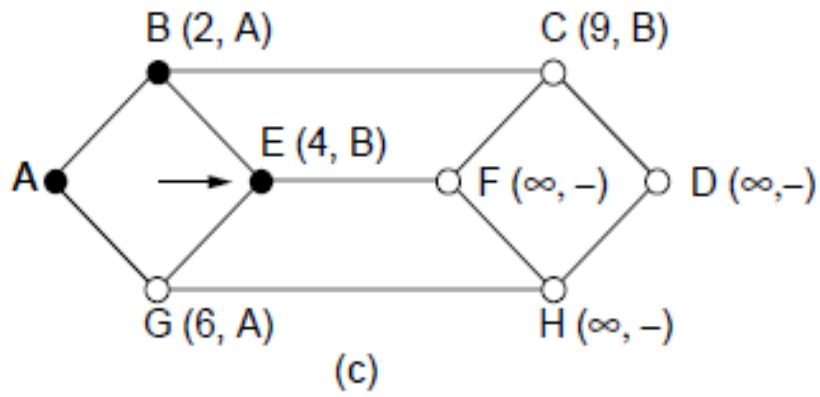
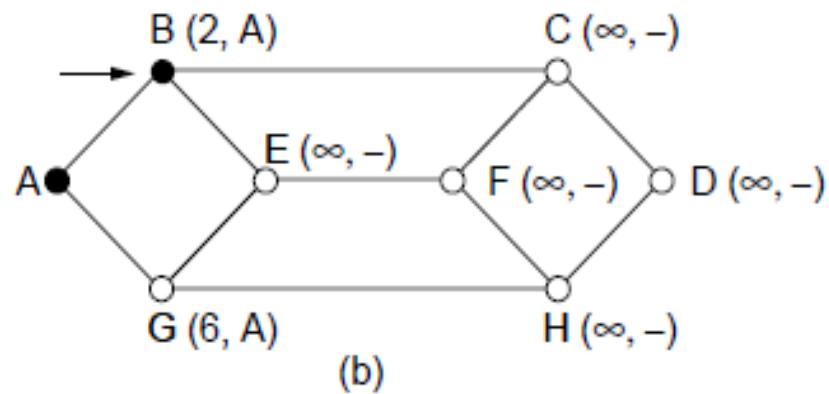
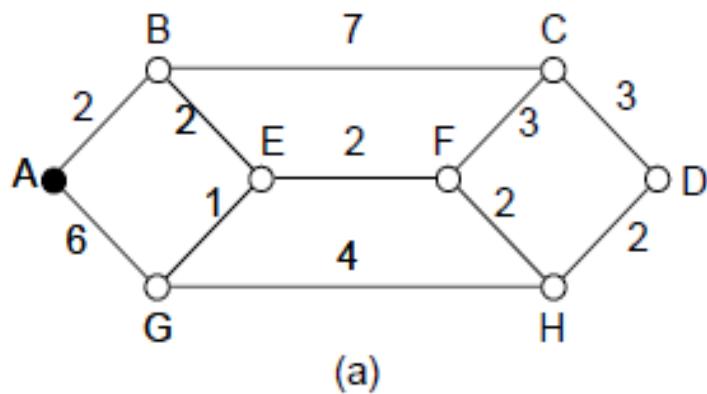
(b) A sink tree for router B.

Shortest Path Routing (Dijkstra's)

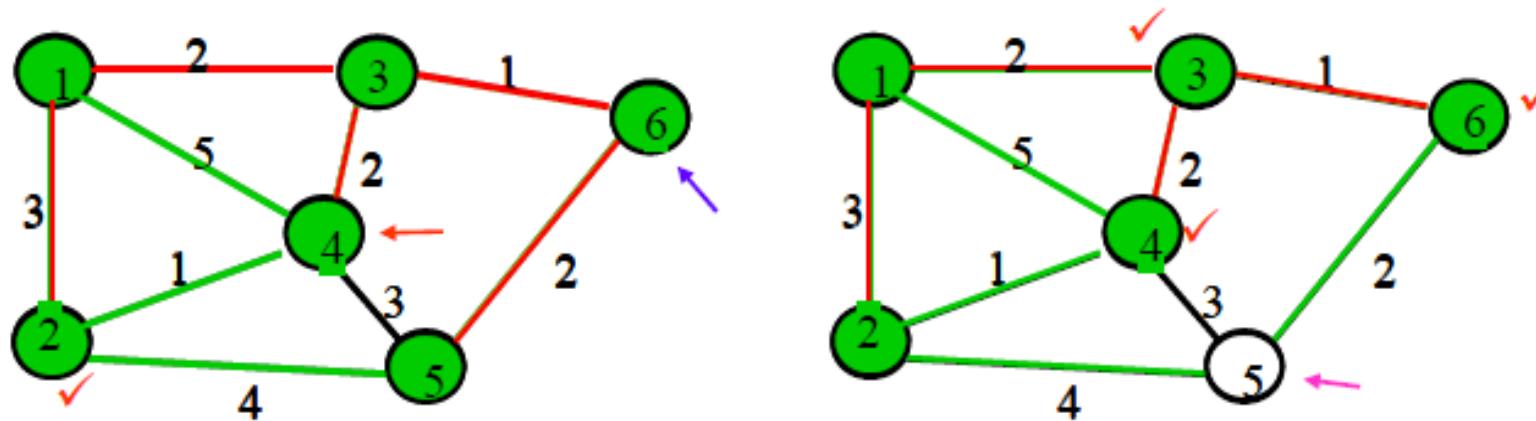
The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line or link.

To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph

1. Start with the local node (router) as the root of the tree. Assign a cost of 0 to this node and make it the first permanent node.
2. Examine each neighbor of the node that was the last permanent node.
3. Assign a cumulative cost to each node and make it tentative
4. Among the list of tentative nodes
 - a. Find the node with the smallest cost and make it Permanent
 - b. If a node can be reached from more than one route then select the route with the shortest cumulative cost.
5. Repeat steps 2 to 4 until every node becomes permanent



Execution of Dijkstra's algorithm



Iteration	Permanent	tentative	D_2	D_3	D_4	D_5	D_6
Initial	{1}	{2,3,4}	3	2 ✓	5	∞	∞
1	{1,3}	{2,4,6}	3 ✓	2	4	∞	3
2	{1,2,3}	{4,6,5}	3	2	4	7	3 ✓
3	{1,2,3,6}	{4,5}	3	2	4 ✓	5	3
4	{1,2,3,4,6}	{5}	3	2	4	5 ✓	3
5	{1,2,3,4,5,6}	{}	3	2	4	5	3

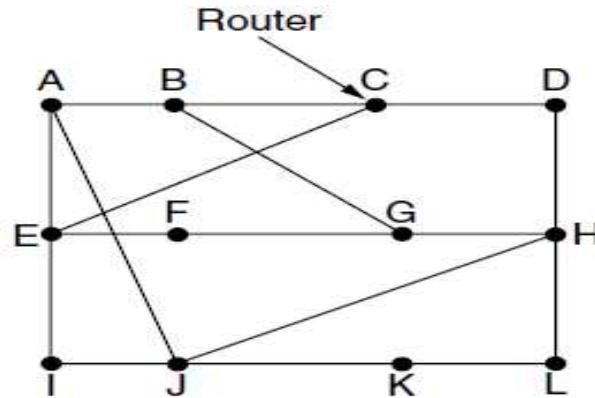
Flooding

- Another static algorithm is flooding, in which every incoming packet is sent out on every outgoing line except the one it arrived on.
- Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process.
- One such measure is to have a hop counter contained in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero.

Ideally, the hop counter should be initialized to the length of the path from source to destination.

Distance Vector Routing

A **distance vector routing** algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination.



(a)

Vector received from J's four neighbors

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K

JA delay is 8 JI delay is 10 JH delay is 12 JK delay is 6

New routing table for J

(b)

The Count-to-Infinity Problem

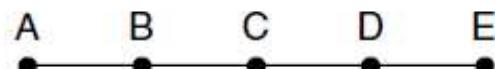
Consider a router whose best route to destination X is long. If, on the next exchange, neighbor A suddenly reports a short delay to X , the router just switches over to using the line to A to send traffic to X . In one vector exchange, the good news is processed.



Initially

1 • • • •
After 1 exchange
1 2 • • •
After 2 exchanges
1 2 3 • After 3 exchanges
1 2 3 4 After 4 exchanges

(a)



Initially

3 2 3 4 •
After 1 exchange
3 4 3 4 •
After 2 exchanges
5 4 5 4 •
After 3 exchanges
5 6 5 6 •
After 4 exchanges
7 6 7 6 •
After 5 exchanges
7 8 7 8 •
After 6 exchanges
• • • • •

(b)

Figure 5-10. The count-to-infinity problem.

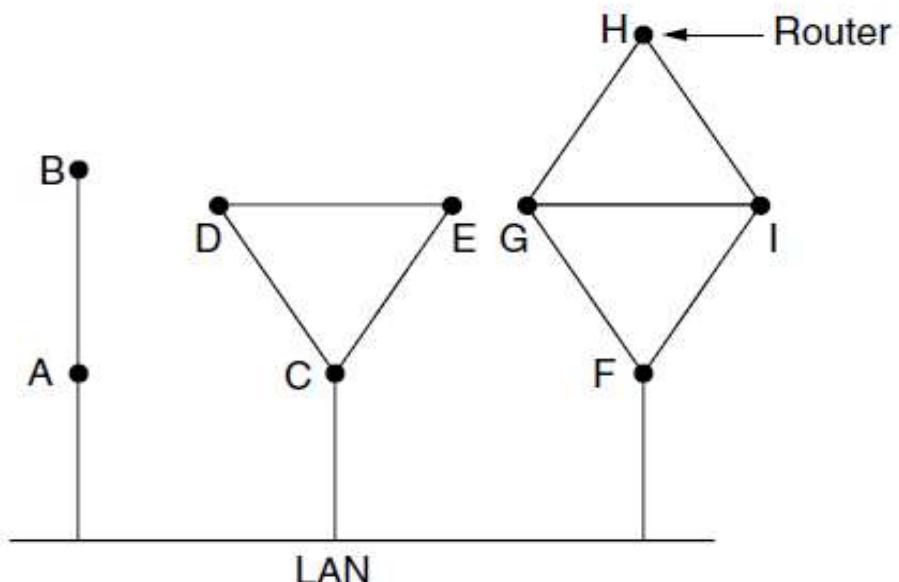
Link State Routing

Each router must do the following things to make it work:

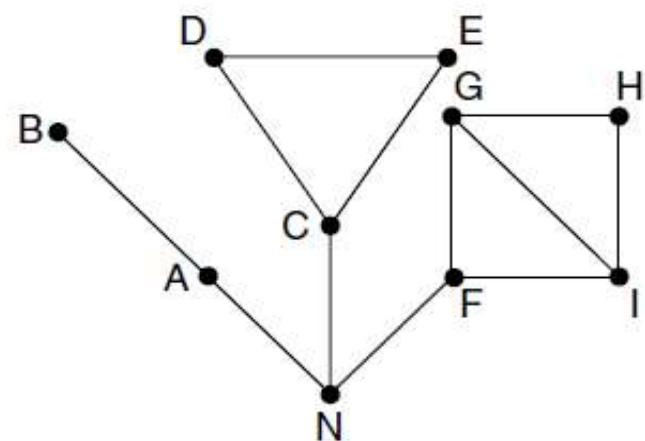
1. Discover its neighbors and learn their network addresses.
2. Set the distance or cost metric to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to and receive packets from all other routers.
5. Compute the shortest path to every other router.

Learning about the Neighbors

When a router is booted, its first task is to learn who its neighbors are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply giving its name.



(a)



(b)

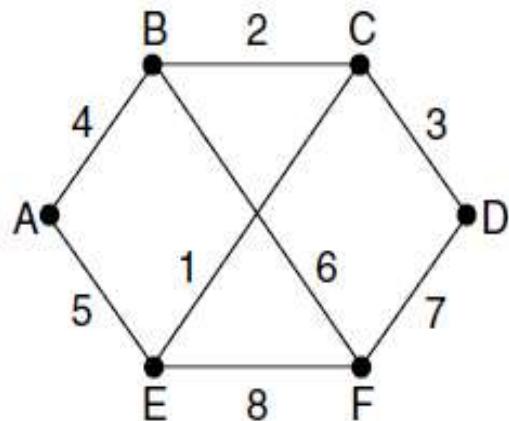
Figure 5-11. (a) Nine routers and a broadcast LAN. (b) A graph model of (a).

Setting Link Costs

The link state routing algorithm requires each link to have a distance or cost metric for finding shortest paths. The cost to reach neighbors can be set automatically, or configured by the network operator. A common choice is to make the cost inversely proportional to the bandwidth of the link. For example, 1-Gbps Ethernet may have a cost of 1 and 100-Mbps Ethernet a cost of 10. This makes higher-capacity paths better choices.

Building Link State Packets

Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and age (to be described later) and a list of neighbors. The cost to each neighbor is also given



(a)

Link	State	Packets
A	B	D
Seq.	Seq.	E
Age	Age	Seq.
B 4	B 2	Age
A 5	C 3	A 5
C 2	D 3	B 6
F 6	F 7	C 1
	E 1	D 7
		F 8
		E 8

(b)

Figure 5-12. (a) A network. (b) The link state packets for this network.

Distributing the Link State Packets

The trickiest part of the algorithm is distributing the link state packets. All of the routers must get all of the link state packets quickly and reliably. If different routers are using different versions of the topology, the routes they compute can have inconsistencies such as loops, unreachable machines, and other problems. First, we will describe the basic distribution algorithm. After that we will give some refinements. The fundamental idea is to use flooding to distribute the link state packets to all routers.

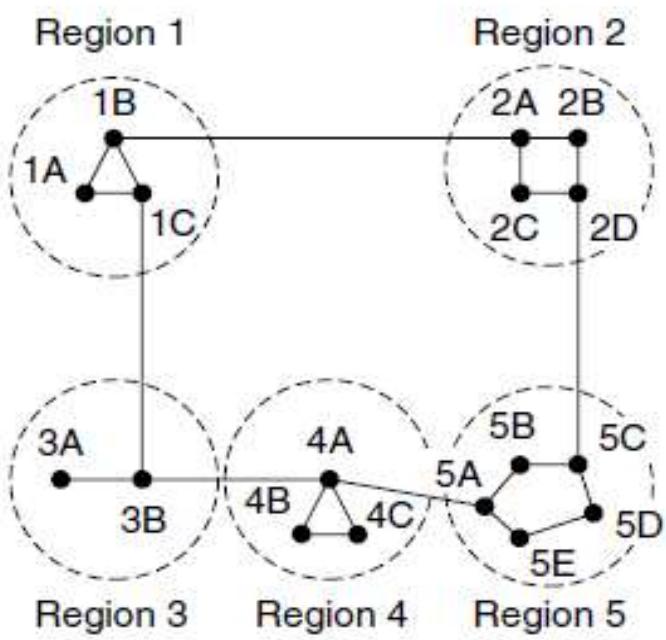
Source	Seq.	Age	Send flags			ACK flags			Data
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Figure 5-13. The packet buffer for router B in Fig. 5-12(a).

Hierarchical Routing

As networks grow in size, the router routing tables grow proportionally. Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them.

When hierarchical routing is used, the routers are divided into what we will call **regions**. Each router knows all the details about how to route packets to destinations within its own region but knows nothing about the internal structure of other regions. When different networks are interconnected, it is natural to regard each one as a separate region to free the routers in one network from having to know the topological structure of the other ones.



(a)

Full table for 1A

Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Figure 5-14. Hierarchical routing.

Broadcast Routing

Sending a packet to all destinations simultaneously is called **broadcasting**. Various methods have been proposed for doing it.

An improvement is **multidestination routing**, in which each packet contains either a list of destinations or a bit map indicating the desired destinations.

reverse path forwarding is When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the link that is normally used for sending packets *toward* the source of the broadcast.

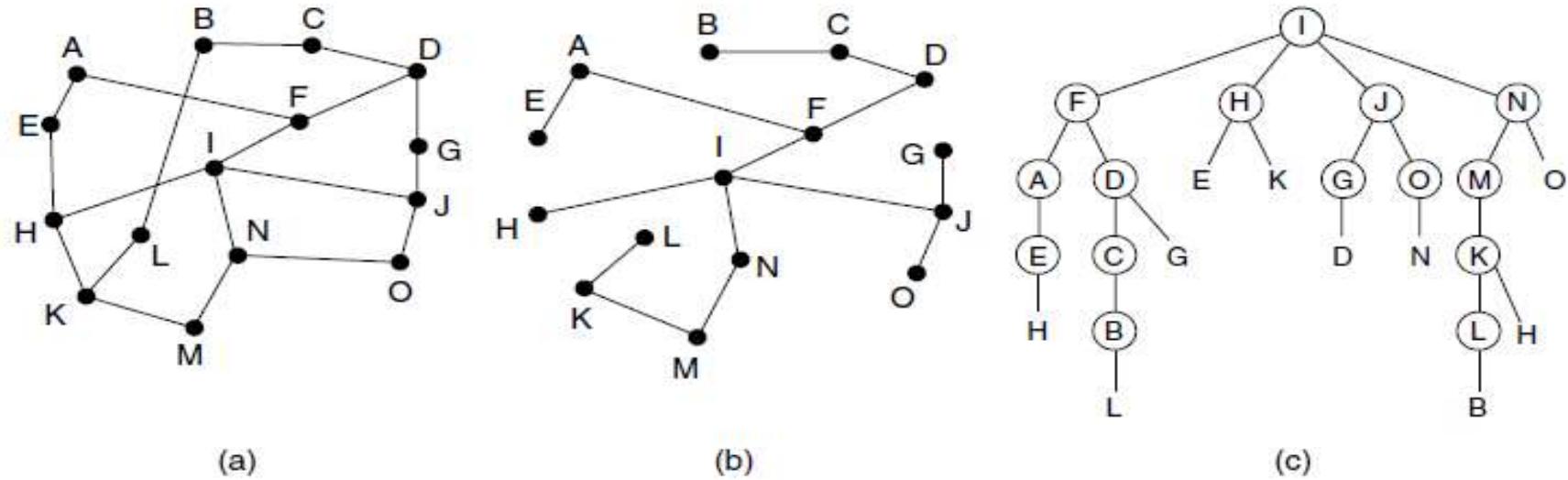


Figure 5-15. Reverse path forwarding. (a) A network. (b) A sink tree. (c) The tree built by reverse path forwarding.

A **spanning tree** is a subset of the network that includes all the routers but contains no loops. Sink trees are spanning trees. If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on. This method makes excellent use of bandwidth, generating the absolute minimum number of packets necessary to do the job.

Multicast Routing

Sending a message to a group is called **multicasting**, and the routing algorithm used is called **multicast routing**. All multicasting schemes require some way to create and destroy groups and to identify which routers are members of a group. How these tasks are accomplished is not of concern to the routing algorithm.

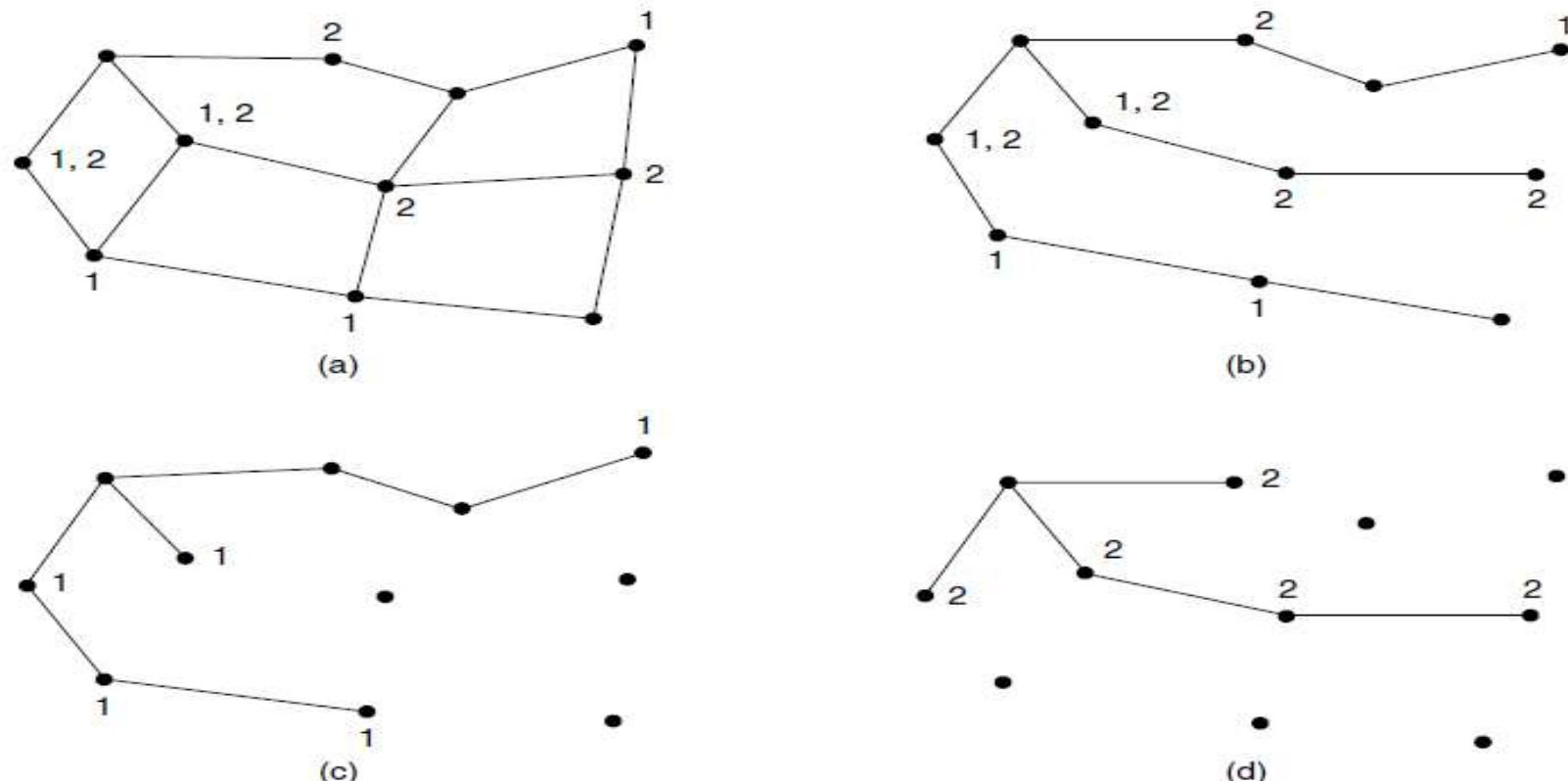
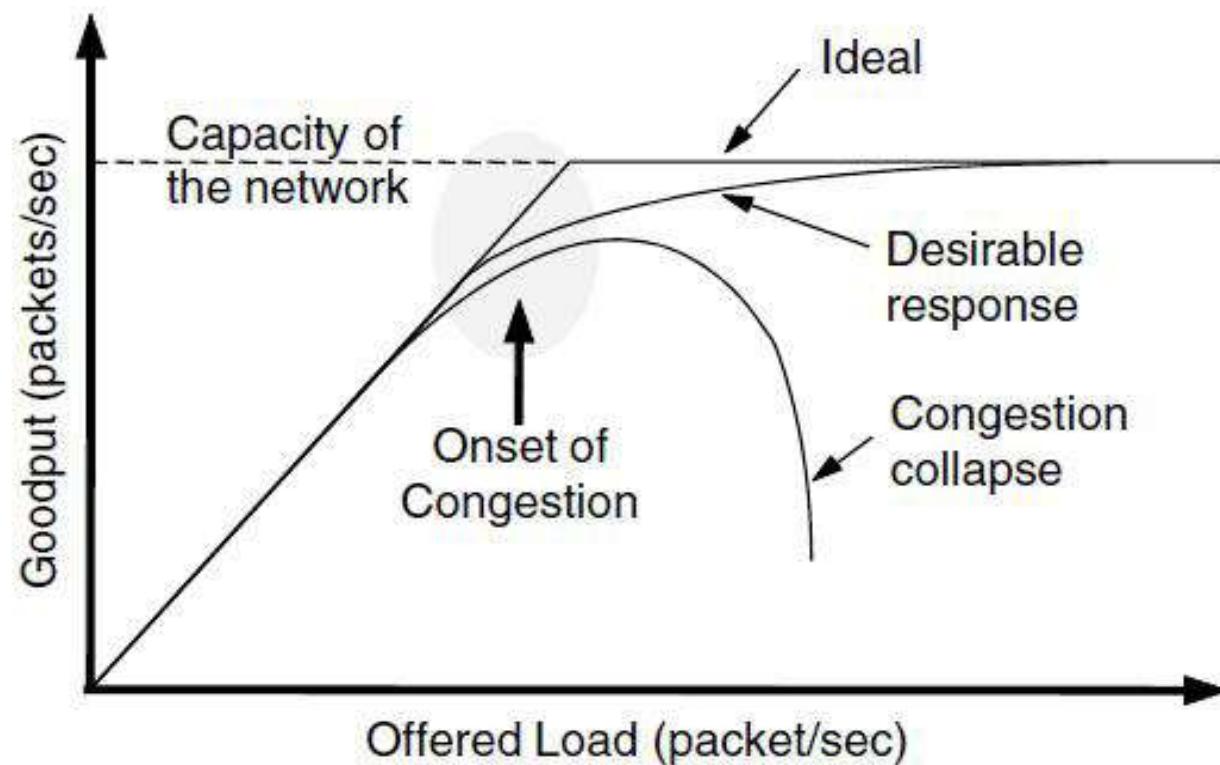


Figure 5-16. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

CONGESTION CONTROL ALGORITHMS

Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called **congestion**. The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets.



Difference between congestion control and flow control

Congestion control has to do with making sure the network is able to carry the offered traffic. It is a global issue, involving the behavior of all the hosts and routers.

Flow control, in contrast, relates to the traffic between a particular sender and a particular receiver. Its job is to make sure that a fast sender cannot continually transmit data faster than the receiver is able to absorb it.

To see the difference between these two concepts, consider a network made up of 100-Gbps fiber optic links on which a supercomputer is trying to force feed a large file to a personal computer that is capable of handling only 1 Gbps.

Several techniques can be employed. These include:

1. Warning bit
2. Choke packets
3. Load shedding
4. Random early discard
5. Traffic shaping

Choke Packets

1. A more direct way of telling the source to slow down.
2. A choke packet is a control packet generated at a congested node and transmitted to restrict traffic flow.
3. The source, on receiving the choke packet must reduce its transmission rate by a certain percentage.

An example of a choke packet is the **Internet Control Message Protocol(ICMP)Source Quench Packet.**

Hop-by-Hop Choke Packets

1. Over long distances or at high speeds choke packets are not very effective.
2. A more efficient method is to send to choke packets hop-by-hop.
3. This requires each hop to reduce its transmission even before the choke packet arrive at the source

Load Shedding

1. When buffers become full, routers simply discard packets.
2. Which packet is chosen to be the victim depends on the application and on the error strategy used in the data link layer.
3. For a file transfer, for, e.g. cannot discard older packets since this will cause a gap in the received data.
4. For real-time voice or video it is probably better to throw away old data and keep new packets.
5. Get the application to mark packets with discard priority.

Random Early Discard (RED)

1. This is a proactive approach in which the router discards one or more packets *before* the buffer becomes completely full.
2. Each time a packet arrives, the RED algorithm computes the average queue length, *avg*.
3. If *avg* is lower than some lower threshold, congestion is assumed to be minimal or non-existent and the packet is queued.
4. If *avg* is greater than some upper threshold, congestion is assumed to be serious and the packet is discarded.

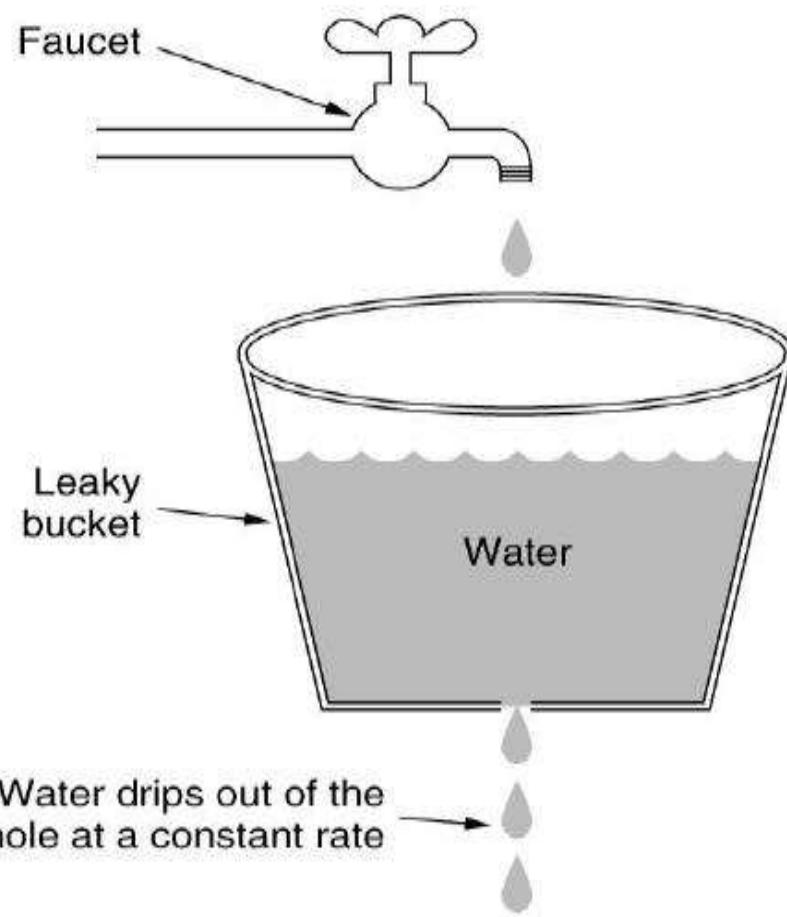
Traffic Shaping

1. Another method of congestion control is to “shape” the traffic before it enters the network.
2. Traffic shaping controls the *rate* at which packets are sent (not just how many). Used in ATM and Integrated Services networks.
3. At connection set-up time, the sender and carrier negotiate a traffic pattern (shape).

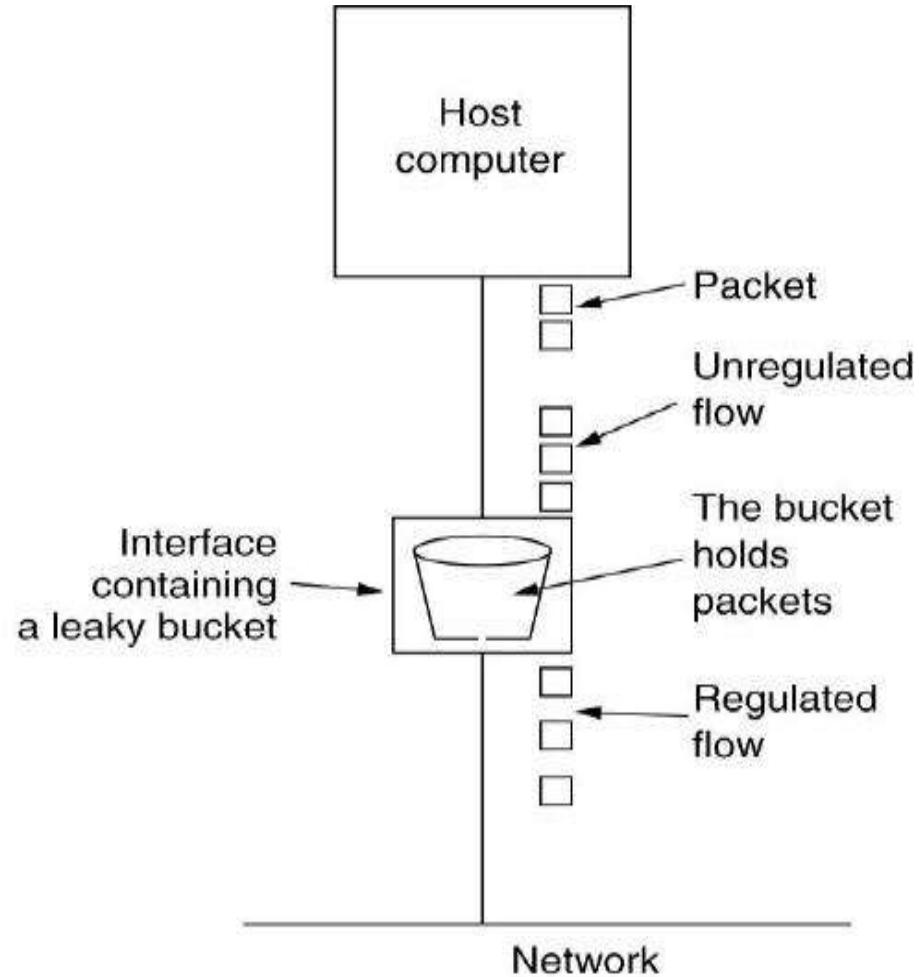
Two traffic shaping algorithms are:

- Leaky Bucket
- Token Bucket

The **Leaky Bucket Algorithm** used to control rate in a network. It is implemented as a single server queue with constant service time. If the bucket (buffer) overflows then packets are discarded.



(a) A leaky bucket with water.

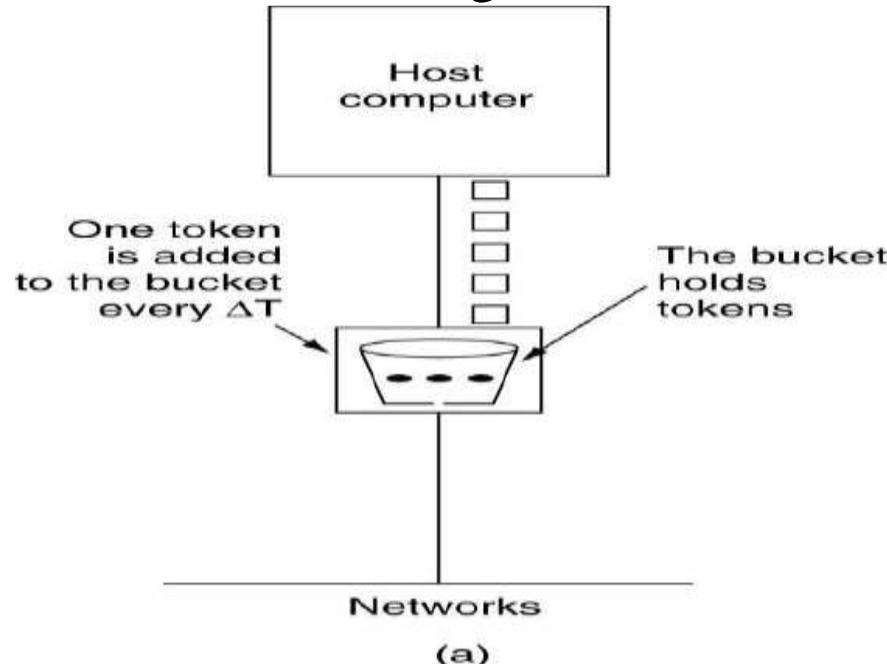


(b) a leaky bucket with packets.

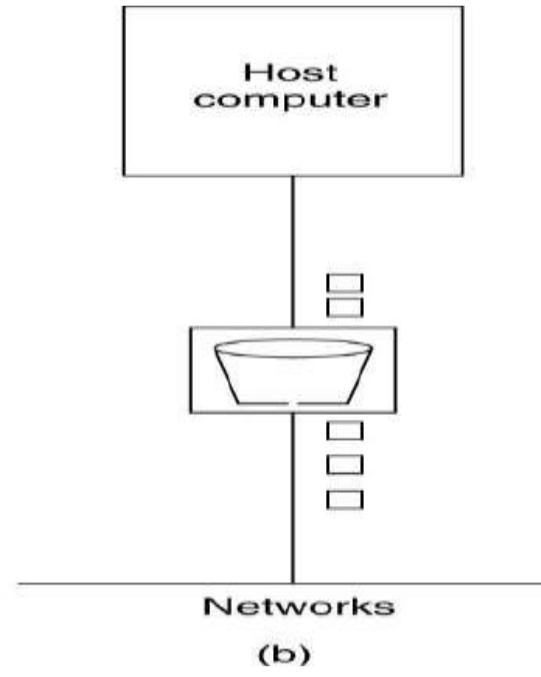
1. The leaky bucket enforces a constant output rate (average rate) regardless of the burstiness of the input. Does nothing when input is idle.
2. The host injects one packet per clock tick onto the network. This results in a uniform flow of packets, smoothing out bursts and reducing congestion.
3. When packets are the same size (as in ATM cells), the one packet per tick is okay. For variable length packets though, it is better to allow a fixed number of bytes per tick. E.g. 1024 bytes per tick will allow one 1024-byte packet or two 512-byte packets or four 256- byte packets on 1 tick

Token Bucket Algorithm

1. In contrast to the LB, the Token Bucket Algorithm, allows the output rate to vary, depending on the size of the burst.
2. In the TB algorithm, the bucket holds tokens. To transmit a packet, the host must capture and destroy one token.
3. Tokens are generated by a clock at the rate of one token every Δt sec.
4. Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.



a) Before.



(b) After.

Leaky Bucket vs. Token Bucket

1. LB discards packets; TB does not. TB discards tokens.
2. With TB, a packet can only be transmitted if there are enough tokens to cover its length in bytes.
3. LB sends packets at an average rate. TB allows for large bursts to be sent faster by speeding up the output.
4. TB allows saving up tokens (permissions) to send large bursts. LB does not allow saving.

QUALITY OF SERVICE

An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it. The name for this solution is **overprovisioning**. The resulting network will carry application traffic without significant loss and, assuming a decent routing scheme, will deliver packets with low latency.

Four issues must be addressed to ensure quality of service:

1. What applications need from the network.
2. How to regulate the traffic that enters the network.
3. How to reserve resources at routers to guarantee performance.
4. Whether the network can safely accept more traffic.

Application Requirements

A stream of packets from a source to a destination is called a **flow**. A flow might be all the packets of a connection in a connection-oriented network, or all the packets sent from one process to another process in a connectionless network. The needs of each flow can be characterized by four primary parameters: bandwidth, delay, jitter, and loss. Together, these determine the **QoS (Quality of Service)** the flow requires.

Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

Figure 5-27. Stringency of applications' quality-of-service requirements.

Traffic Shaping

Before the network can make QoS guarantees, it must know what traffic is being guaranteed. In the telephone network, this characterization is simple. For example, a voice call (in uncompressed format) needs 64 kbps and consists of one 8-bit sample every 125 μ sec. However, traffic in data networks is **bursty**.

Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network.

Traffic shaping reduces congestion and thus helps the network live up to its promise.

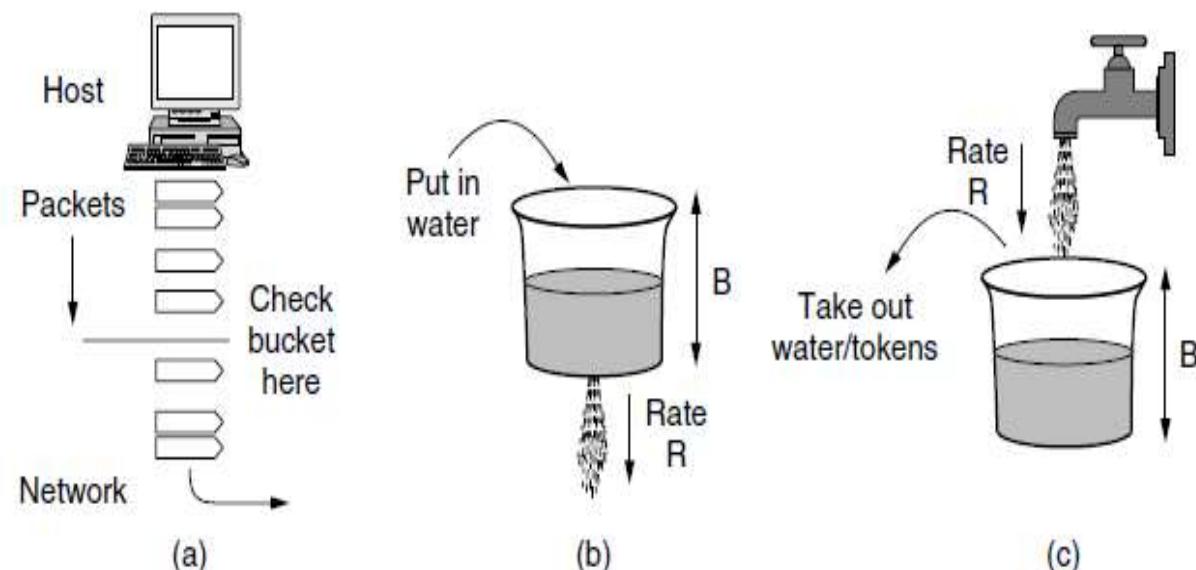


Figure 5-28. (a) Shaping packets. (b) A leaky bucket. (c) A token bucket.

Packet Scheduling

- Algorithms that allocate router resources among the packets of a flow and between competing flows are called **packet scheduling algorithms**. Three different kinds of resources can potentially be reserved for different flows:
 1. Bandwidth.
 2. Buffer space.
 3. CPU cycles.
- The first one, bandwidth, is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work. Thus, reserving bandwidth means not oversubscribing any output line.
- A second resource that is often in short supply is buffer space. When a packet arrives, it is buffered inside the router until it can be transmitted on the chosen outgoing line. The purpose of the buffer is to absorb small bursts of traffic as the flows contend with each other. If no buffer is available, the packet has to be discarded since there is no place to put it.

Admission Control

QoS guarantees are established through the process of admission control. We first saw admission control used to control congestion, which is a performance guarantee, albeit a weak one. The guarantees we are considering now are stronger, but the model is the same. The user offers a flow with an accompanying QoS requirement to the network. The network then decides whether to accept or reject the flow based on its capacity and the commitments it has made to other flows. If it accepts, the network reserves capacity in advance at routers to guarantee QoS when traffic is sent on the new flow.

INTERNETWORKING

How Networks Differ

Networks can differ in many ways. Some of the differences, such as different modulation techniques or frame formats, are internal to the physical and data link layers. These differences will not concern us here.

Item	Some Possibilities
Service offered	Connectionless versus connection oriented
Addressing	Different sizes, flat or hierarchical
Broadcasting	Present or absent (also multicast)
Packet size	Every network has its own maximum
Ordering	Ordered and unordered delivery
Quality of service	Present or absent; many different kinds
Reliability	Different levels of loss
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, packet, byte, or not at all

Figure 5-38. Some of the many ways networks can differ.

How Networks Can Be Connected

There are two basic choices for connecting different networks: we can build devices that translate or convert packets from each kind of network into packets for each other network, or, like good computer scientists, we can try to solve the problem by adding a layer of indirection and building a common layer on top of the different networks. In either case, the devices are placed at the boundaries between networks.

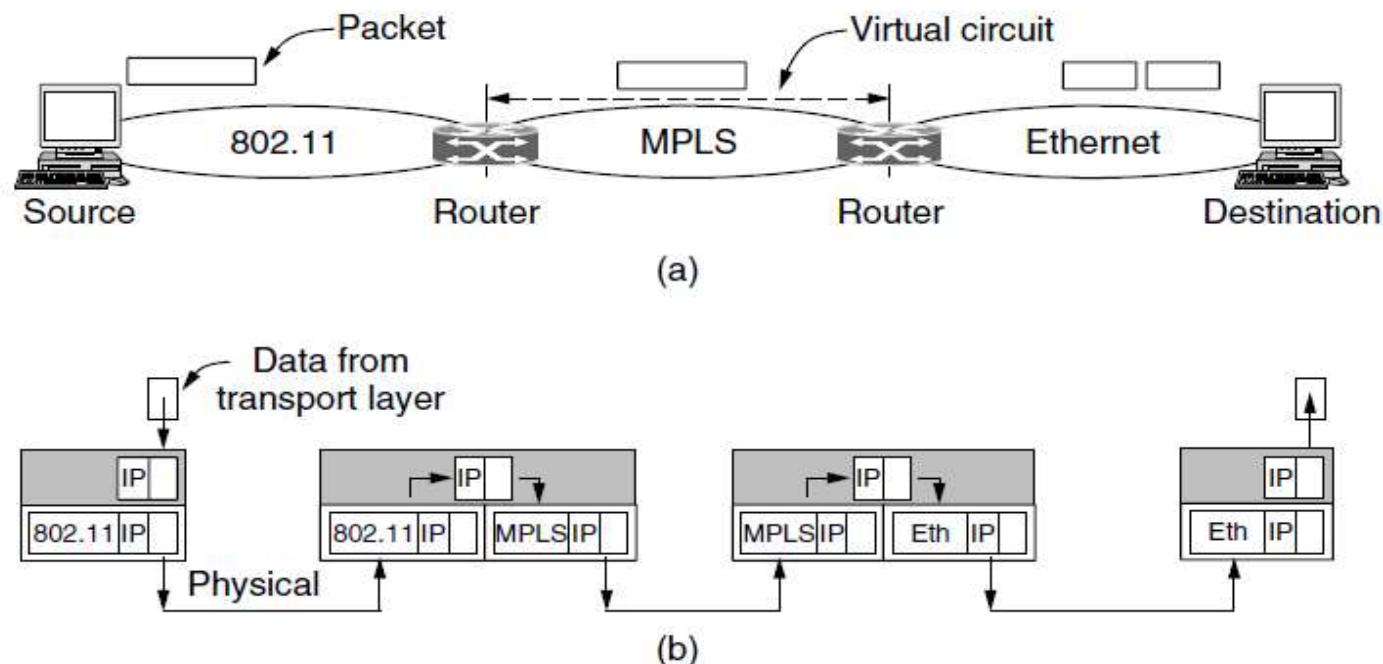


Figure 5-39. (a) A packet crossing different networks. (b) Network and link layer protocol processing.

Tunneling

To send an IP packet to a host in the London office, a host in the Paris office constructs the packet containing an IPv6 address in London, and sends it to the multiprotocol router that connects the Paris IPv6 network to the IPv4 Internet. When this router gets the IPv6 packet, it encapsulates the packet with an IPv4 header addressed to the IPv4 side of the multiprotocol router that connects to the London IPv6 network. That is, the router puts a (IPv6) packet inside a (IPv4) packet. When this wrapped packet arrives, the London router removes the original IPv6 packet and sends it onward to the destination host.

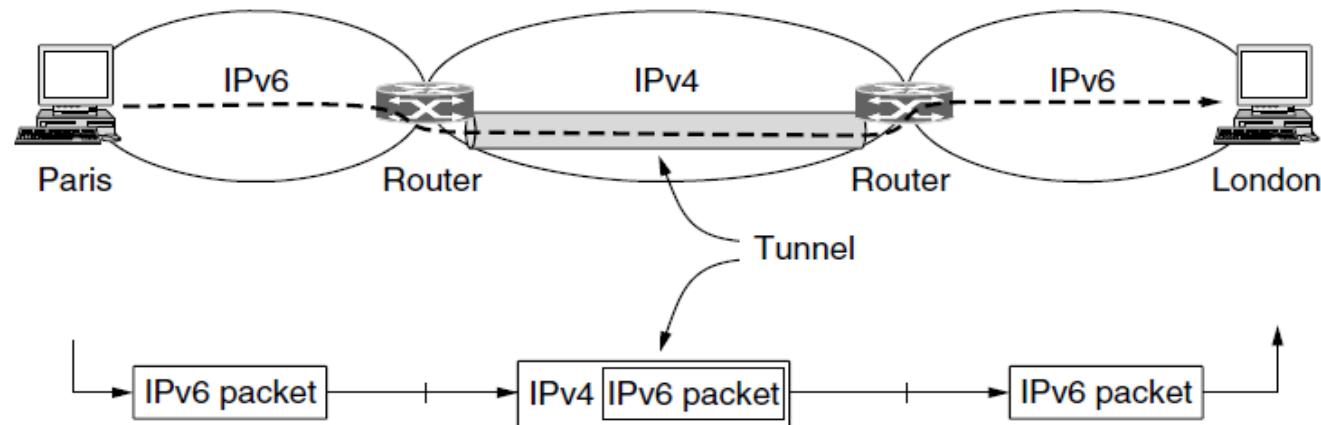


Figure 5-40. Tunneling a packet from Paris to London.

Internet Routing

Routing through an internet poses the same basic problem as routing within a single network, but with some added complications. To start, the networks may internally use different routing algorithms. For example, one network may use link state routing and another distance vector routing. Since link state algorithms need to know the topology but distance vector algorithms do not, this difference alone would make it unclear how to find the shortest paths across the internet.

Within each network, an **intradomain** or **interior gateway protocol** is used for routing. (“Gateway” is an older term for “router.”) It might be a link state protocol of the kind we have already described. Across the networks that make up the internet, an **interdomain** or **exterior gateway protocol** is used.

Packet Fragmentation

Each network or link imposes some maximum size on its packets. These limits have various causes, among them

1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.

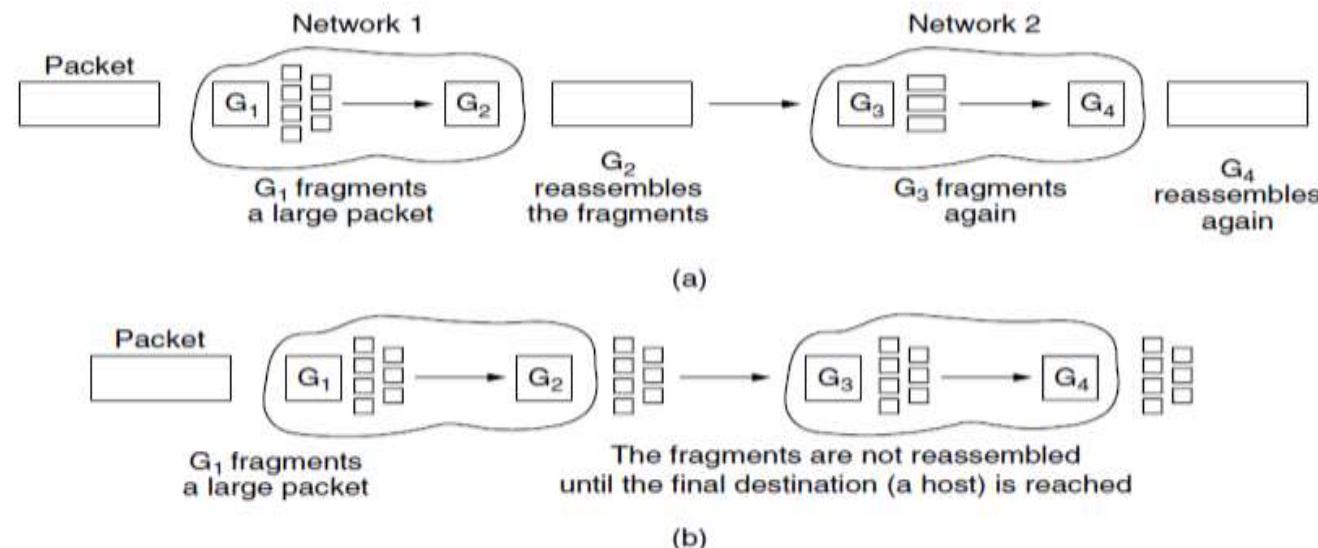
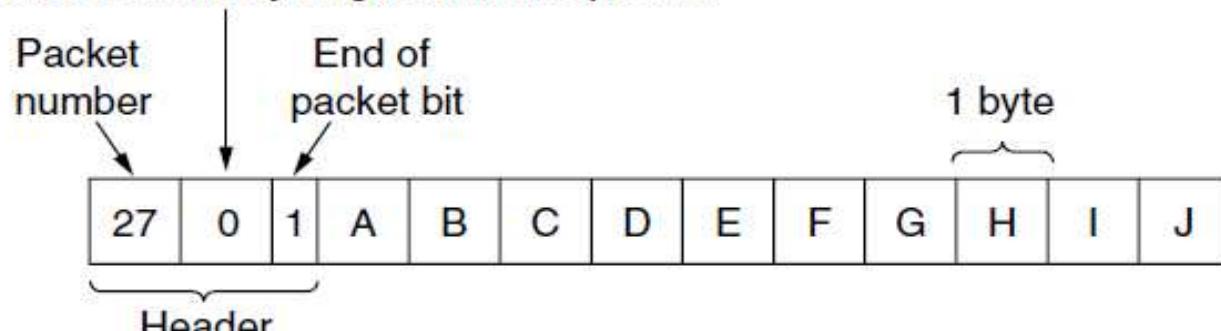
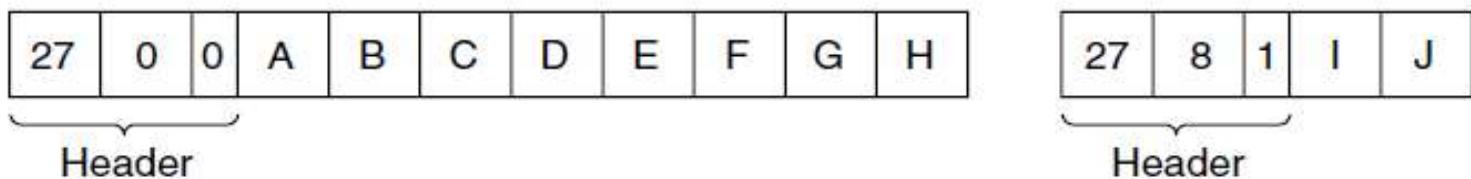


Figure 5-42. (a) Transparent fragmentation. (b) Nontransparent fragmentation.

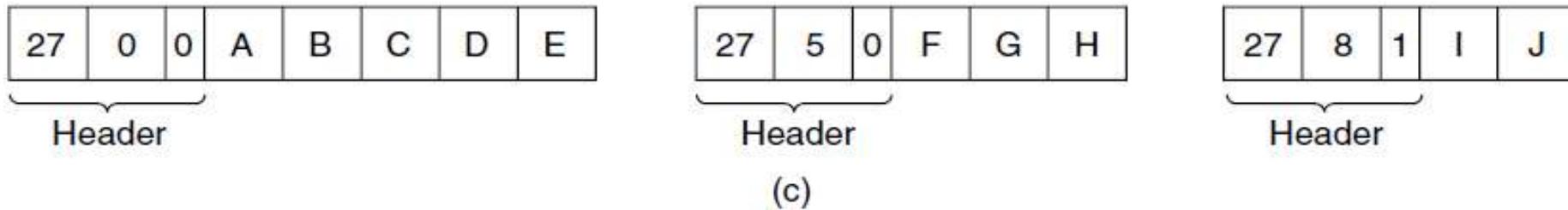
Number of the first elementary fragment in this packet



(a)



(b)



(c)

Figure 5-43. Fragmentation when the elementary data size is 1 byte. (a) Original packet, containing 10 data bytes. (b) Fragments after passing through a network with maximum packet size of 8 payload bytes plus header. (c) Fragments after passing through a size 5 gateway.

THE NETWORK LAYER IN THE INTERNET

the top 10 principles (from most important to least important).

Make sure it works. Do not finalize the design or standard until multiple prototypes have successfully communicated with each other. All too often, designers first write a 1000-page standard, get it approved, then discover it is deeply flawed and does not work. Then they write version 1.1 of the standard. This is not the way to go.

Keep it simple. When in doubt, use the simplest solution. William of Occam stated this principle (Occam's razor) in the 14th century. Put in modern terms: fight features. If a feature is not absolutely essential, leave it out, especially if the same effect can be achieved by combining other features.

Make clear choices. If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble. Standards often have multiple options or modes or parameters because several powerful parties insist that their way is best. Designers should strongly resist this tendency. Just say no.

Exploit modularity. This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if circumstances require one module or layer to be changed, the other ones will not be affected.

Expect heterogeneity. Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.

Avoid static options and parameters. If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value rather than defining fixed choices.

Look for a good design; it need not be perfect. Often, the designers have a good design but it cannot handle some weird special case. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.

- **Be strict when sending and tolerant when receiving.** In other words, send only packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.
- **Think about scalability.** If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.
- **Consider performance and cost.** If a network has poor performance or outrageous costs, nobody will use it.

The IP Version 4 Protocol (IPV4)

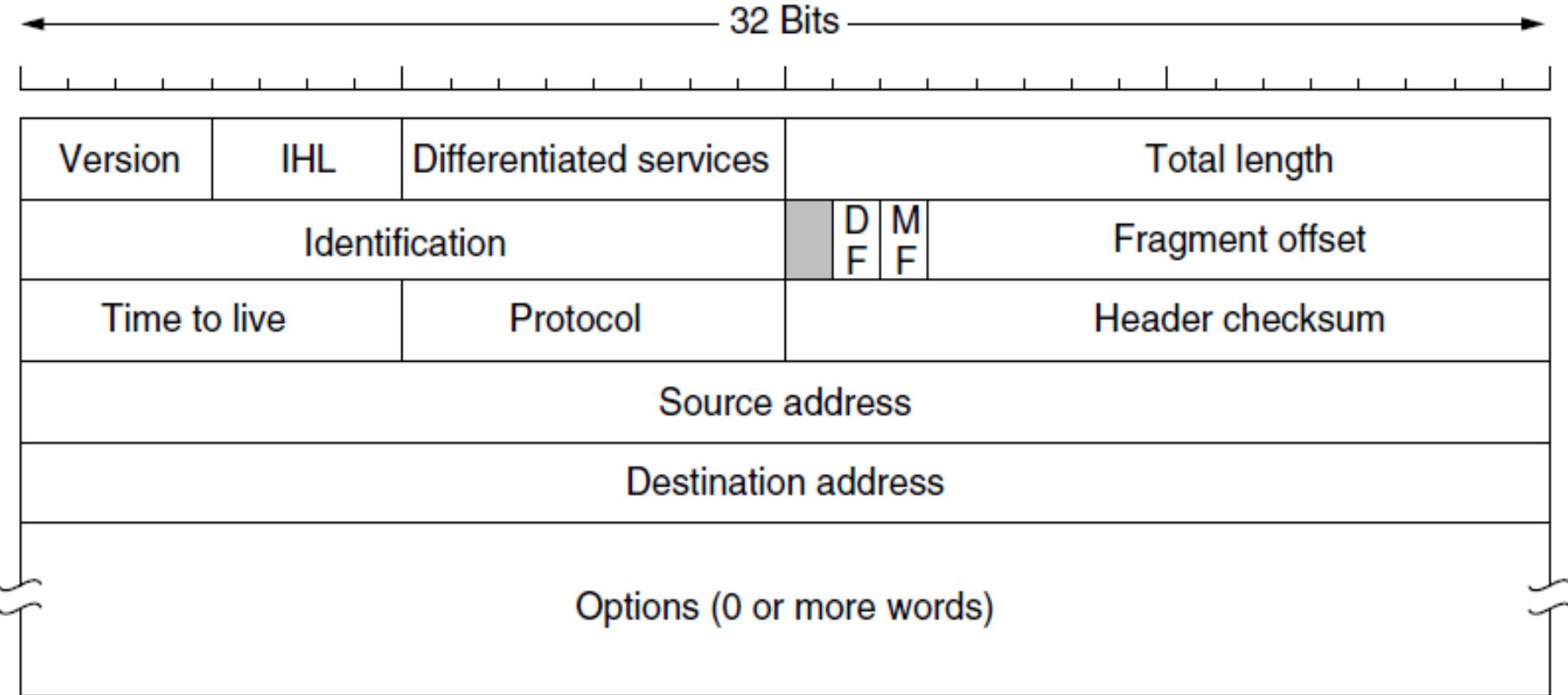


Figure 5-46. The IPv4 (Internet Protocol) header.

The *Version* field keeps track of which version of the protocol the datagram belongs to. Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making those options useless.

The *Differentiated services* field is one of the few fields that has changed its meaning (slightly) over the years. Originally, it was called the *Type of service* field. It was and still is intended to distinguish between different classes of service. Various combinations of reliability and speed are possible.

The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes.

The *Identification* field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same *Identification* value.

Next comes an unused bit, which is surprising, as available real estate in the IP header is extremely scarce.

Then come two 1-bit fields related to fragmentation. *DF* stands for Don't Fragment. It is an order to the routers not to fragment the packet.

MF stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.

The *Fragment offset* tells where in the current packet this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit.

The *TtL* (*Time to live*) field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec.

When the network layer has assembled a complete packet, it needs to know what to do with it. The *Protocol* field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet.

Since the header carries vital information such as addresses, it rates its own checksum for protection, the *Header checksum*. The algorithm is to add up all the 16-bit halfwords of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result.

The *Source address* and *Destination address* indicate the IP address of the source and destination network interfaces.

IP Version 6(IPV6)

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP has become a victim of its own popularity: it is close to running out of addresses.

IPv6 (IP version 6) is a replacement design that does just that. It uses 128-bit addresses; a

shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case.

Its major goals were:

1. Support billions of hosts, even with inefficient address allocation.
2. Reduce the size of the routing tables.
3. Simplify the protocol, to allow routers to process packets faster.
4. Provide better security (authentication and privacy).
5. Pay more attention to the type of service, particularly for real-time data.
6. Aid multicasting by allowing scopes to be specified.
7. Make it possible for a host to roam without changing its address.
8. Allow the protocol to evolve in the future.
9. Permit the old and new protocols to coexist for years.

The Main IPv6 Header

The IPv6 header is shown in Fig. 5-56. The *Version* field is always 6 for IPv6 (and 4 for IPv4).

For example, the Ethernet *Type* field has different values to indicate an IPv4 or an IPv6 payload. The discussions between the “Do it right” and “Make it fast” camps will no doubt be lengthy and vigorous.

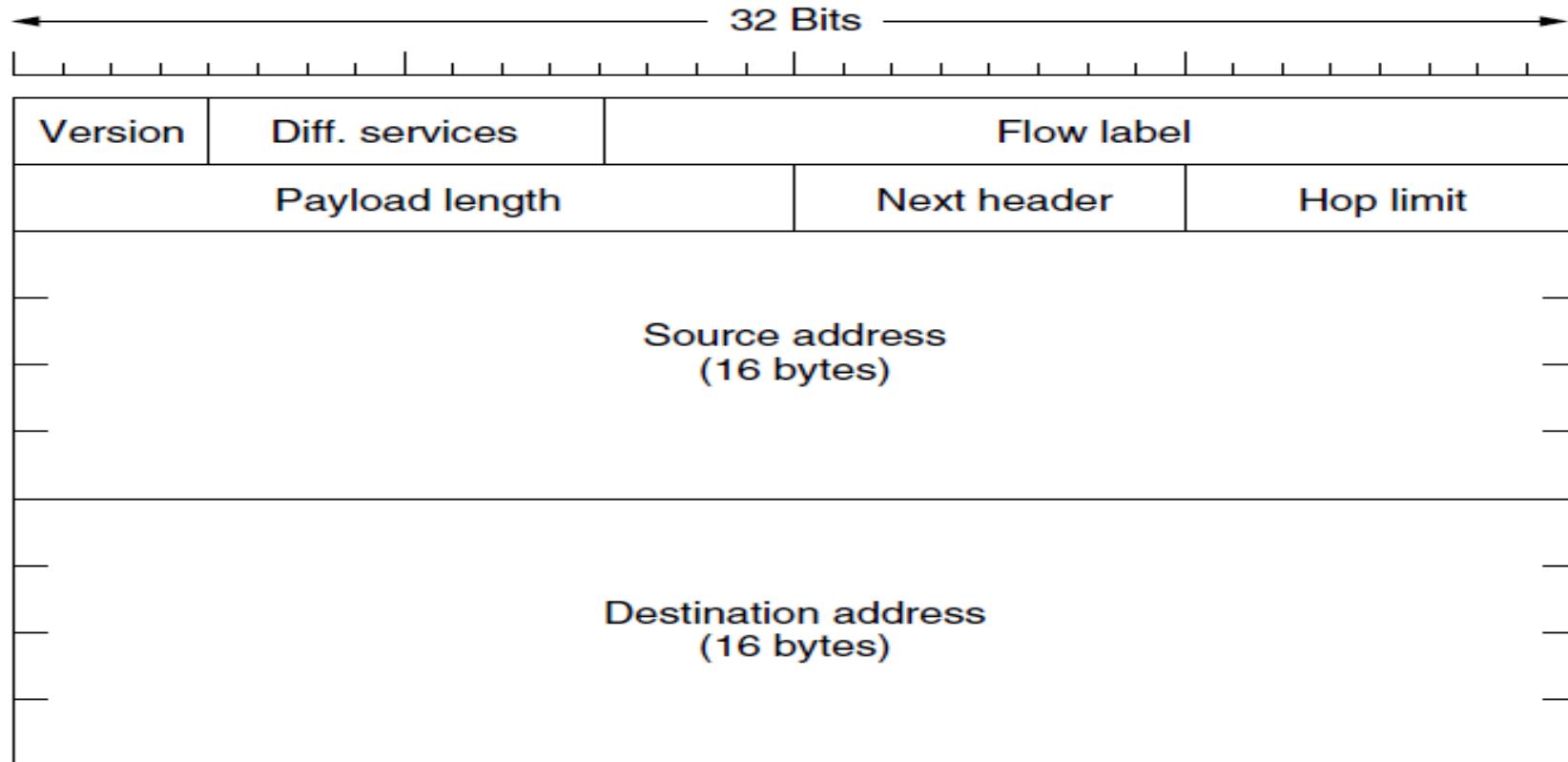


Figure 5-56. The IPv6 fixed header (required).

The *Differentiated services* field (originally called *Traffic class*) is used to distinguish the class of service for packets with different real-time delivery requirements.

The *Flow label* field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudoconnection.

The *Payload length* field tells how many bytes follow the 40-byte header
The *Next header* field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers.

The *Hop limit* field is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4

Next come the *Source address* and *Destination address* fields.

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes

8000::123:4567:89AB:CDEF

Finally, IPv4 addresses can be written as a pair of colons and an old dotted decimal number, for example:

::192.31.20.46

INTERNET CONTROL PROTOCOLS

In addition to IP, which is used for data transfer, the Internet has several companion control protocols that are used in the network layer. They include ICMP, ARP, and DHCP.

IMCP—The Internet Control Message Protocol

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the **ICMP (Internet Control Message Protocol)**. ICMP is also used to test the Internet.

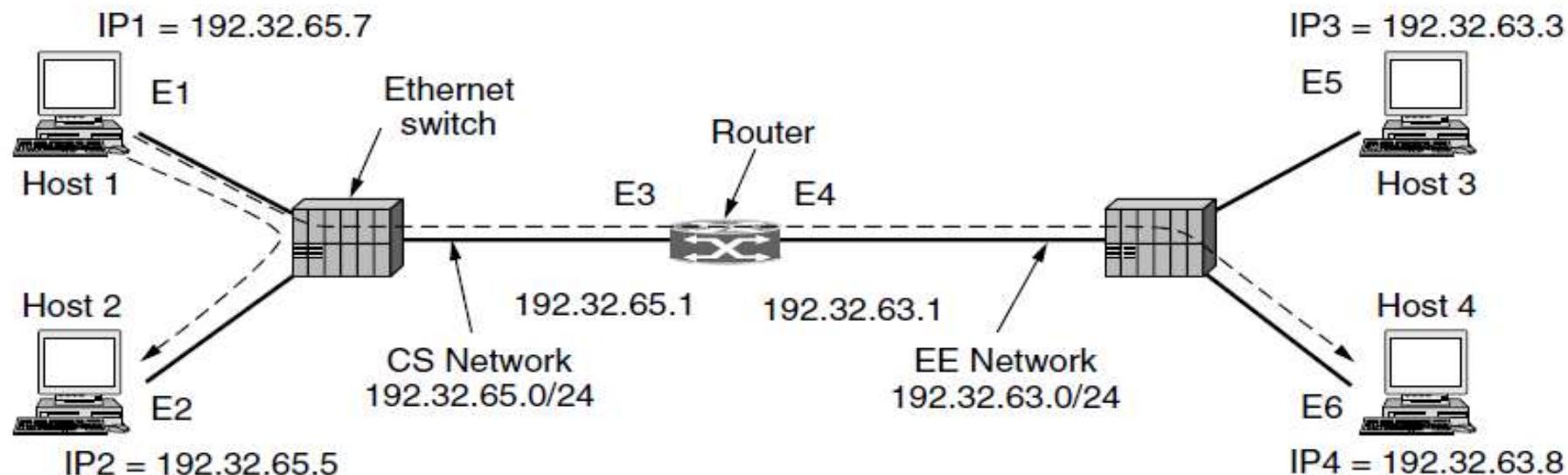
The DESTINATION UNREACHABLE message is used when the router cannot locate the destination or when a packet with the *DF* bit cannot be delivered because a “small-packet” network stands in the way.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

Figure 5-60. The principal ICMP message types.

ARP—The Address Resolution Protocol

Although every machine on the Internet has one or more IP addresses, these addresses are not sufficient for sending packets. Data link layer NICs (Network Interface Cards) such as Ethernet cards do not understand Internet addresses.



Frame	Source IP	Source Eth.	Destination IP	Destination Eth.
Host 1 to 2, on CS net	IP1	E1	IP2	E2
Host 1 to 4, on CS net	IP1	E1	IP4	E3
Host 1 to 4, on EE net	IP1	E4	IP4	E6

Figure 5-61. Two switched Ethernet LANs joined by a router.

A better solution is for host 1 to output a broadcast packet onto the Ethernet asking who owns IP address 192.32.65.5. The broadcast will arrive at every machine on the CS Ethernet, and each one will check its IP address. Host 2 alone will respond with its Ethernet address (E_2). In this way host 1 learns that IP address 192.32.65.5 is on the host with Ethernet address E_2 . The protocol used for asking this question and getting the reply is called **ARP (Address Resolution Protocol)**.

DHCP—The Dynamic Host Configuration Protocol

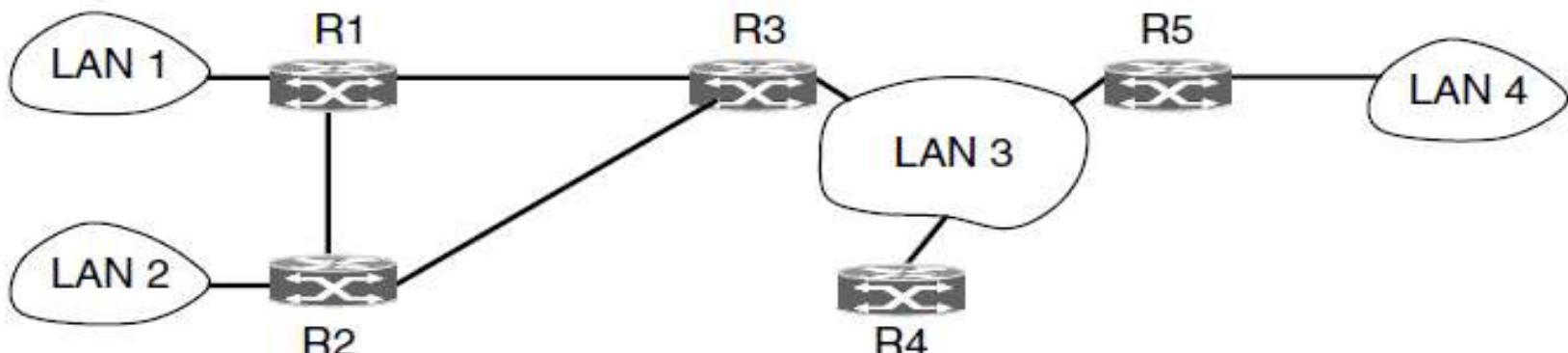
ARP (as well as other Internet protocols) makes the assumption that hosts are configured with some basic information, such as their own IP addresses. How do hosts get this information? It is possible to manually configure each computer, but that is tedious and error-prone. There is a better way, and it is called **DHCP (Dynamic Host Configuration Protocol)**.

With DHCP, every network must have a DHCP server that is responsible for configuration. When a computer is started, it has a built-in Ethernet or other link layer address embedded in the NIC, but no IP address. Much like ARP, the computer broadcasts a request for an IP address on its network. It does this by using a DHCP DISCOVER packet. This packet must reach the DHCP server.

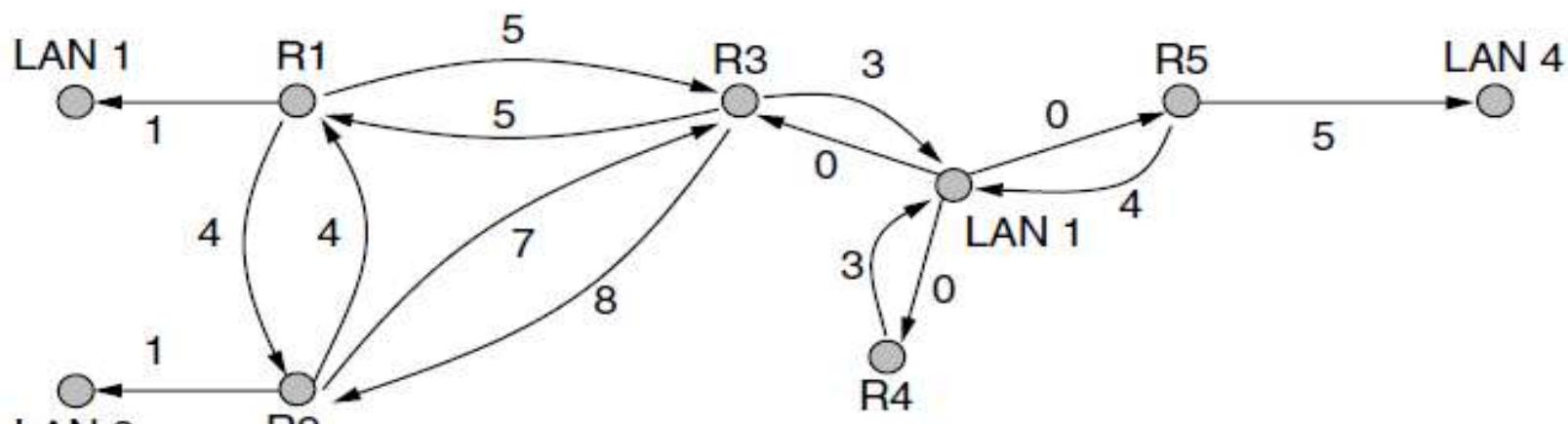
OSPF—An Interior Gateway Routing Protocol

the Internet is made up of a large number of independent networks or **ASes (Autonomous Systems)** that are operated by different organizations, usually a company, university, or ISP. Inside of its own network, an organization can use its own algorithm for internal routing, or **intradomain routing**, as it is more commonly known. Nevertheless, there are only a handful of standard protocols that are popular. An intradomain routing protocol is also called an **interior gateway protocol**.

The ARPANET switched over to a link state protocol in May 1979 because of these problems, and in 1988 IETF began work on a link state protocol for intradomain routing. That protocol, called **OSPF (Open Shortest Path First)**, became a standard in 1990. It drew on a protocol called **IS-IS (Intermediate-System to Intermediate-System)**, which became an ISO standard.



(a)



(b)

Figure 5-64. (a) An autonomous system. (b) A graph representation of (a).

BGP—The Exterior Gateway Routing Protocol

Within a single AS, OSPF and IS-IS are the protocols that are commonly used. Between ASes, a different protocol, called **BGP (Border Gateway Protocol)**, is used. A different protocol is needed because the goals of an intradomain protocol and an interdomain protocol are not the same. All an intradomain protocol has to do is move packets as efficiently as possible from the source to the destination. It does not have to worry about politics.

A few examples of possible routing constraints are:

1. Do not carry commercial traffic on the educational network.
2. Never send traffic from the Pentagon on a route through Iraq.
3. Use TeliaSonera instead of Verizon because it is cheaper.
4. Don't use AT&T in Australia because performance is poor.
5. Traffic starting or ending at Apple should not transit Google.

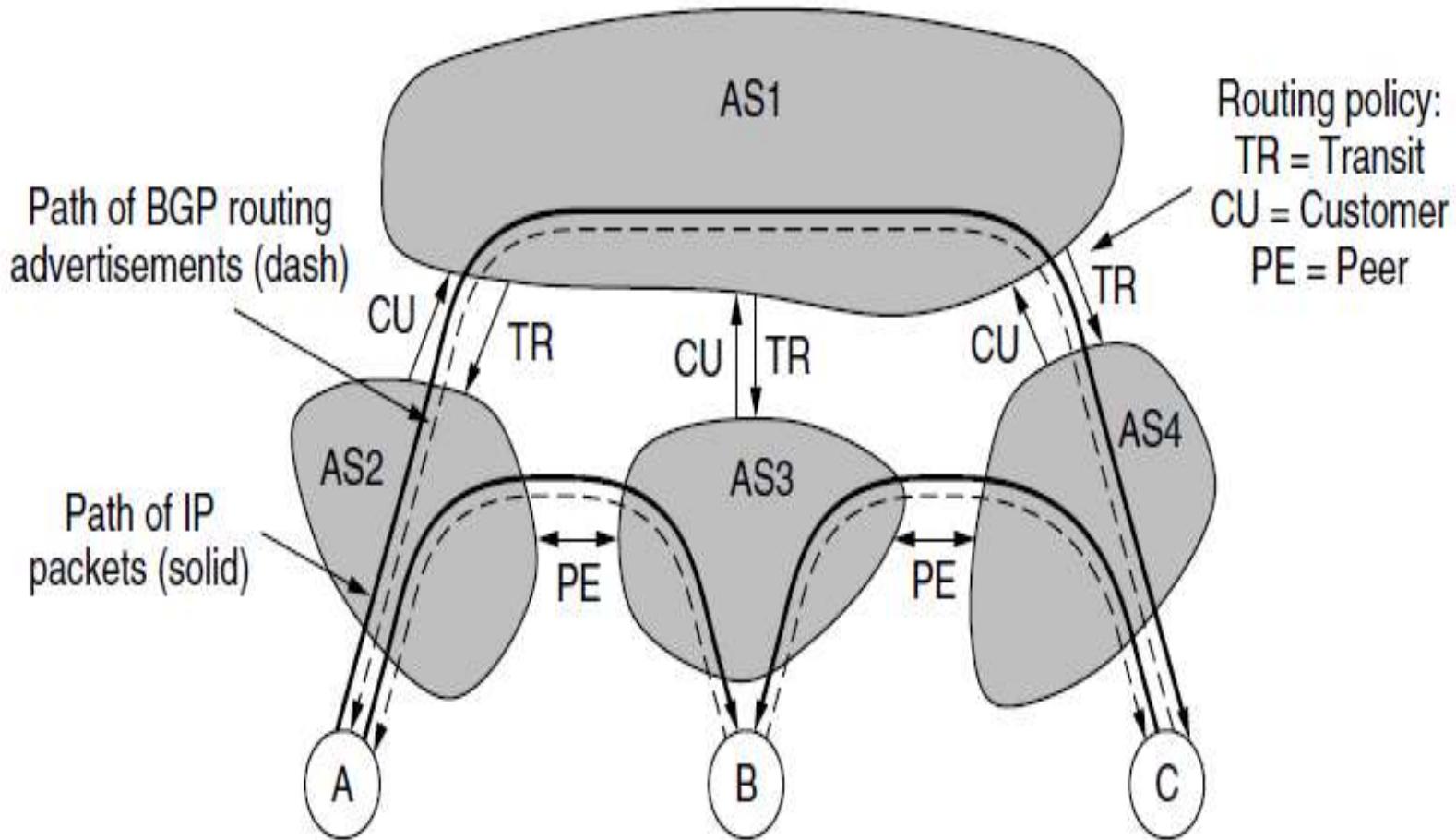


Figure 5-67. Routing policies between four autonomous systems.

UNIT – IV

The Transport Layer: The Transport Service-Elements of Transport Protocols- Congestion Control- The internet transport protocols: UDP, TCP- Performance problems in computer networks

THE TRANSPORT LAYER

Services Provided to the Upper Layers:

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the network layer. The software and/or hardware within the transport layer that does the work is called the **transport entity**.

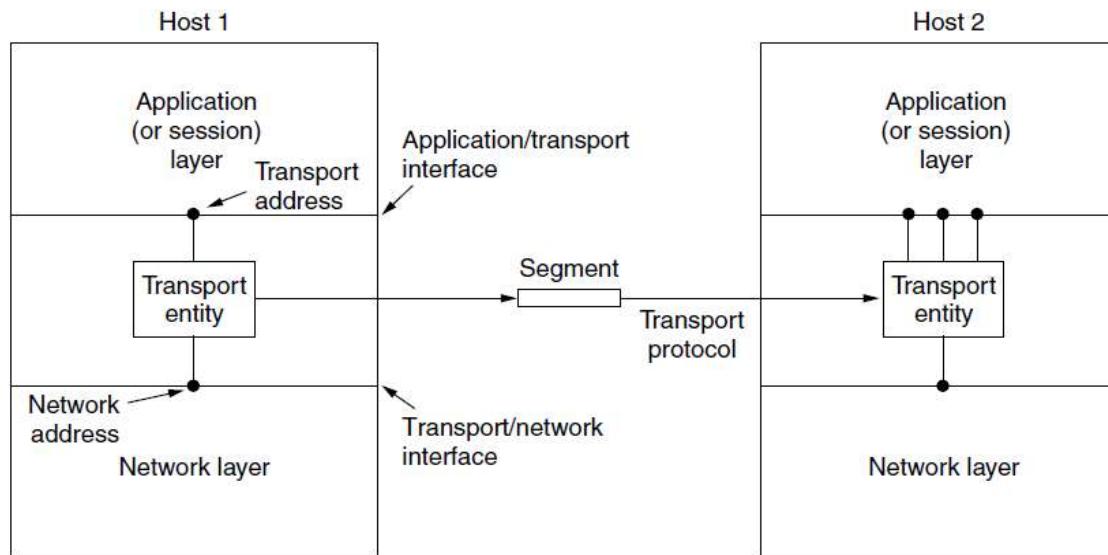


Figure 6-1. The network, transport, and application layers.

Transport Service Primitives:

The transport service is similar to the network service, but there are also some important differences. The main difference is that the network service is intended to model the service offered by real networks, warts and all. Real networks can lose packets, so the network service is generally unreliable.

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	Request a release of the connection

Figure 6-2. The primitives for a simple transport service.

TCP, UDP and other Internet protocols use data as **segment**. Thus, segments (exchanged by the transport layer) are contained in packets (exchanged by the network layer). In turn, these packets are contained in frames (exchanged by the data link layer).

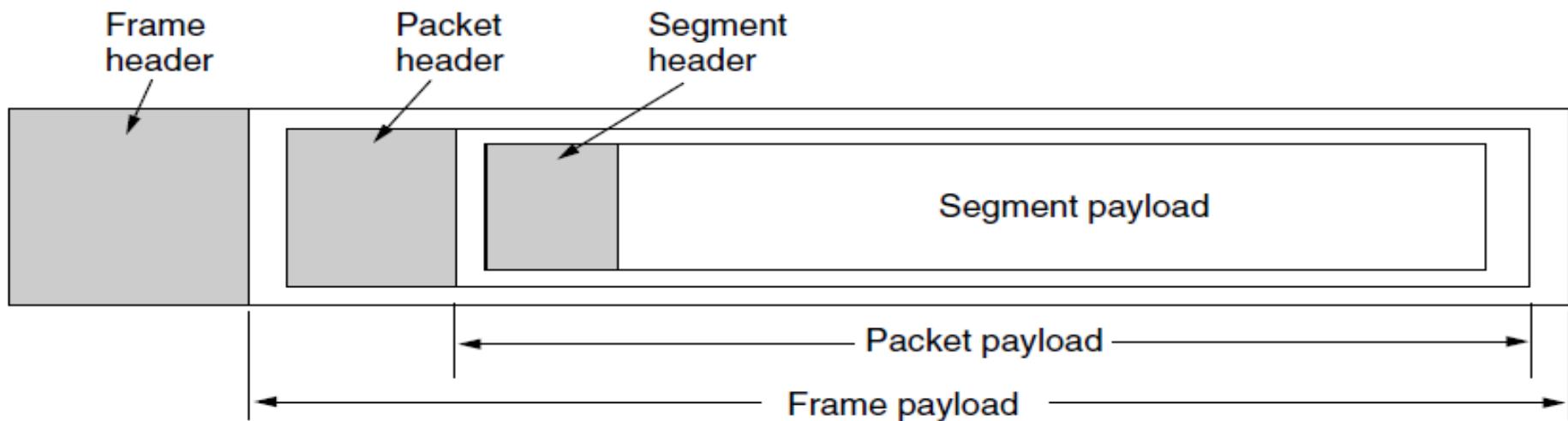


Figure 6-3. Nesting of segments, packets, and frames.

Berkeley Sockets

Sockets were first released as part of the Berkeley UNIX 4.2BSD software distribution in 1983. They quickly became popular. The primitives are now widely used for Internet programming on many operating systems, especially UNIX-based systems, and there is a socket-style API for Windows called “winsock.”

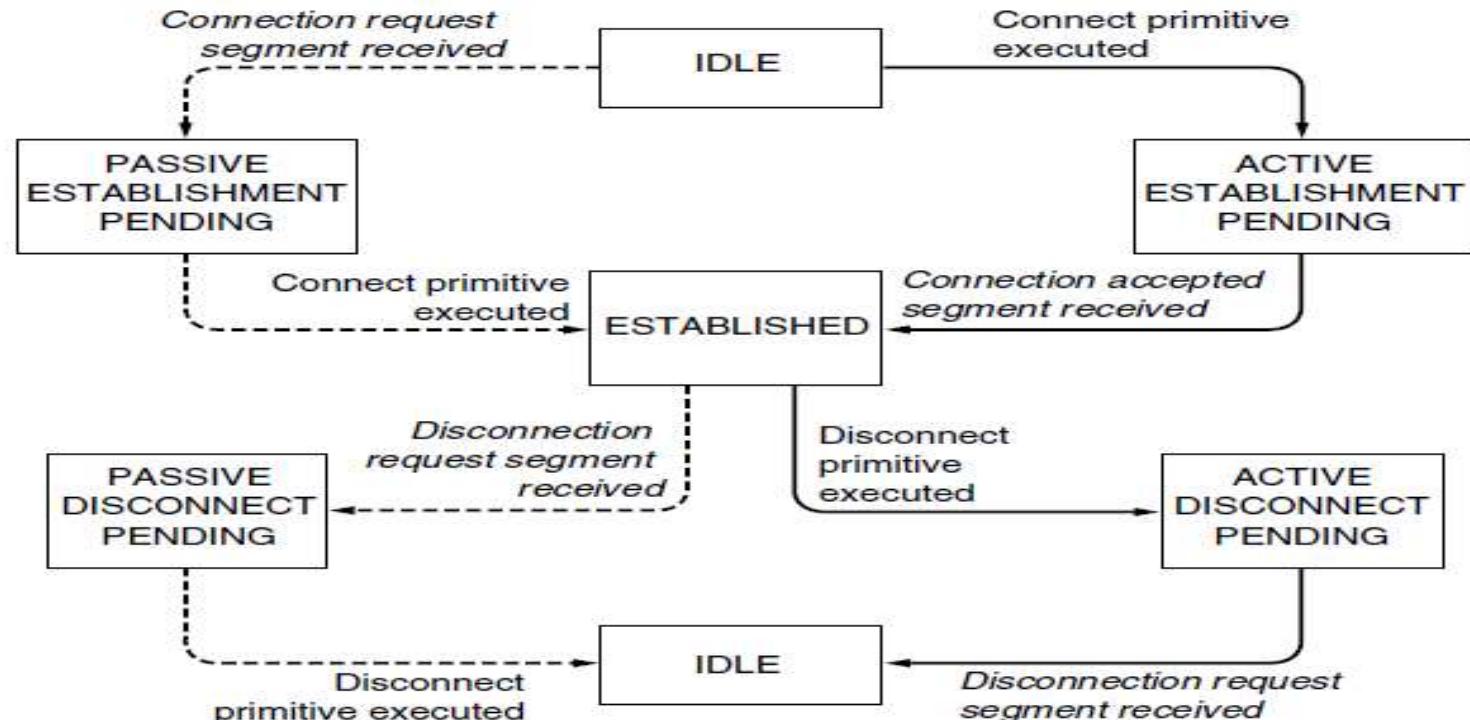


Figure 6-4. A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

the following are the socket primitives for TCP.

Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

ELEMENTS OF TRANSPORT PROTOCOLS

The transport service is implemented by a **transport protocol** used between the two transport entities. At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network.

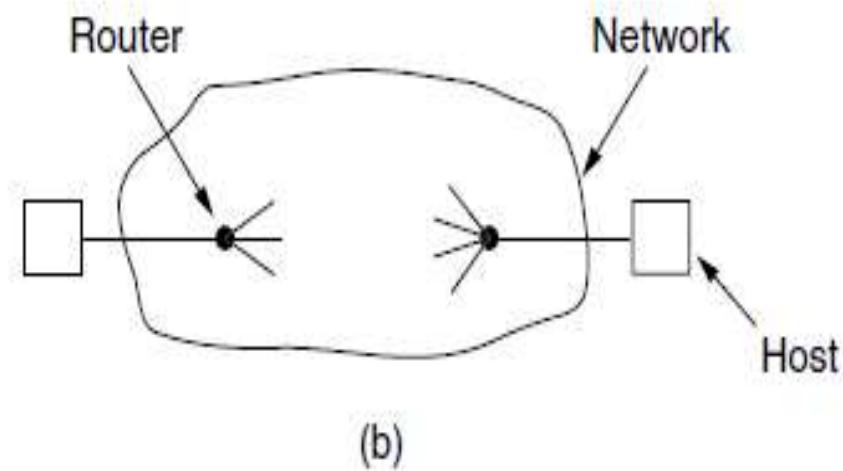
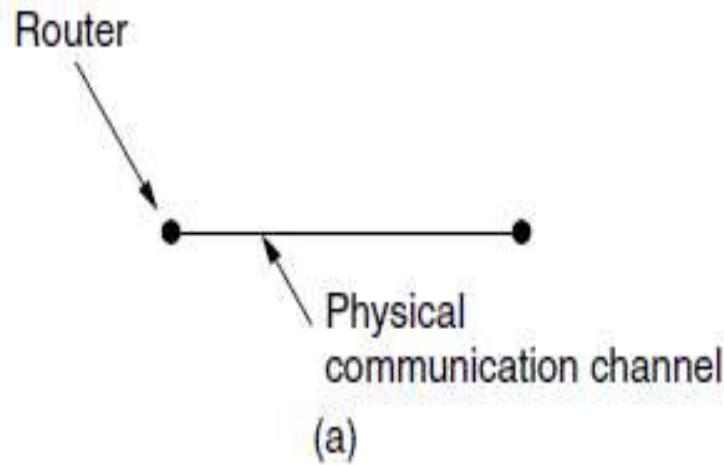


Figure 6-7. (a) Environment of the data link layer. (b) Environment of the transport layer.

For another thing, the process of establishing a connection over the wire of is simple: the other end is always there (unless it has crashed, in which case it is not there).

Another (exceedingly annoying) difference between the data link layer and the transport layer is the potential existence of storage capacity in the network.

A final difference between the data link and transport layers is one of degree rather than of kind. Buffering and flow control are needed in both layers, but the presence in the transport layer of a large and varying number of connections with bandwidth that fluctuates as the connections compete with each other may require a different approach than we used in the data link layer.

Addressing

The end points are called ports. **TSAP** (**T**ransport **S**ervice **A**ccess **P**oint) to mean a specific endpoint in the transport layer. The analogous endpoints in the network layer (i.e., network layer addresses) are not-surprisingly called **NSAPs** (**N**etwork **S**ervice **A**ccess **P**oints). IP addresses are examples of NSAPs.

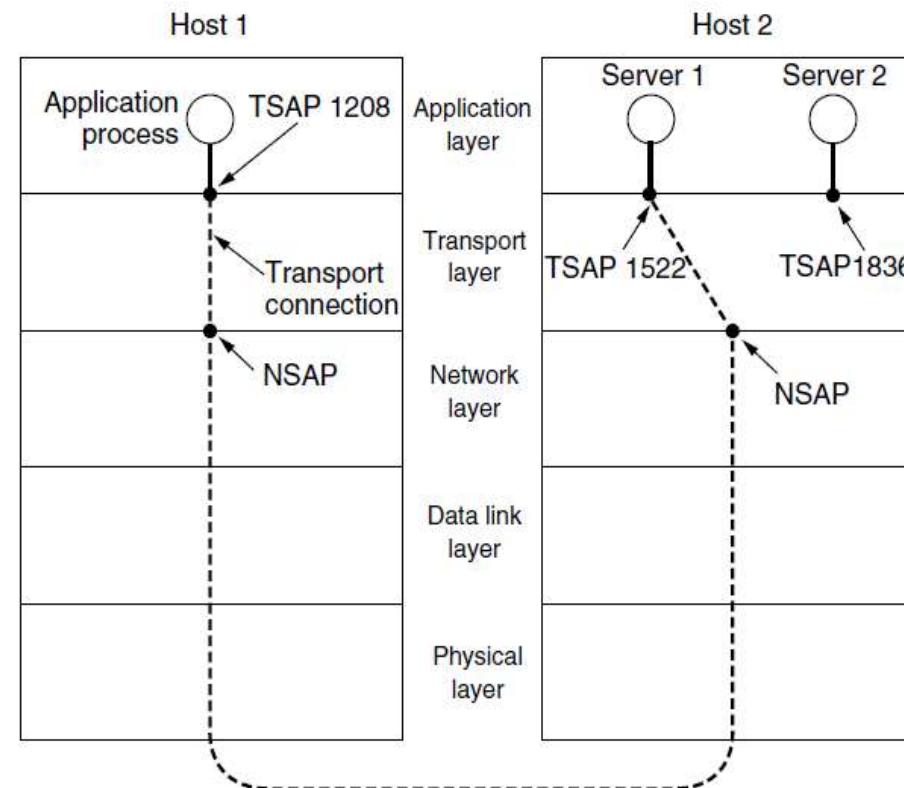


Figure 6-8. TSAPs, NSAPs, and transport connections.

A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.
3. The application process sends over the mail message.
4. The mail server responds to say that it will deliver the message.
5. The transport connection is released.

Connection Establishment

it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, delay, corrupt, and duplicate packets. This behavior causes serious complications.

Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Time stamping each packet.

The first technique includes any method that prevents packets from looping, combined with some way of bounding delay including congestion over the (now known) longest possible path.

The second method consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded.

The third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time.

way
with

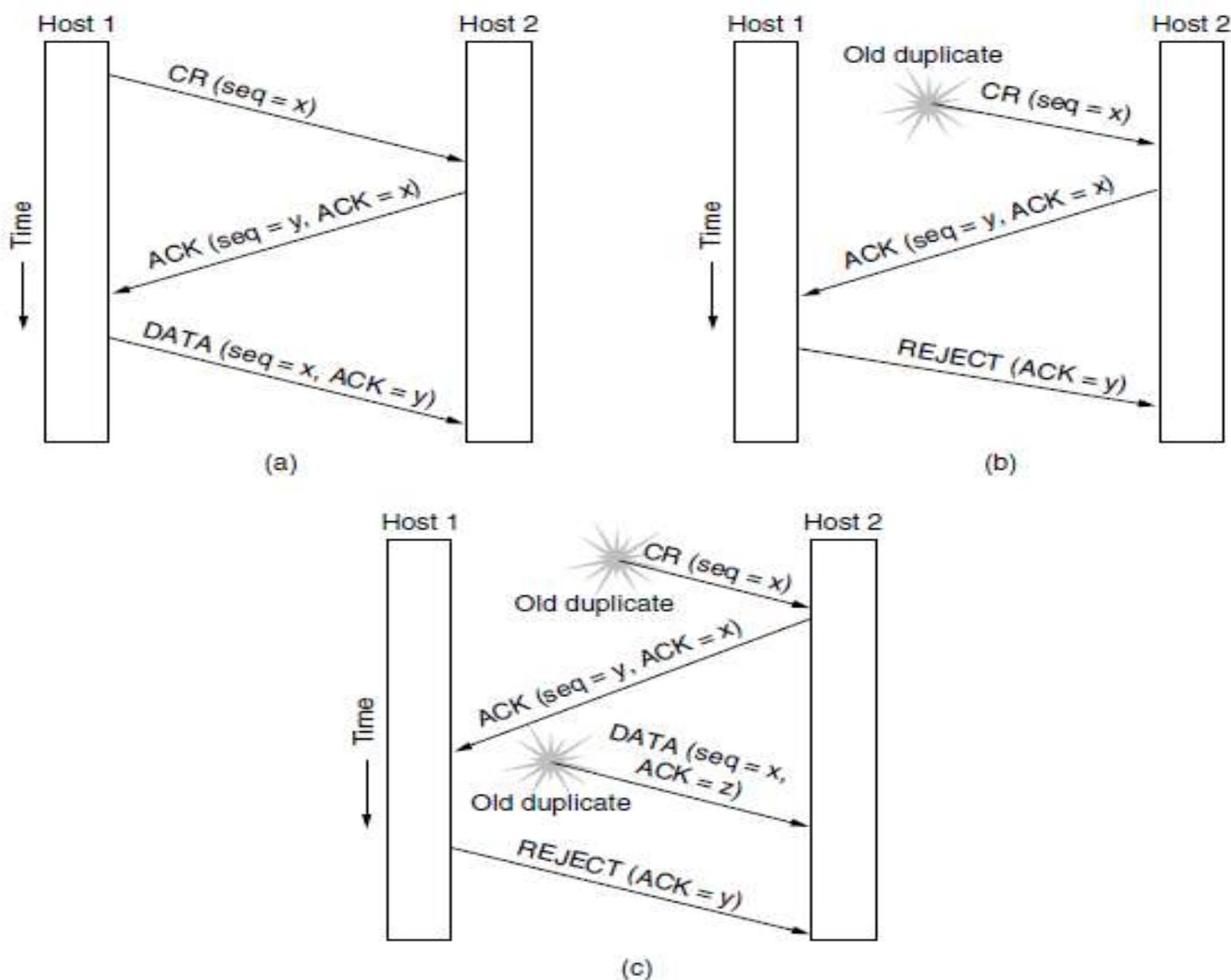


Figure 6-11. Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK.

Host 1 chooses a sequence number, x , and sends a CONNECTION REQUEST segment containing it to host 2. Host 2 replies with an ACK segment acknowledging x and announcing its own initial sequence number, y . Finally, host 1 acknowledges host 2's choice of an initial sequence number in the first data segment that it sends.

In Fig. 6-11(b), the first segment is a delayed duplicate CONNECTION REQUEST from an old connection. This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host 1 an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection. When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.

The worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet. This case is shown in Fig. 6-11(c). As in the previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it.

Connection Release

There are two styles of terminating a connection: asymmetric release and symmetric release.

Asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken. Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately.

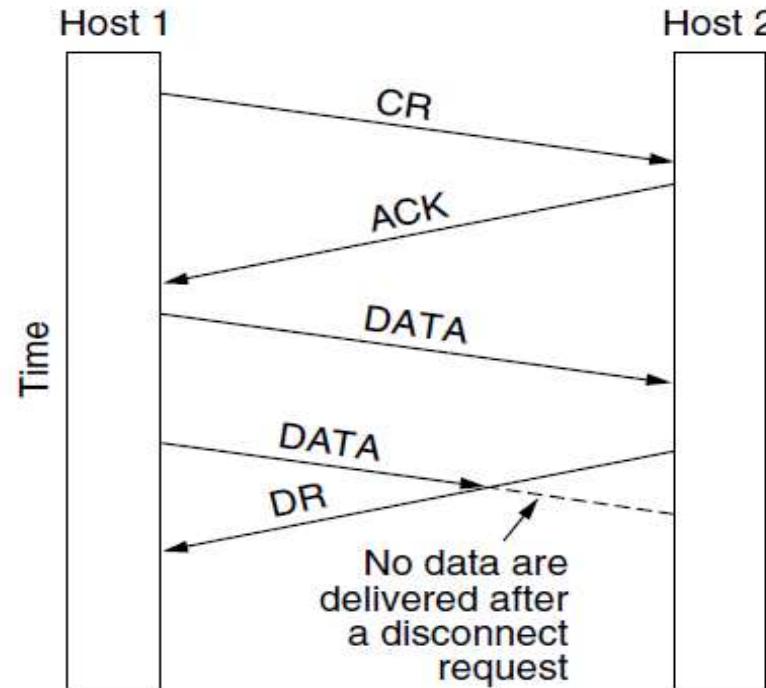


Figure 6-12. Abrupt disconnection with loss of data.

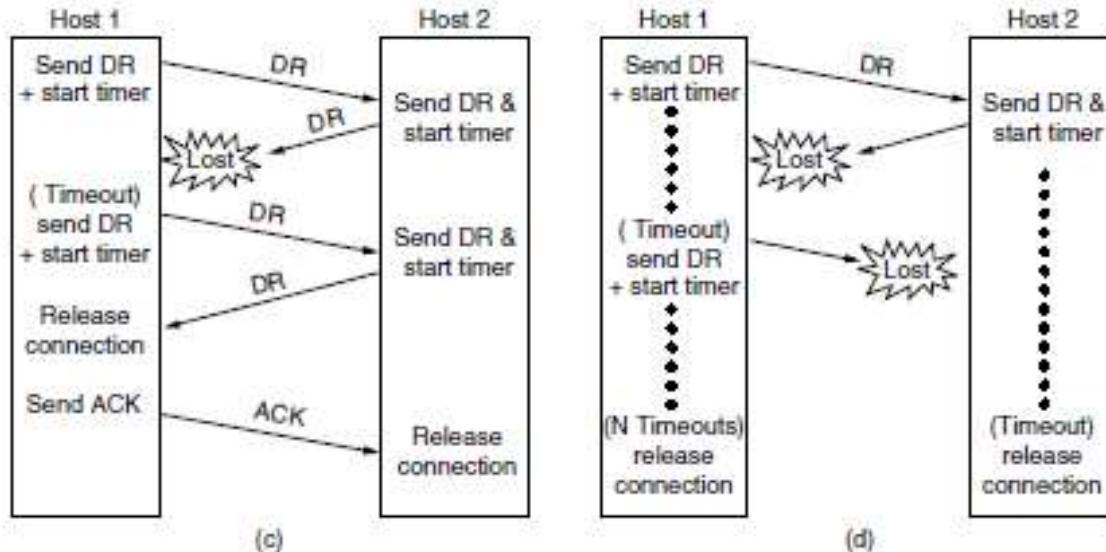
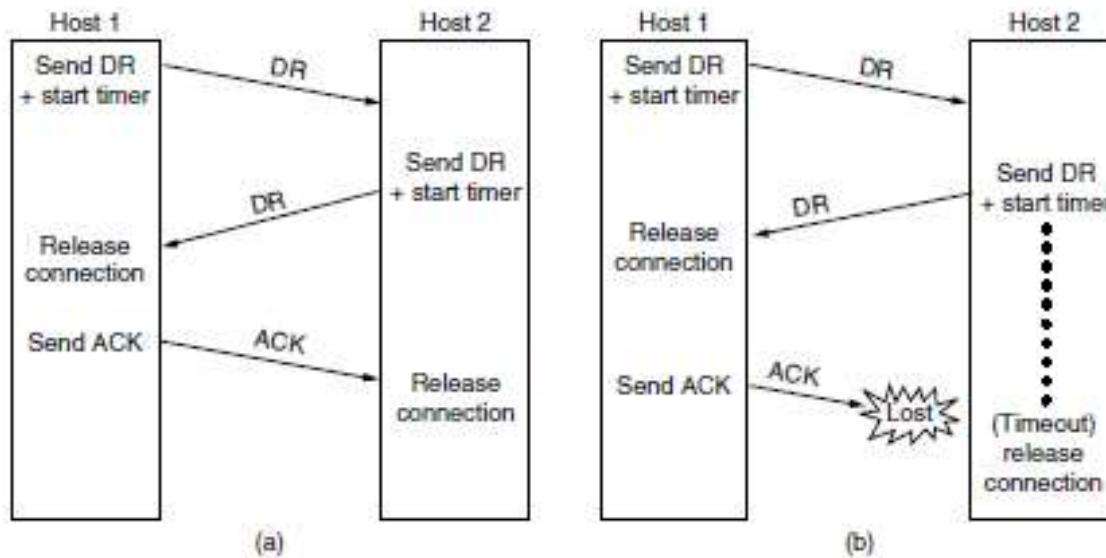


Figure 6-14. Four protocol scenarios for releasing a connection. (a) Normal case of three-way handshake. (b) Final ACK lost. (c) Response lost. (d) Response lost and subsequent DRs lost.

Error Control and Flow Control

Error control is ensuring that the data is delivered with the desired level of reliability, usually that all of the data is delivered without any errors. Flow control is keeping a fast transmitter from overrunning a slow receiver.

- A frame carries an error-detecting code (e.g., a CRC or checksum) that is used to check if the information was correctly received.
- A frame carries a sequence number to identify itself and is retransmitted by the sender until it receives an acknowledgement of successful receipt from the receiver. This is called **ARQ (Automatic Repeat reQuest)**.
- There is a maximum number of frames that the sender will allow to be outstanding at any time, pausing if the receiver is not acknowledging frames quickly enough. If this maximum is one packet the protocol is called **stop-and-wait**. Larger windows enable pipelining and improve performance on long, fast links.
- The **sliding window** protocol combines these features and is also used to support bidirectional data transfer.

The link layer checksum protects a frame while it crosses a single link.
The transport layer checksum protects a segment while it crosses an entire network path.

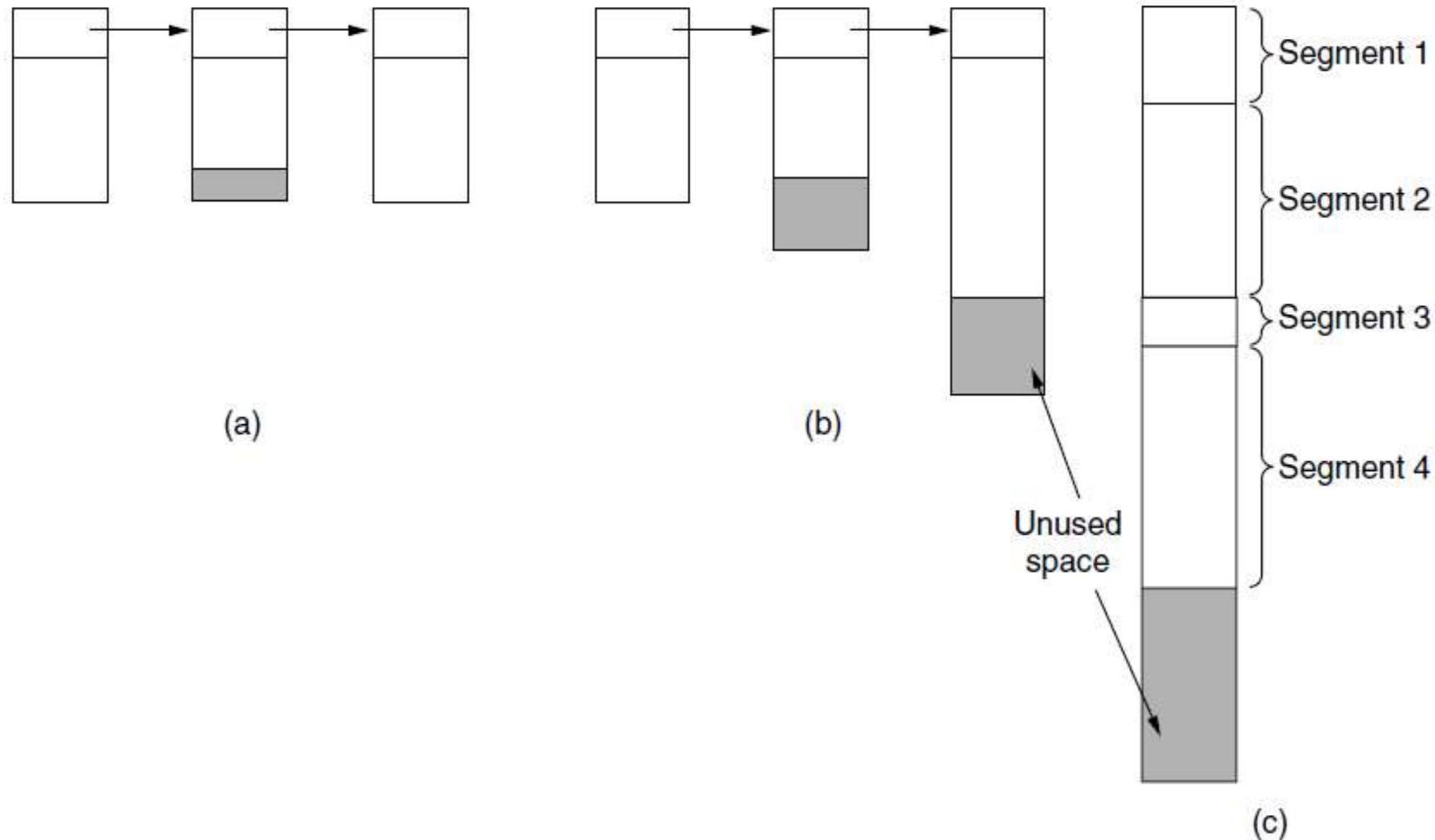


Figure 6-15. (a) Chained fixed-size buffers. (b) Chained variable-sized buffers.
(c) One large circular buffer per connection.

Multiplexing

Multiplexing, or sharing several conversations over connections, virtual circuits, and physical links plays a role in several layers of the network architecture. In the transport layer, the need for multiplexing can arise in a number of ways. For example, if only one network address is available on a host, all transport connections on that machine have to use it. When a segment comes in, some way is needed to tell which process to give it to. This situation, called **multiplexing**.

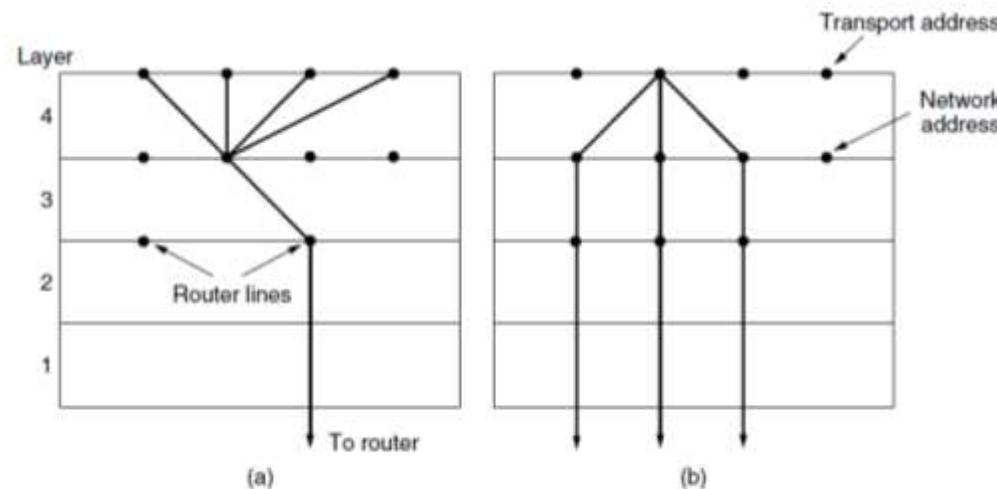


Figure 6-17. (a) Multiplexing. (b) Inverse multiplexing.

CONGESTION CONTROL

If the transport entities on many machines send too many packets into the network too quickly, the network will become congested, with performance degraded as packets are delayed and lost. Controlling congestion to avoid this problem is the combined responsibility of the network and transport layers. Congestion occurs at routers, so it is detected at the network layer.

Desirable Bandwidth Allocation

The goal is more than to simply avoid congestion. It is to find a good allocation of bandwidth to the transport entities that are using the network. A good allocation will deliver good performance because it uses all the available bandwidth but avoids congestion, it will be fair across competing transport entities, and it will quickly track changes in traffic demands.

Efficiency and Power

An efficient allocation of bandwidth across transport entities will use all of the network capacity that is available. However, it is not quite right to think that if there is a 100-Mbps link, five transport entities should get 20 Mbps each. They should usually get less than 20 Mbps for good performance. The reason is that the traffic is often bursty.

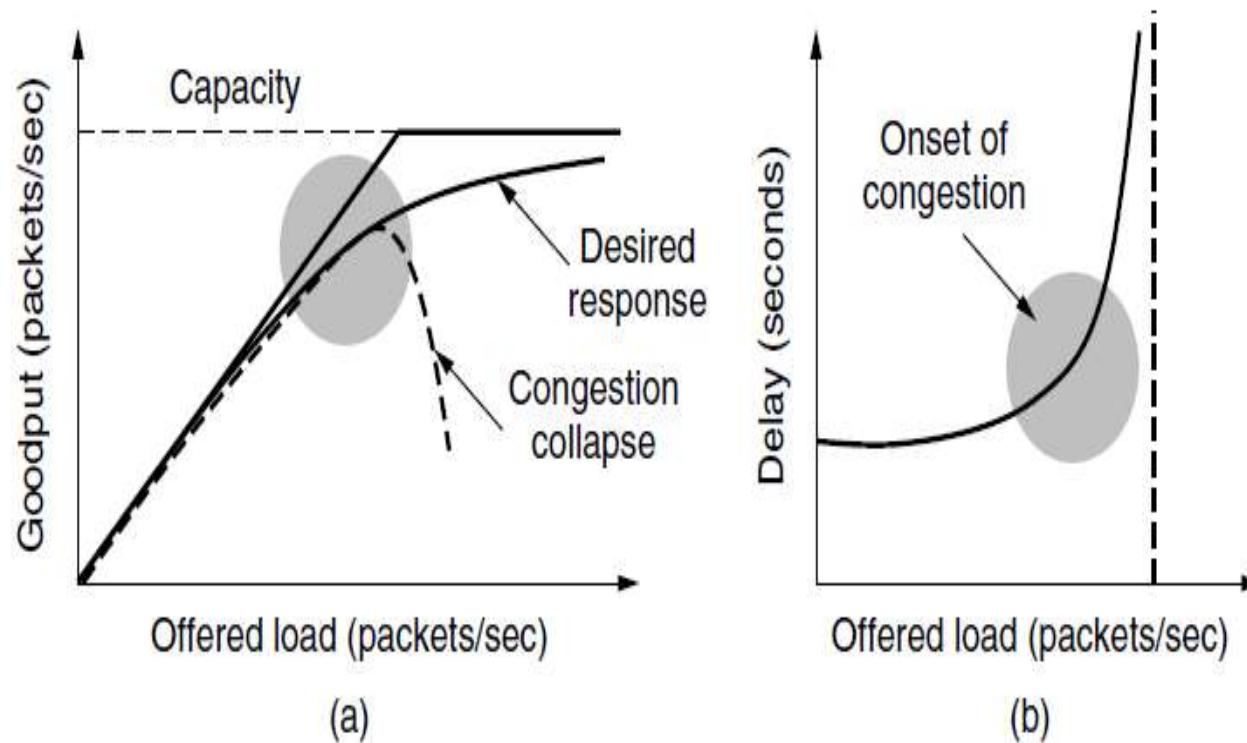


Figure 6-19. (a) Goodput and (b) delay as a function of offered load.

Max-Min Fairness

The form of fairness that is often desired for network usage is **max-min fairness**. An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation that is no larger. That is, increasing the bandwidth of a flow will only make the situation worse for flows that are less well off.

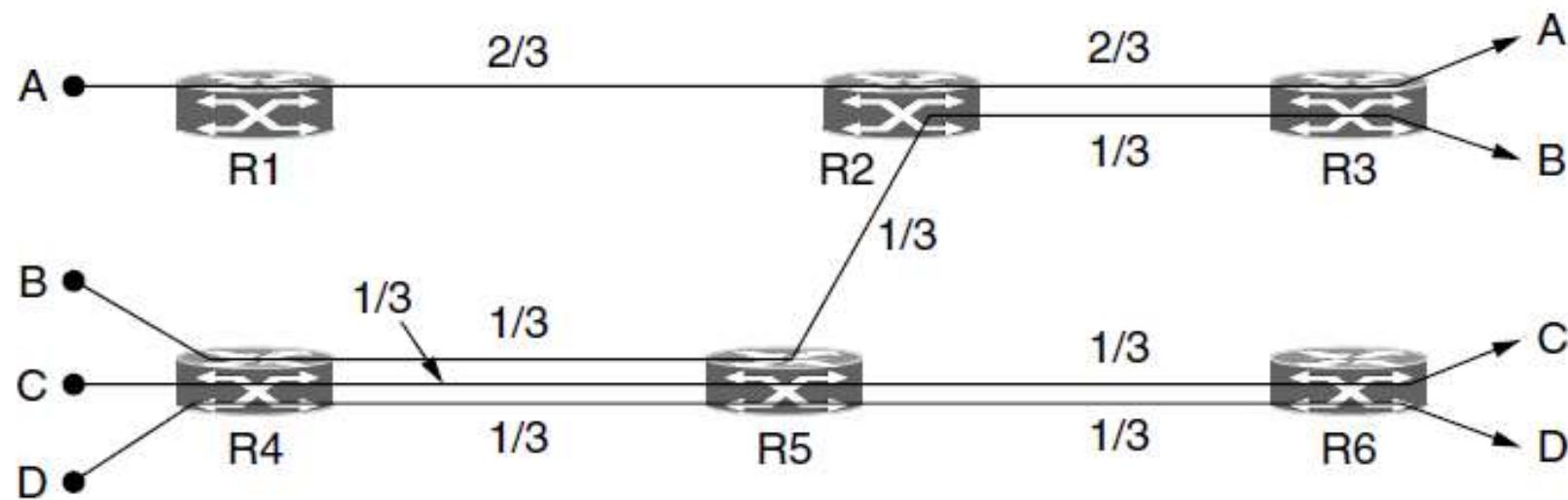


Figure 6-20. Max-min bandwidth allocation for four flows.

Let us see an example. A max-min fair allocation is shown for a network with four flows, A , B , C , and D , in Fig. 6-20. Each of the links between routers has the same capacity, taken to be 1 unit, though in the general case the links will have different capacities. Three flows compete for the bottom-left link between routers $R4$ and $R5$. Each of these flows therefore gets $1/3$ of the link. The remaining flow, A , competes with B on the link from $R2$ to $R3$. Since B has an allocation of $1/3$, A gets the remaining $2/3$ of the link. Notice that all of the other links have spare capacity. However, this capacity cannot be given to any of the flows without

decreasing the capacity of another, lower flow. For example, if more of the bandwidth on the link between $R2$ and $R3$ is given to flow B , there will be less for flow A .

Convergence

A final criterion is that the congestion control algorithm converge quickly to a fair and efficient allocation of bandwidth. Because of the variation in demand, the ideal operating point for the network varies over time. A good congestion control algorithm should rapidly converge to the ideal operating point, and it should track that point as it changes over time. If

the convergence is too slow, the algorithm will never be close to the changing operating point. If the algorithm is not stable, it may fail to converge to the right point in some cases, or even oscillate around the right point.

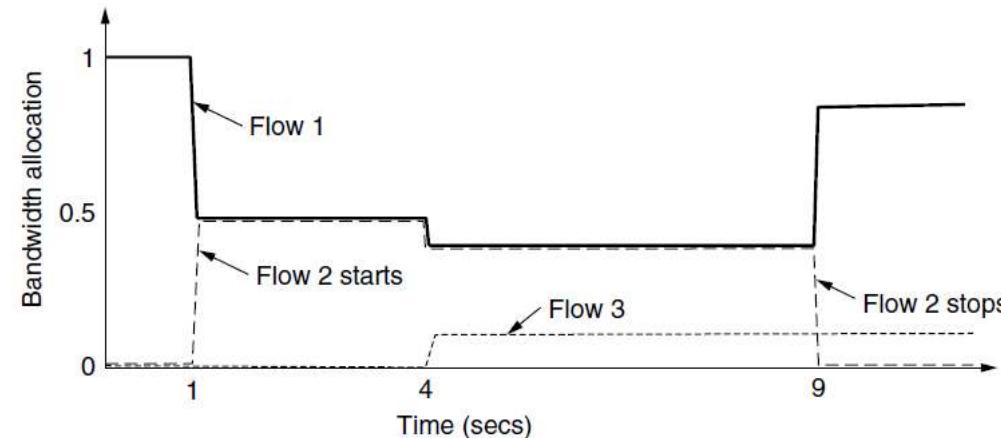


Figure 6-21. Changing bandwidth allocation over time.

Regulating the Sending Rate

The sending rate may be limited by two factors. The first is flow control, in the case that there is insufficient buffering at the receiver. The second is congestion, in the case that there is insufficient capacity in the network. In Fig. 6-22, we see this problem illustrated hydraulically. In Fig. 6-22(a), we see a thick pipe leading to a small-capacity receiver. This is a flow-control limited situation. As long as the sender does not send more water than the bucket can contain, no water will be lost. In Fig. 6-22(b), the limiting factor is not the bucket capacity, but the internal carrying capacity of the network. If too much water comes in too fast, it will back up and some will be lost (in this case, by overflowing the funnel).

If an explicit and precise signal is given, the transport entity can use that signal to adjust its rate to the new operating point. For example, if XCP tells senders the rate to use, the senders may simply use that rate. In the other cases, however, some guesswork is involved. In the absence of a congestion signal, the senders should decrease their rates. When a congestion signal is given, the senders should decrease their rates. The way in which the rates are increased or decreased is given by a **control law**.

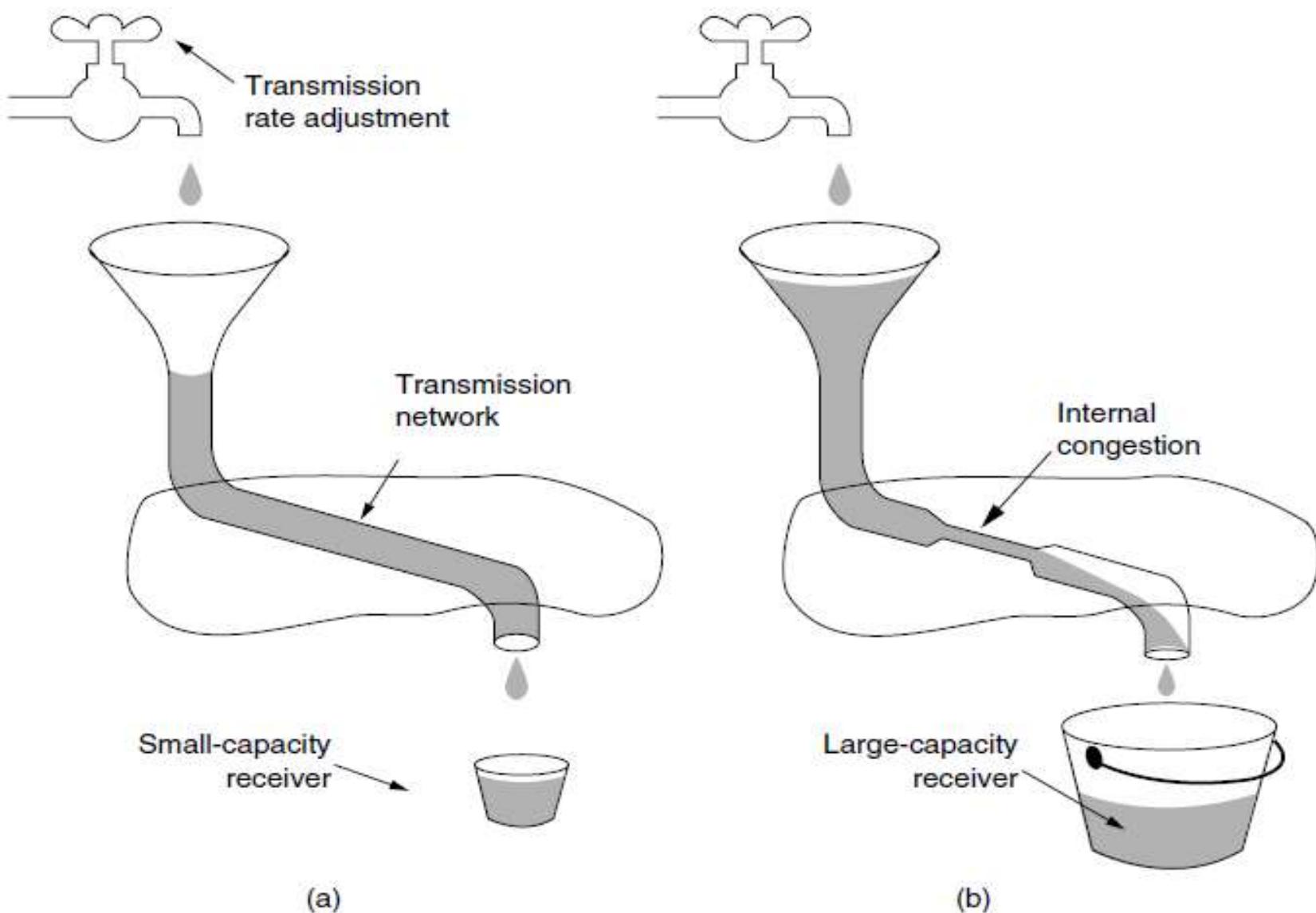


Figure 6-22. (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver.

Wireless Issues

Transport protocols such as TCP that implement congestion control should be independent of the underlying network and link layer technologies. That is a good theory, but in practice there are issues with wireless networks. The main issue is that packet loss is often used as a congestion signal, including by TCP. Wireless networks lose packets all the time due to transmission errors.

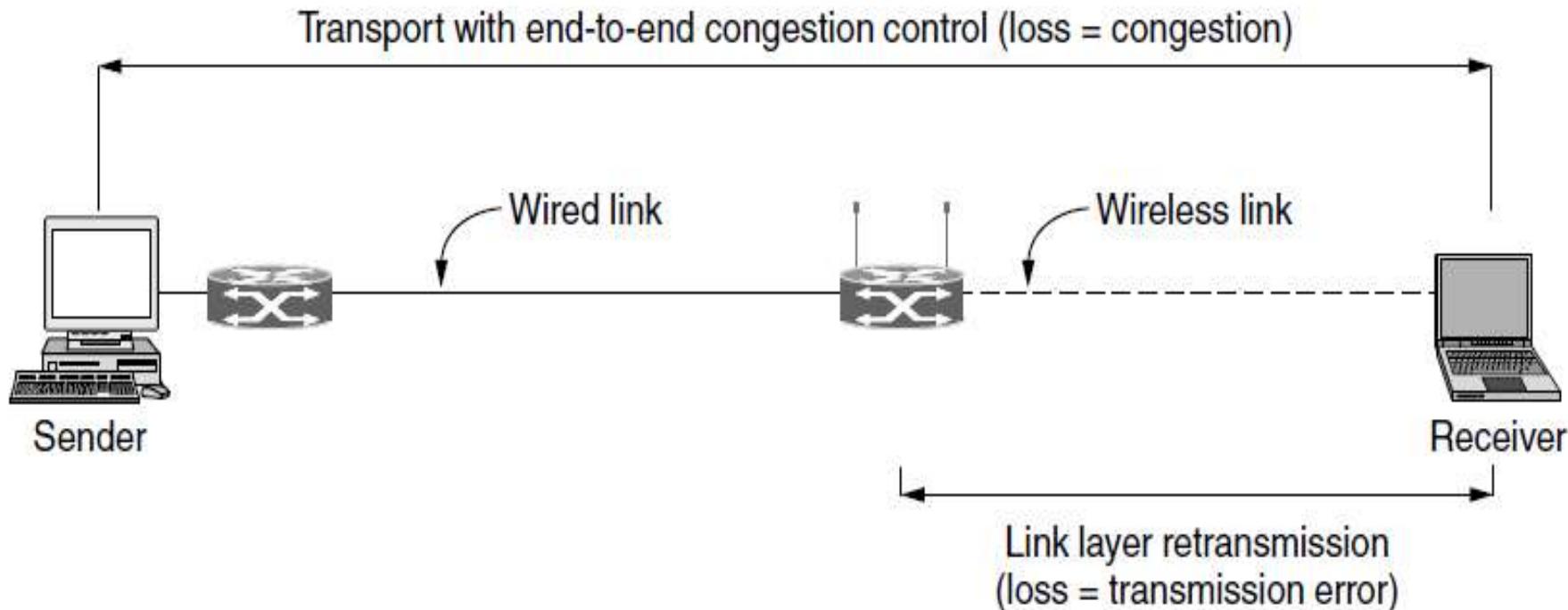


Figure 6-26. Congestion control over a path with a wireless link.

THE INTERNET TRANSPORT PROTOCOLS: UDP

Introduction to UDP

The Internet protocol suite supports a connectionless transport protocol called **UDP (User Datagram Protocol)**. UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection.

UDP transmits **segments** consisting of an 8-byte header followed by the payload. The header contains two **ports** serve to identify the endpoints within the source and destination machines. When a UDP packet arrives, its payload is handed to the process attached to the destination port.

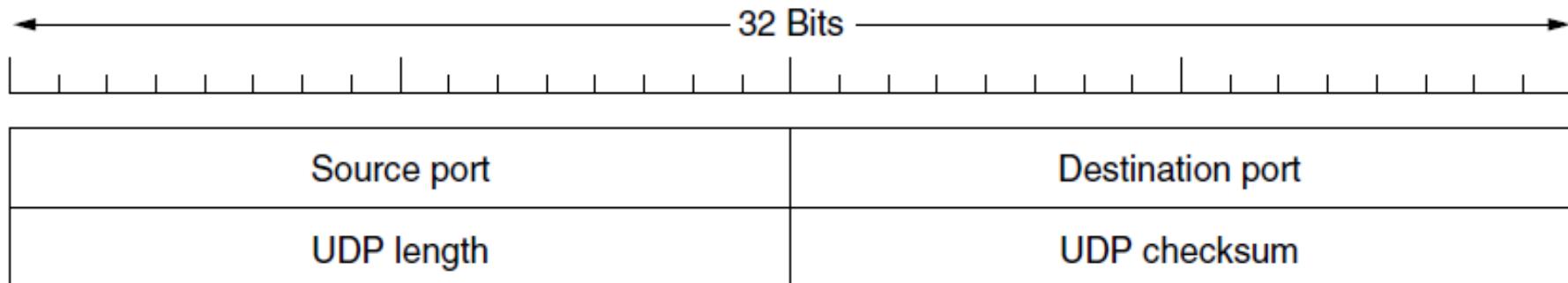


Figure 6-27. The UDP header.

Remote Procedure Call

a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called procedure takes place on 2. Information can be transported from the caller to the callee in the parameters and can come back in the procedure result. No message passing is visible to the application programmer. This technique is known as **RPC (Remote Procedure Call)** and has become the basis for many networking applications.

to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub**, that represents the server procedure in the client's address space. Similarly, the server is bound with a procedure called the **server stub**.

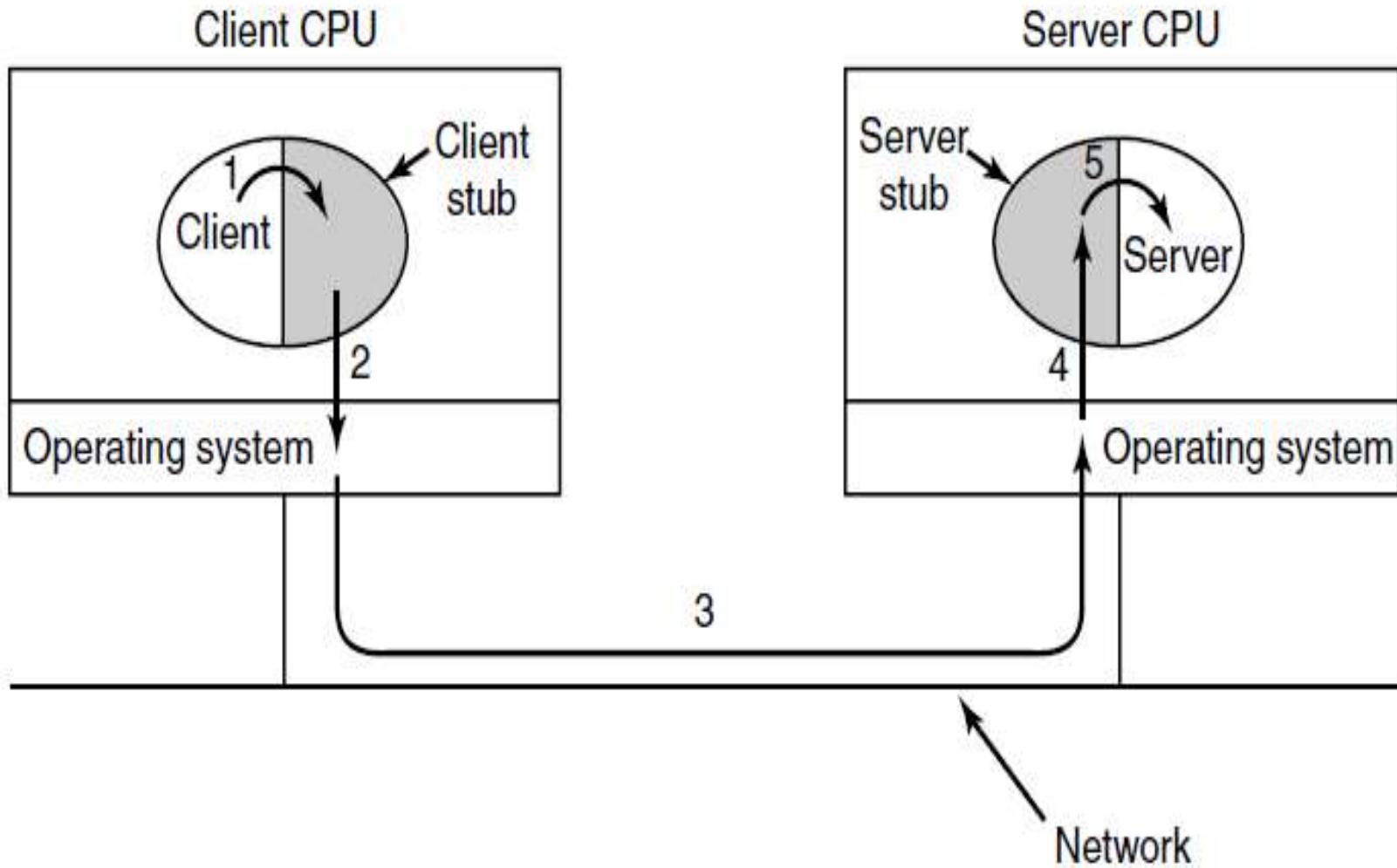


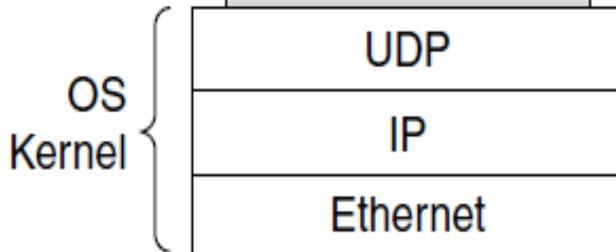
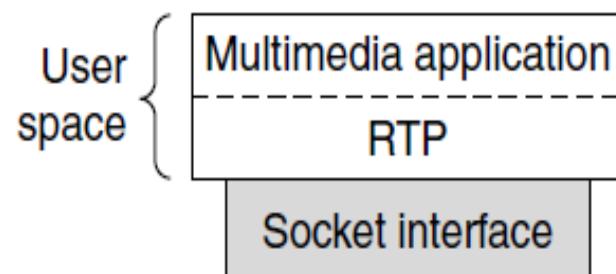
Figure 6-29. Steps in making a remote procedure call. The stubs are shaded.

Real-Time Transport Protocols

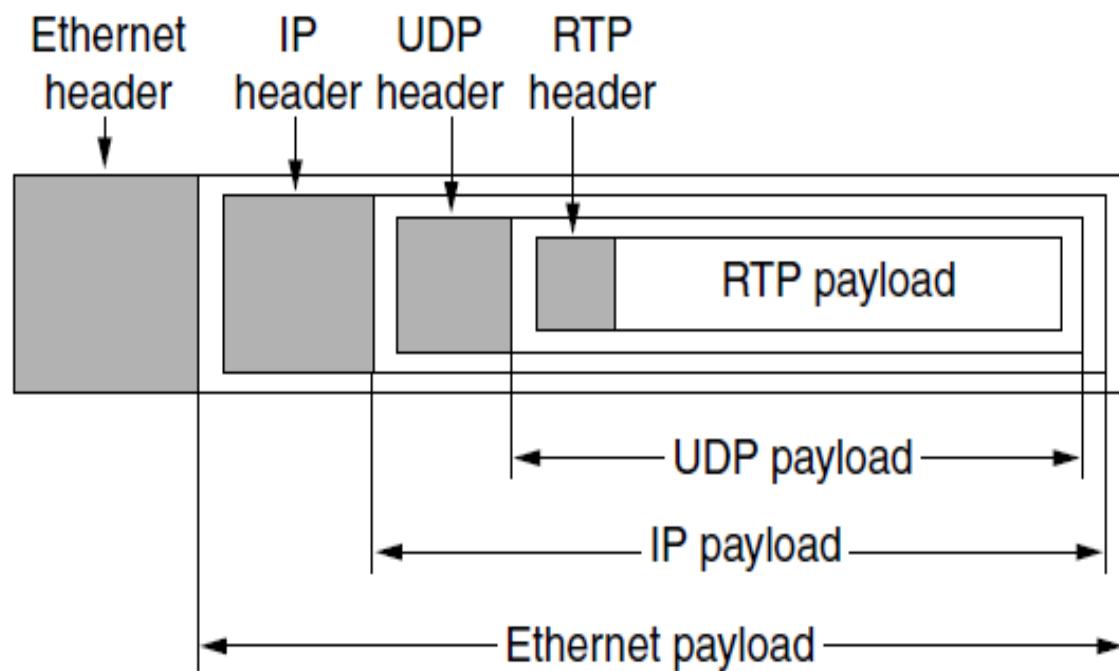
Client-server RPC is one area in which UDP is widely used. Another one is for real-time multimedia applications. In particular, as Internet radio, Internet telephony, music-on-demand, videoconferencing, video-on-demand, and other multimedia applications became more commonplace, people have discovered that each application was reinventing more or less the same real-time transport protocol.

Two aspects of real-time transport are the first is the RTP protocol for transporting audio and video data in packets. The second is the processing that takes place, mostly at the receiver, to play out the audio and video at the right time.

RTP normally runs in user space over UDP (in the operating system). It operates as follows. The multimedia application consists of multiple audio, video, text, and possibly other streams.



(a)



(b)

Figure 6-30. (a) The position of RTP in the protocol stack. (b) Packet nesting.

RTP—The Real-time Transport Protocol

The basic function of RTP is to multiplex several real-time data streams onto a single stream of UDP packets. The UDP stream can be sent to a single destination (unicasting) or to multiple destinations (multicasting). Because RTP just uses normal UDP, its packets are not treated specially by the routers unless some normal IP quality-of-service features are enabled.

Another facility many real-time applications need is timestamping. The idea here is to allow the source to associate a timestamp with the first sample in each packet. The timestamps are relative to the start of the stream, so only the differences between timestamps are significant.

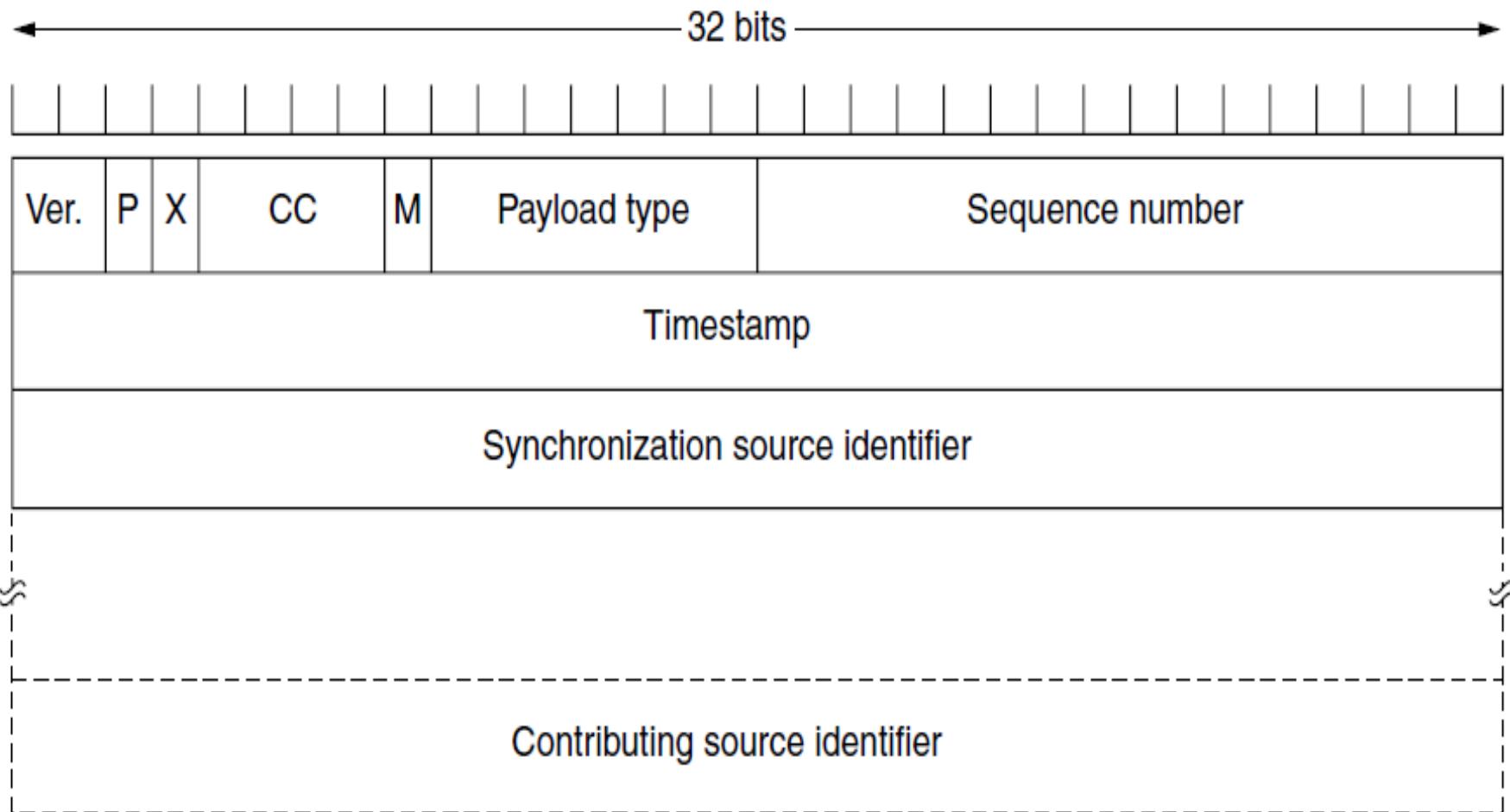


Figure 6-31. The RTP header.

The P bit indicates that the packet has been padded to a multiple of 4 bytes.

The X bit indicates that an extension header is present.

The CC field tells how many contributing sources are present, from 0 to 15 (see below). The M bit is an application-specific marker bit.

The *Payload type* field tells which encoding algorithm has been used (e.g., uncompressed 8-bit audio, MP3, etc.).

Since every packet carries this field, the encoding can change during transmission.

The *Sequence number* is just a counter that is incremented on each RTP packet sent. It is used to detect lost packets.

The *Timestamp* is produced by the stream's source to note when the first sample in the packet was made.

The *Synchronization source identifier* tells which stream the packet belongs to. It is the method used to multiplex and demultiplex multiple data streams onto a single stream of UDP packets. Finally, the *Contributing source identifiers*, if any, are used when mixers are present in the studio.

RTCP—The Real-time Transport Control Protocol

RTP has a little sister protocol (little sibling protocol?) called **RTCP (Realtime Transport Control Protocol)**.

An issue with providing feedback is that the RTCP reports are sent to all participants. For a multicast application with a large group, the bandwidth used by RTCP would quickly grow large. To prevent this from happening, RTCP senders scale down the rate of their reports to collectively consume no more than, say, 5% of the media bandwidth. RTCP also handles interstream synchronization.

Playout with Buffering and Jitter Control

if the packets are injected with exactly the right intervals between them at the sender, they will reach the receiver with different relative times. This variation in delay is called **jitter**. Even a small amount of packet jitter can cause distracting media artifacts, such as jerky video frames and unintelligible audio, if the media is simply played out as it arrives.

The **solution to this problem** is to **buffer** packets at the receiver before they are played out to reduce the jitter.

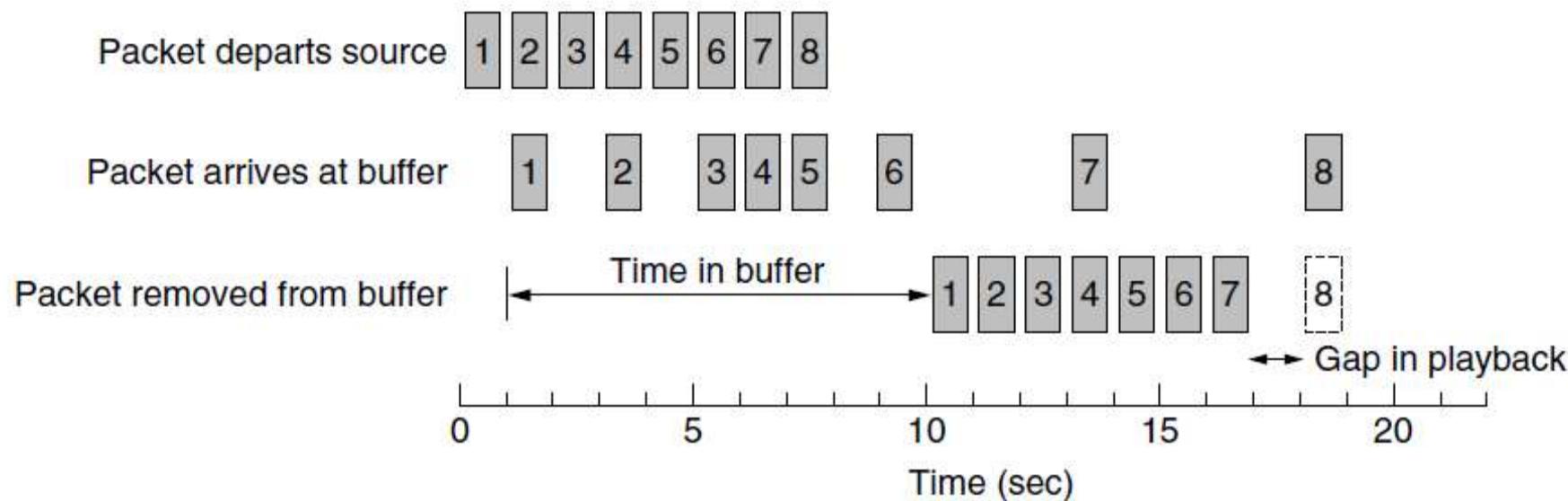


Figure 6-32. Smoothing the output stream by buffering packets.

A key consideration for smooth playout is the **playback point**, or how long to wait at the receiver for media before playing it out. Deciding how long to wait depends on the jitter.

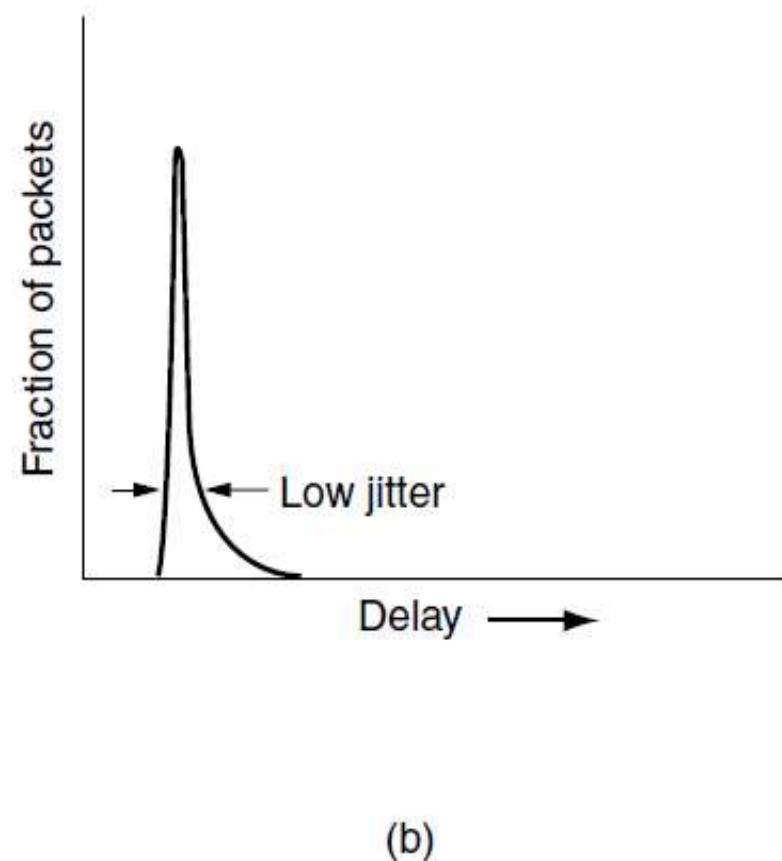
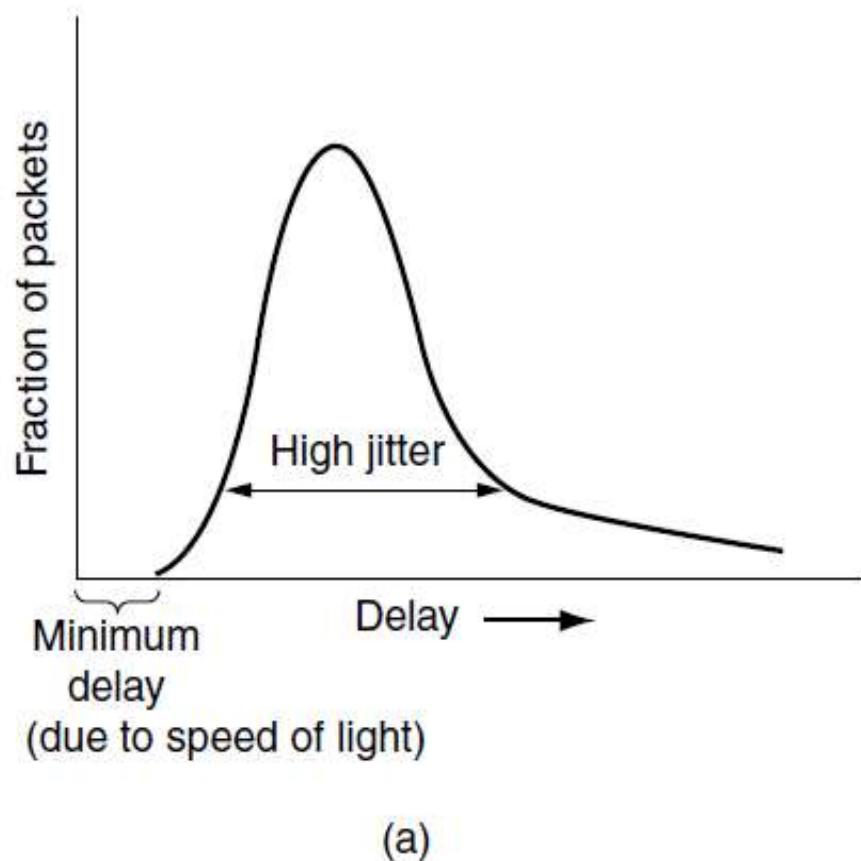


Figure 6-33. (a) High jitter. (b) Low jitter.

THE INTERNET TRANSPORT PROTOCOLS: TCP

Introduction to TCP

TCP (Transmission Control Protocol) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters.

The TCP Service Model

TCP service is obtained by both the sender and the receiver creating end points, called **sockets**

Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port**. A port is the TCP name for a TSAP. For TCP service to be obtained, a connection must be explicitly established between a socket on one machine and a socket on another machine.

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Figure 6-34. Some assigned ports.

The TCP Protocol

A key feature of TCP, and one that dominates the protocol design, is that every byte on a TCP connection has its own 32-bit sequence number.

The sending and receiving TCP entities exchange data in the form of segments. A **TCP segment** consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes.

Two limits restrict the segment size. First, each segment, including the TCP header, must fit in the 65,515- byte IP payload. Second, each link has an **MTU (Maximum Transfer Unit)**.

Each segment must fit in the MTU at the sender and receiver so that it can be sent and received in a single, unfragmented packet.

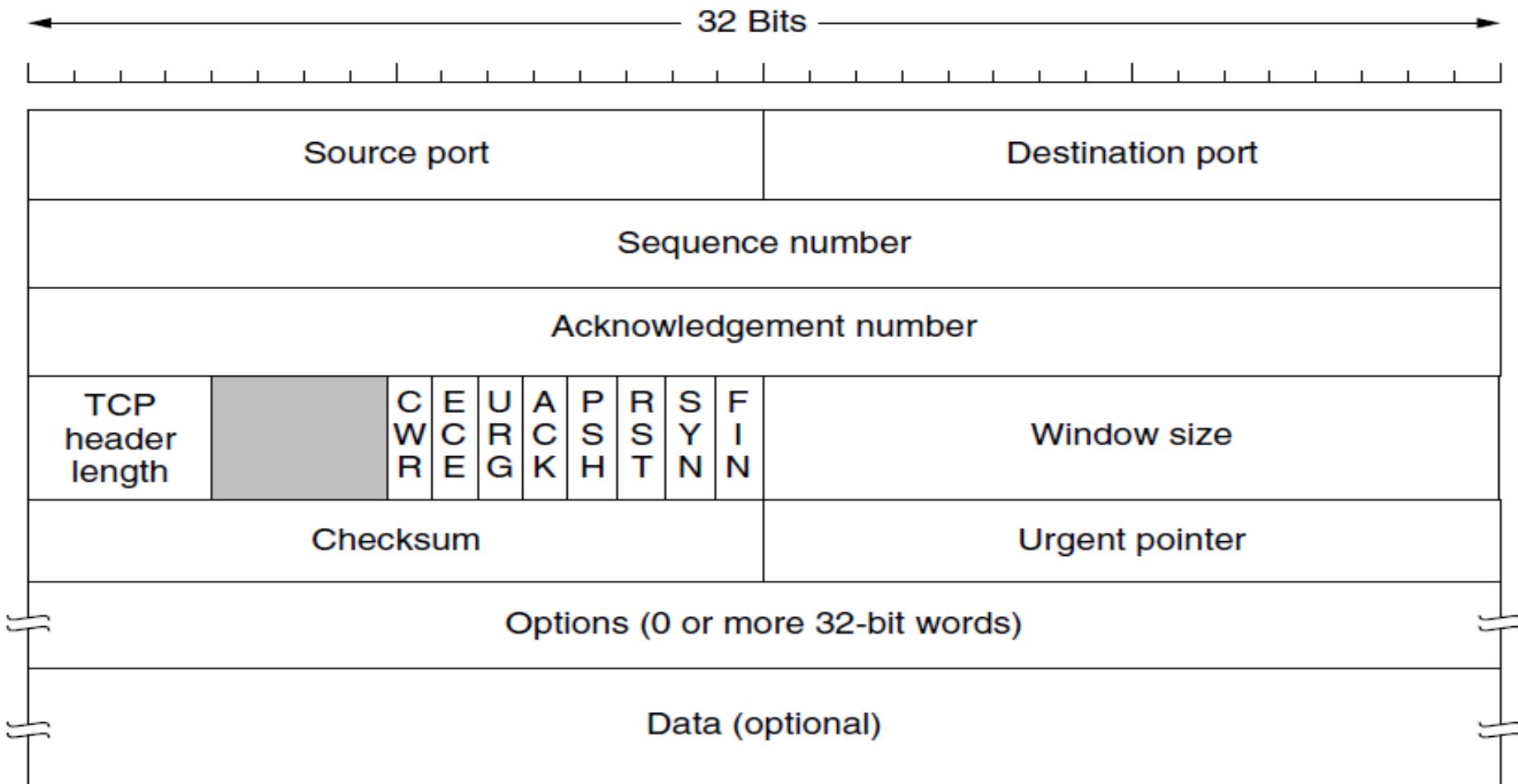


Figure 6-36. The TCP header.

Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options. After the options, if any, up to $65,535 - 20 - 20 = 65,495$ data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header. Segments without any data are legal and are commonly used for acknowledgements and control messages.

Let us dissect the TCP header field by field. The *Source port* and *Destination port* fields identify the local end points of the connection. A TCP port plus its host's IP address forms a 48-bit unique end point.

The *Sequence number* and *Acknowledgement number* fields perform their usual functions.

The *TCP header length* tells how many 32-bit words are contained in the TCP header. This information is needed because the *Options* field is of variable length, so the header is, too.

eight 1-bit flags. *CWR* and *ECE* are used to signal congestion when ECN (Explicit Congestion Notification) is used, as specified in RFC 3168. *ECE* is set to signal an *ECN-Echo* to a TCP sender to tell it to slow down when the TCP receiver gets a congestion indication from the network.

CWR is set to signal *Congestion Window Reduced* from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the *ECN-Echo*.

URG is set to 1 if the *Urgent pointer* is in use. The *Urgent pointer* is used to indicate a byte offset from the current sequence number at which urgent data are to be found.

The *ACK* bit is set to 1 to indicate that the *Acknowledgement number* is valid. This is the case for nearly all packets. If *ACK* is 0, the segment does not contain an acknowledgement, so the *Acknowledgement number* field is ignored.

The *PSH* bit indicates PUSHed data.

The *RST* bit is used to abruptly reset a connection that has become confused due to a host crash or some other reason.

The *SYN* bit is used to establish connections.

The *FIN* bit is used to release a connection

A *Checksum* is also provided for extra reliability. It checksums the header, the data, and a conceptual pseudoheader in exactly the same way as UDP, except that the pseudoheader has the protocol number for TCP (6) and the checksum is mandatory.

The *Options* field provides a way to add extra facilities not covered by the regular header.

TCP Connection Establishment

Connections are established in TCP by means of the three-way handshake

To establish a connection, one side, say, the server, passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives in that order, either specifying a specific source or nobody in particular

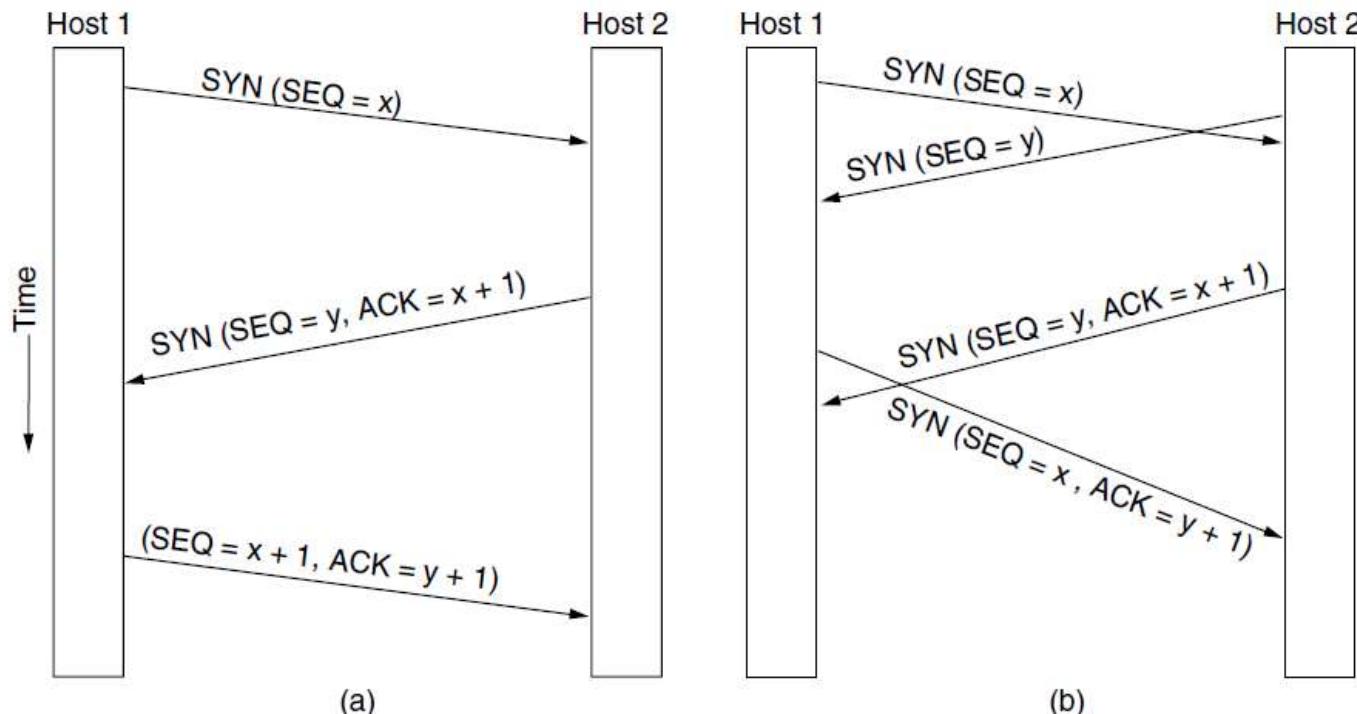


Figure 6-37. (a) TCP connection establishment in the normal case. (b) Simultaneous connection establishment on both sides.

TCP Connection Release

To release a connection, either party can send a TCP segment with the *FIN* bit set, which means that it has no more data to transmit. When the *FIN* is acknowledged, that direction is shut down for new data.

Normally, four TCP segments are needed to release a connection: one *FIN* and one *ACK* for each direction. However, it is possible for the first *ACK* and the second *FIN* to be contained in the same segment, reducing the total count to three.

TCP Connection Management Modeling

The steps required to establish and release connections can be represented in a finite state machine with the 11 states.

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

Figure 6-38. The states used in the TCP connection management finite state machine.

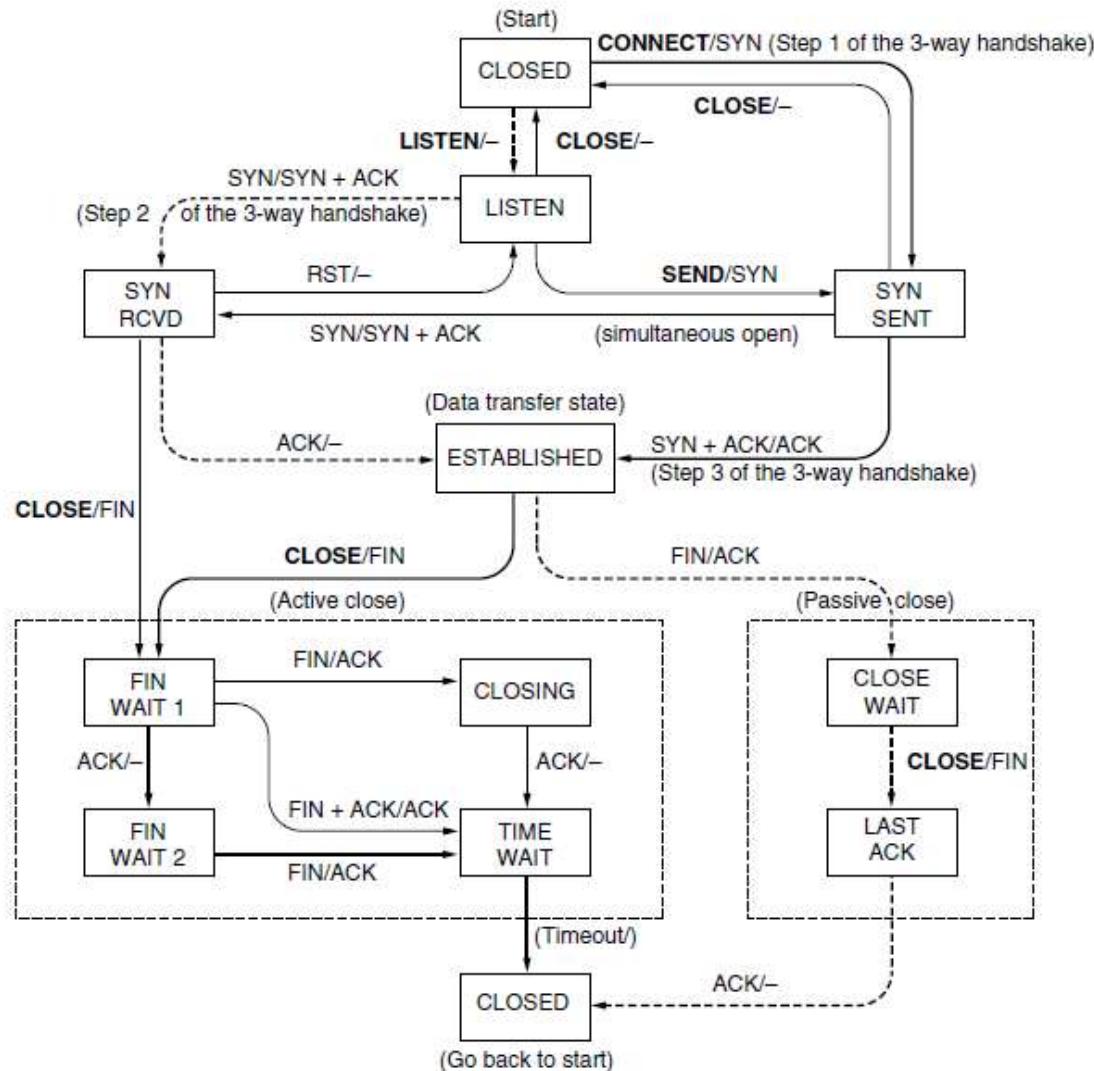


Figure 6-39. TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled with the event causing it and the action resulting from it, separated by a slash.

TCP Sliding Window

Window management in TCP decouples the issues of acknowledgement of the correct receipt of segments and receiver buffer allocation. For example, suppose the receiver has a 4096-byte buffer, as shown in Fig. 6-40. If the sender transmits a 2048-byte segment that is correctly received, the receiver will acknowledge the segment. However, since it now has only 2048 bytes of buffer space (until the application removes some data from the buffer), it will advertise a window of 2048 starting at the next byte expected.

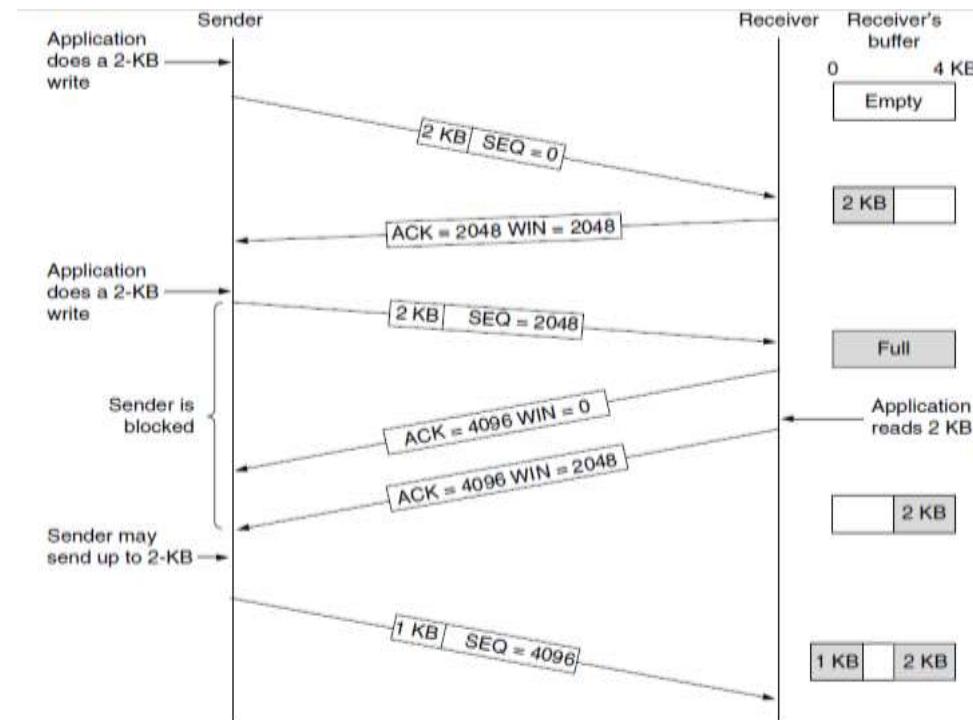


Figure 6-40. Window management in TCP.

TCP Timer Management

TCP uses multiple timers (at least conceptually) to do its work. The most important of these is the RTO (Retransmission TimeOut). When a segment is sent, a retransmission timer is started. If the segment is acknowledged before the timer expires, the timer is stopped. If, on the other hand, the timer goes off before the acknowledgement comes in, the segment is retransmitted.

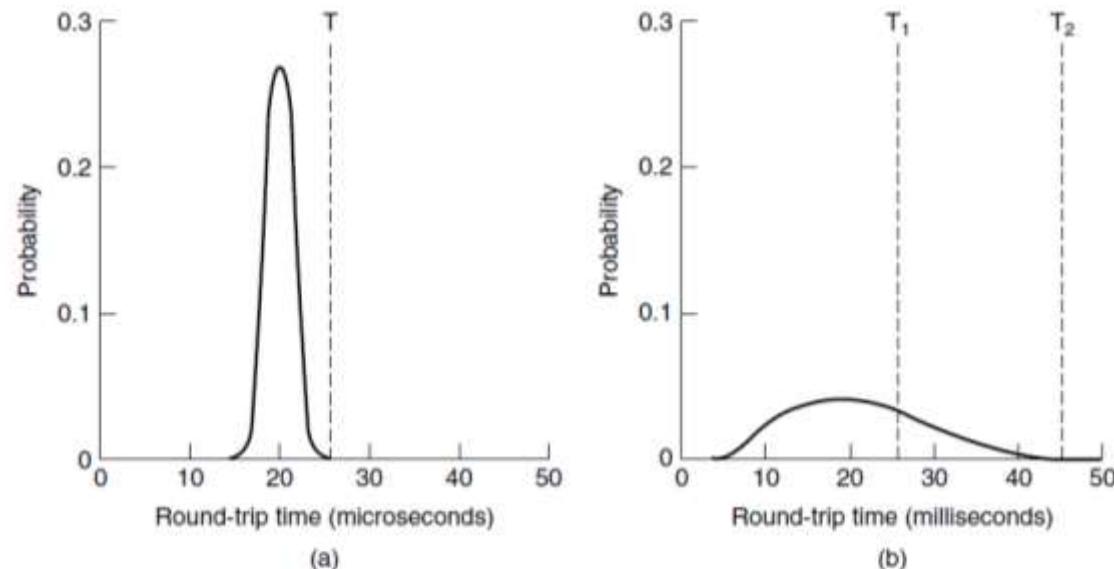


Figure 6-42. (a) Probability density of acknowledgement arrival times in the data link layer. (b) Probability density of acknowledgement arrival times for TCP.

TCP Congestion Control

When the load offered to any network is more than it can handle, congestion builds up. The Internet is no exception.

TCP maintains a **congestion window** whose size is the number of bytes the sender may have in the network at any time. The corresponding rate is the window size divided by the round-trip time of the connection. TCP adjusts the size of the window according to the AIMD(Additive Increase Multiplicative Decrease) rule.

All the Internet TCP algorithms assume that lost packets are caused by congestion and monitor timeouts and look for signs of trouble the way miners watch their canaries.

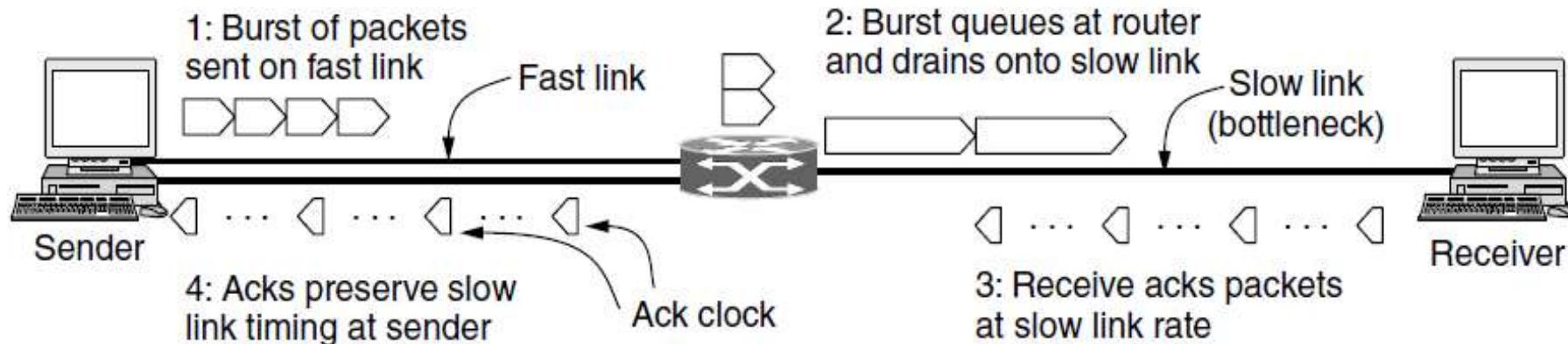


Figure 6-43. A burst of packets from a sender and the returning ack clock.

Fig. 6-44. When the sender gets an acknowledgement, it increases the congestion window by one and immediately sends two packets into the network. (One packet is the increase by one; the other packet is a replacement for the packet that has been acknowledged and left the network.

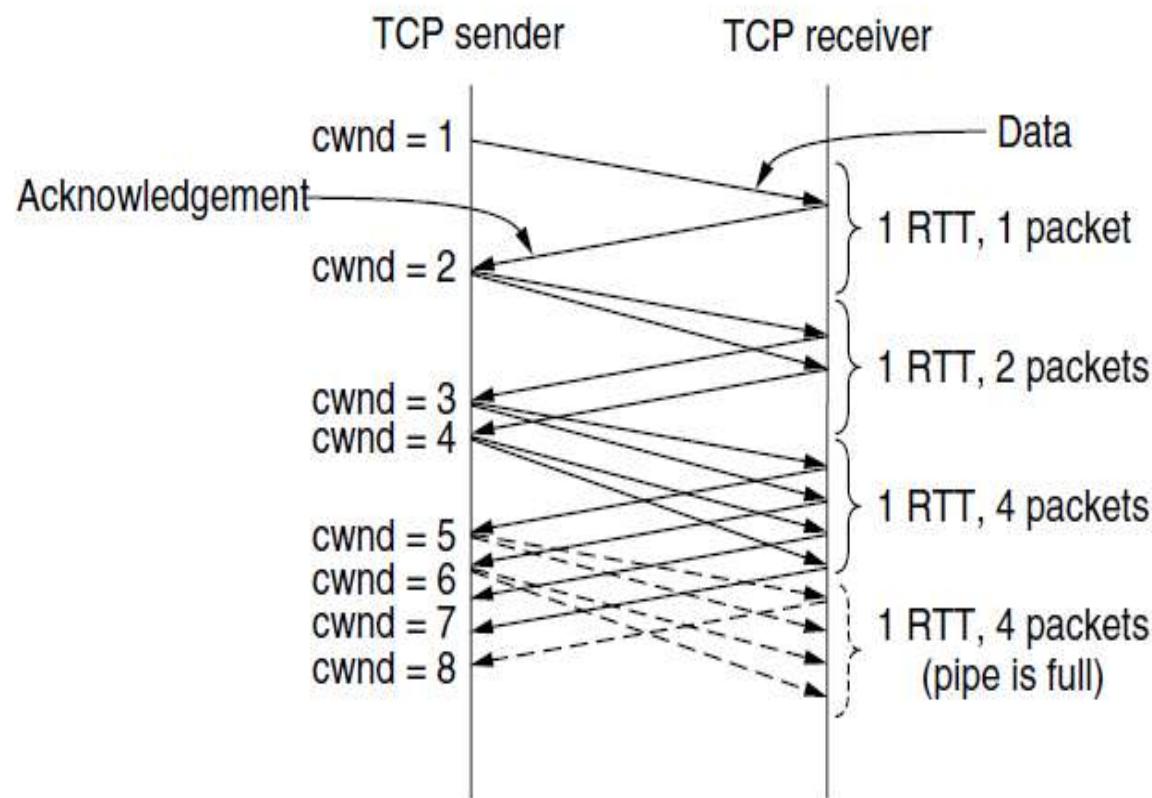


Figure 6-44. Slow start from an initial congestion window of one segment.

Additive increase is shown in Fig. 6-45 for the same situation as slow start. At the end of every RTT, the sender's congestion window has grown enough that it can inject an additional packet into the network. Compared to slow start, the linear rate of growth is much slower.

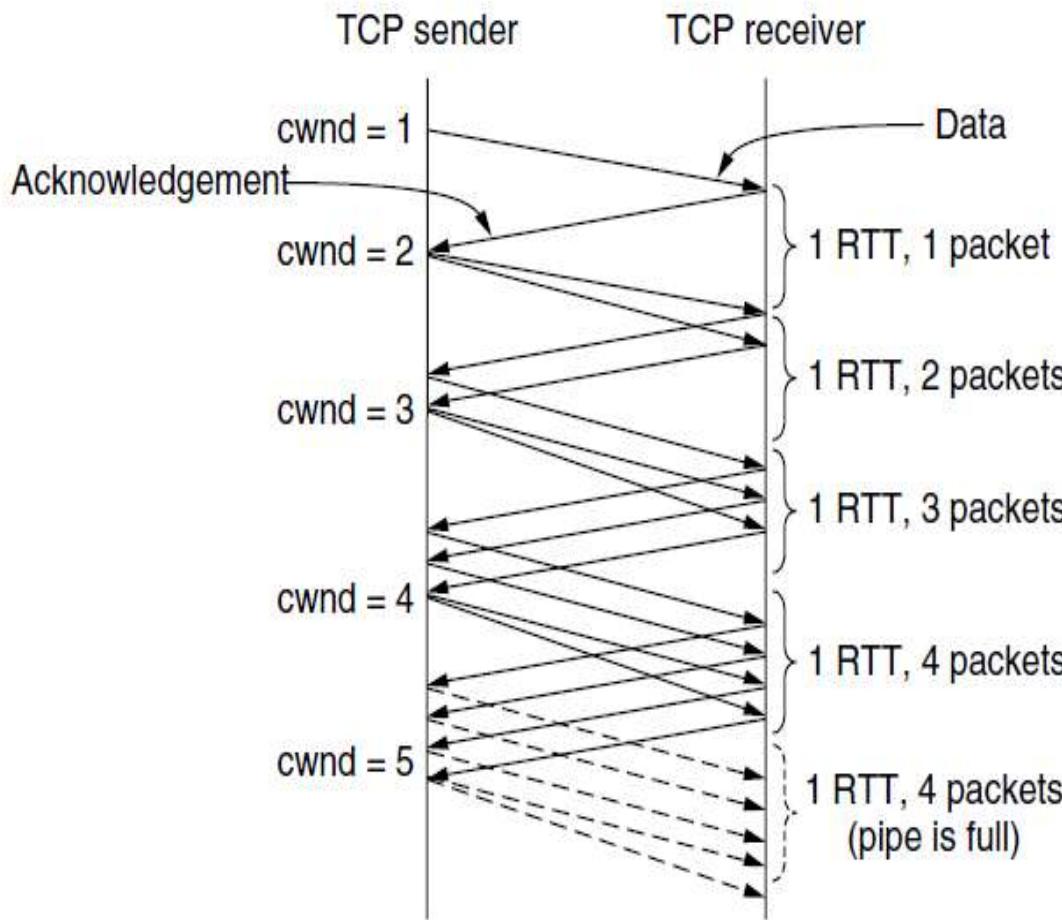


Figure 6-45. Additive increase from an initial congestion window of one segment.

PERFORMANCE ISSUES

Performance issues are very important in computer networks. When hundreds or thousands of computers are interconnected, complex interactions, with unforeseen consequences, are common.

The following are the network performance issues

1. Performance problems.
2. Measuring network performance.
3. Host design for fast networks.
4. Fast segment processing.
5. Header compression.
6. Protocols for “long fat” networks.

Performance Problems in Computer Networks

Performance also degrades when there is a structural resource imbalance. For example, if a gigabit communication line is attached to a low-end PC, the poor host will not be able to process the incoming packets fast enough and some will be lost. These packets will eventually be retransmitted, adding delay, wasting bandwidth, and generally reducing performance.

A second example of synchronous overload is what happens after an electrical power failure. When the power comes back on, all the machines simultaneously start rebooting. A typical reboot sequence might require first going to some (DHCP) server to learn one's true identity, and then to some file server to get a copy of the operating system.

Network Performance Measurement

Measurements can be made in different ways and at many locations (both in the protocol stack and physically). The most basic kind of measurement is to start a timer when beginning some activity and see how long that activity takes. For example, knowing how long it takes for a segment to be acknowledged is a key measurement. Other measurements are made with counters that record how often some event has happened.

- Make Sure That the Sample Size Is Large Enough**

Do not measure the time to send one segment, but repeat the measurement, say, one million times and take the average. Startup effects, such as the 802.16 NIC or cable modem getting a bandwidth reservation after an idle period, can slow the first segment, and queueing introduces variability.

- Make Sure That the Samples Are Representative**

Measurements of congestion, for example, are of little use if they are made at a moment when there is no congestion. Sometimes the results may be counterintuitive at first, such as heavy congestion at 11 A.M., and 1 P.M., but no congestion at noon (when all the users are at lunch).

- Caching Can Wreak Havoc with Measurements**

Repeating a measurement many times will return an unexpectedly fast answer if the protocols use caching mechanisms. For instance, fetching a Web page or looking up a DNS name (to find the IP address) may involve a network exchange the first time, and then return the answer from a local cache without sending any packets over the network.

- Be Sure That Nothing Unexpected Is Going On during Your Tests**

Making measurements at the same time that some user has decided to run a video conference over your network will often give different results than if there is no video conference.

- **Be Careful When Using a Coarse-Grained Clock**

Computer clocks function by incrementing some counter at regular intervals. For example, a millisecond timer adds 1 to a counter every 1 msec. Using such a timer to measure an event that takes less than 1 msec is possible but requires some care. Some computers have more accurate clocks, of course, but there are always shorter events to measure too.

- **Be Careful about Extrapolating the Results**

Suppose that you make measurements with simulated network loads running from 0 (idle) to 0.4 (40% of capacity). For example, the response time to send a voice-over-IP packet over an 802.11 network might be as shown by the data points and solid line through them.

HOST DESIGN FOR FAST NETWORKS

Measuring and tinkering can improve performance considerably, but they cannot substitute for good design in the first place.

First, NICs (Network Interface Cards) and routers have already been engineered (with hardware support) to run at “wire speed.” This means that they can process packets as quickly as the packets can possibly arrive on the link. Second, the relevant performance is that which applications obtain.

Host Speed Is More Important Than Network Speed

Long experience has shown that in nearly all fast networks, operating system and protocol overhead dominate actual time on the wire. For example, in theory, the minimum RPC time on a 1-Gbps Ethernet is 1 μ sec, corresponding to a minimum (512-byte) request followed by a minimum (512-byte) reply. In practice, overcoming the software overhead and getting the RPC time anywhere near there is a substantial achievement. It rarely happens in practice.

Reduce Packet Count to Reduce Overhead

Each segment has a certain amount of overhead (e.g., the header) as well as data (e.g., the payload). Bandwidth is required for both components. Processing is also required for both components (e.g., header processing and doing the checksum). When 1 million bytes are being sent, the data cost is the same no matter what the segment size is. However, using 128-byte segments means 32 times as much per-segment overhead as using 4-KB segments. The bandwidth and processing overheads add up fast to reduce throughput.

Minimize Context Switches

A related rule is that context switches (e.g., from kernel mode to user mode) are deadly. They have the bad properties of interrupts and copying combined. This cost is why transport protocols are often implemented in the kernel. Like reducing packet count, context switches can be reduced by having the library procedure that sends data do internal buffering until it has a substantial amount of them.

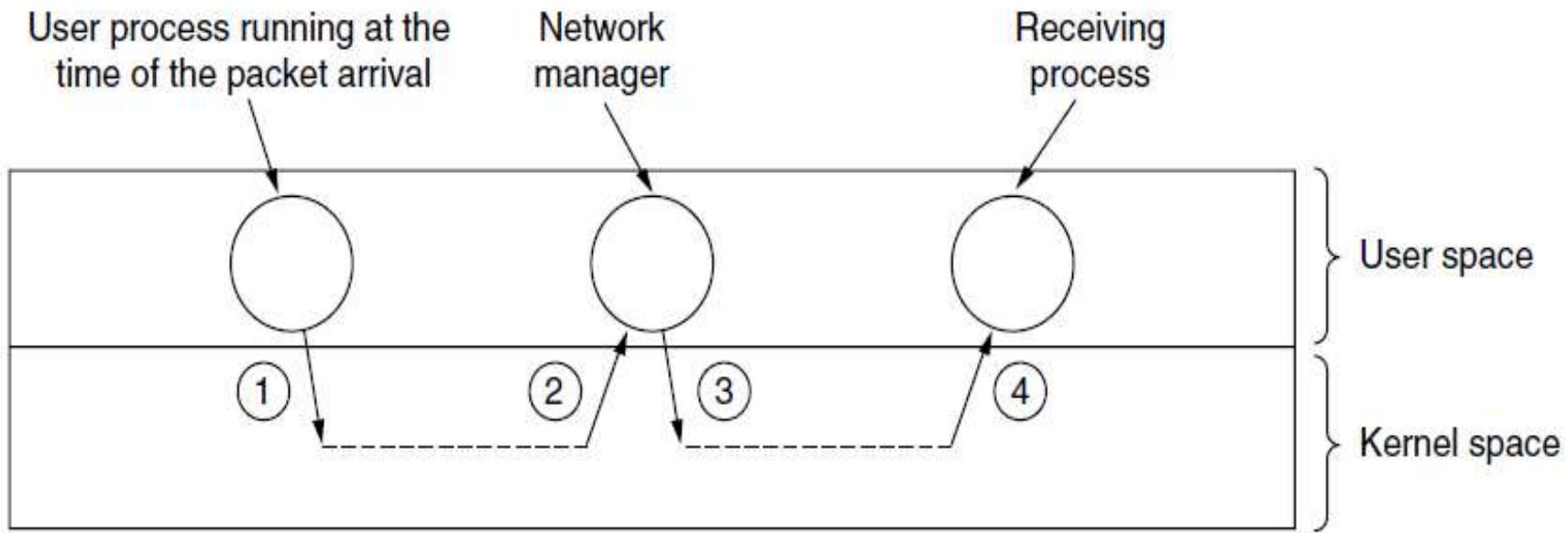


Figure 6-50. Four context switches to handle one packet with a user-space network manager.

FAST SEGMENT PROCESSING

Segment processing overhead has two components: overhead per segment and overhead per byte. Both must be attacked. The key to fast segment processing is to separate out the normal, successful case (one-way data transfer) and handle it specially. Many protocols tend to emphasize what to do when something goes wrong (e.g., a packet getting lost), but to make the protocols run fast, the designer should aim to minimize processing time when everything goes right. Minimizing processing time when an error occurs is secondary.

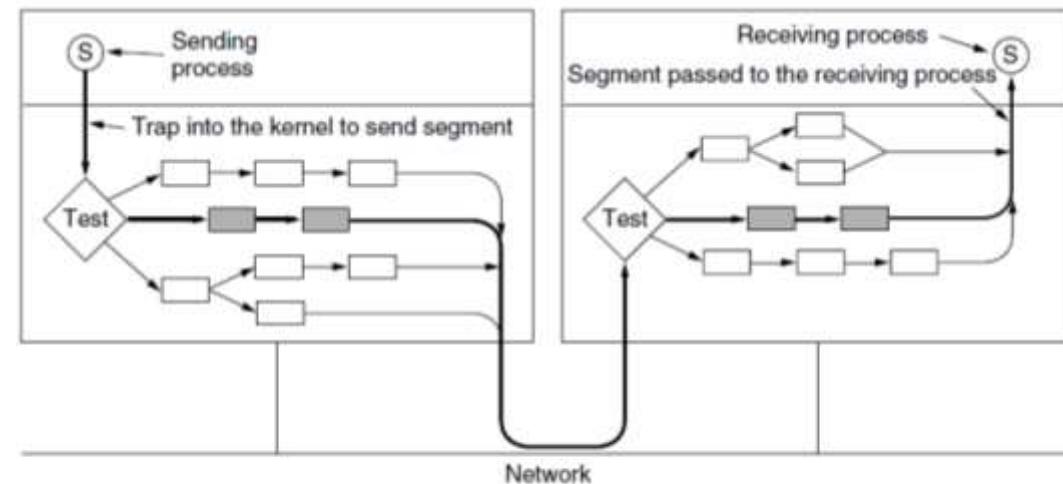


Figure 6-51. The fast path from sender to receiver is shown with a heavy line. The processing steps on this path are shaded.

HEADER COMPRESSION

To use bandwidth well, protocol headers and payloads should be carried with the minimum of bits. For payloads, this means using compact encodings of information, such as images that are in JPEG format rather than a bitmap, or document formats such as PDF that include compression. It also means application-level caching mechanisms, such as Web caches that reduce transfers in the first place.

Header compression is used to reduce the bandwidth taken over links by higher-layer protocol headers. Specially designed schemes are used instead of general purpose methods. This is because headers are short, so they do not compress well individually, and decompression requires all prior data to be received. This will not be the case if a packet is lost. Header compression can help by reducing the amount of data that is sent, and hence reducing transmission delay.

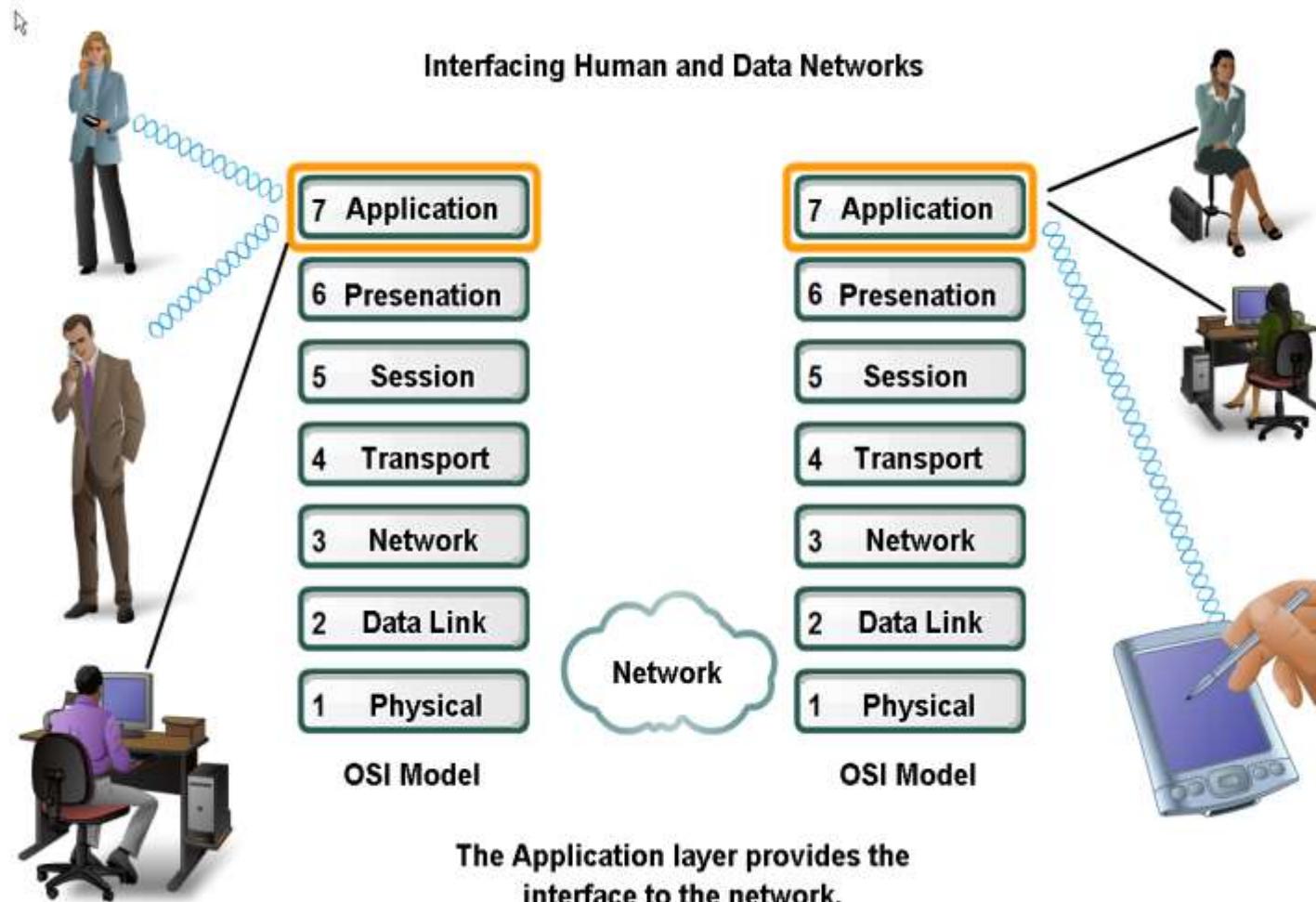
UNIT 5

INTRODUCTION TO APPLICATION LAYER

- Introduction
- WWW and HTTP
- FTP
- E-mail
- TELNET
- Secure Shell
- Domain Name System
- SNMP

INTRODUCTION

Applications Layer – allows user to interface with the network!

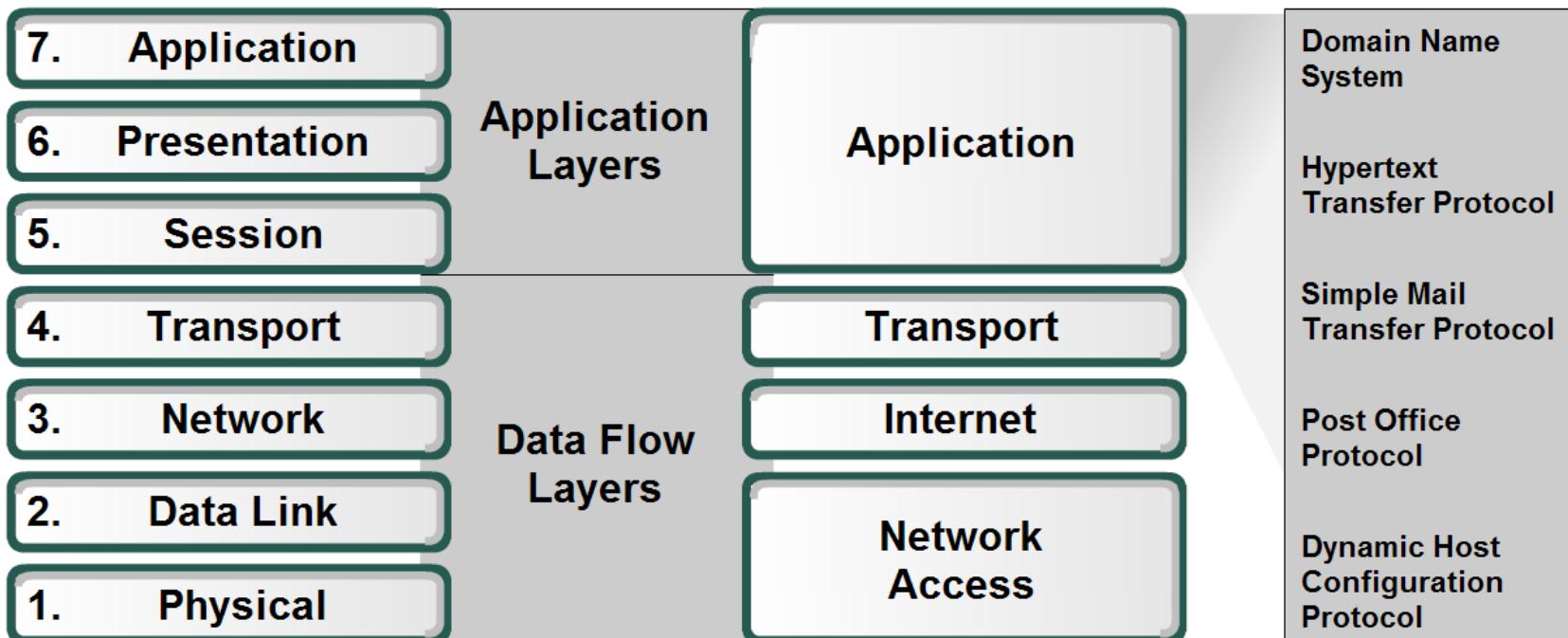


Application Layer – Provides the interface between the applications on either end of the network.

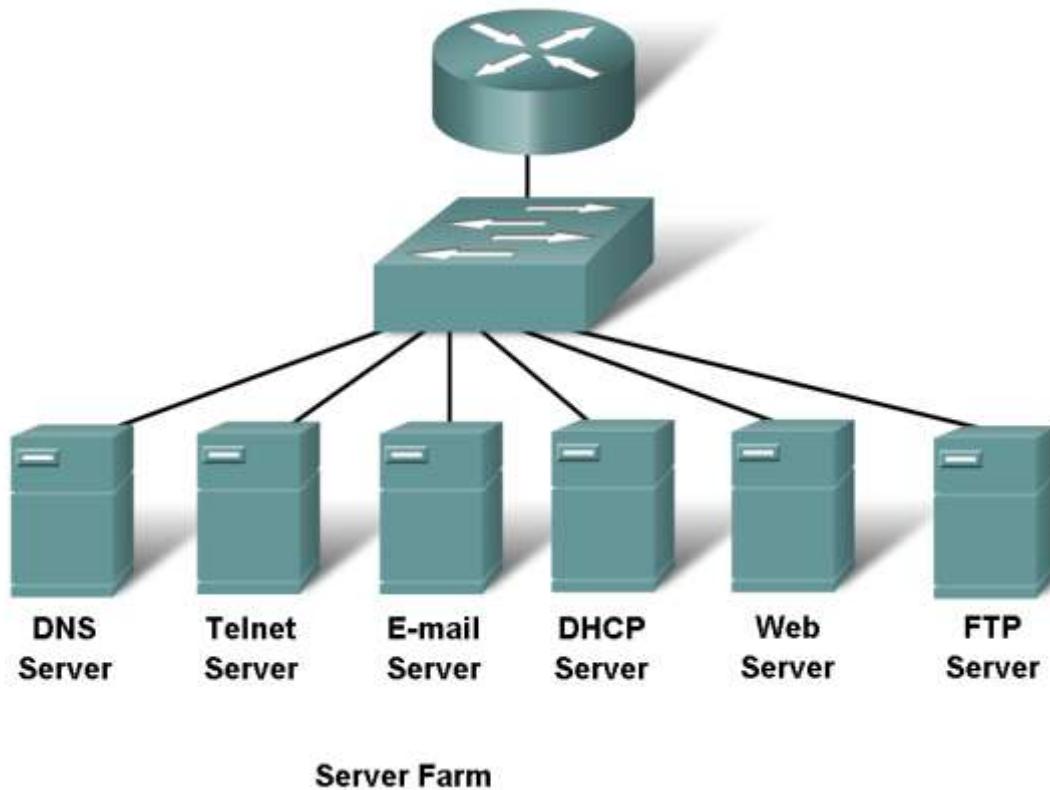


OSI Model

TCP/IP Model



Protocols and networks



Protocols

- DNS – Matches domain names with IP addresses
- HTTP – Used to transfer data between clients/servers using a web browser
- SMTP & POP3 – used to send email messages from clients to servers over the internet
- FTP – allows the download/upload of files between a client/server
- Telnet – allows users to login to a host from a remote location and take control as if they were sitting at the machine (virtual connection)
- DHCP – assigns IP addresses, subnet masks, default gateways, DNS servers, etcs. To users as they login the network

Application layer software

- 2 types
 - Applications – Provide the human (user) interface. Relies on lower layers to complete the communication process.
 - Services – establish an interface to the network where protocols provide the rules and formats that govern how data is treated.

How data requests occur & are filled

- Client/server model
 - Advantages:
 - Centralized administration
 - Security is easier to enforce
- Application layer services and protocols
- Peer-to-peer networking and applications

Client/Server Model

- Client –
 - device requesting information (initiates the data exchange)
 - Can also UPLOAD data to the servers
- Server – device responding to the request
 - How does it handle multiple request from multiple users and keep everything in order?
 - Relies on support from the lower layer functions to distinguish between services and conversations.
 - Server relies on a service called a server daemon – runs in the background and ‘listens’ for requests for that service. It can then exchange messages as appropriate & send requested data.
- Examples:
 - E-mail Client on an employee computer issues a request to the e-mail server for any unread e-mail. The server responds by sending the e-mail to the client.
 - Conversations can originate with either party.

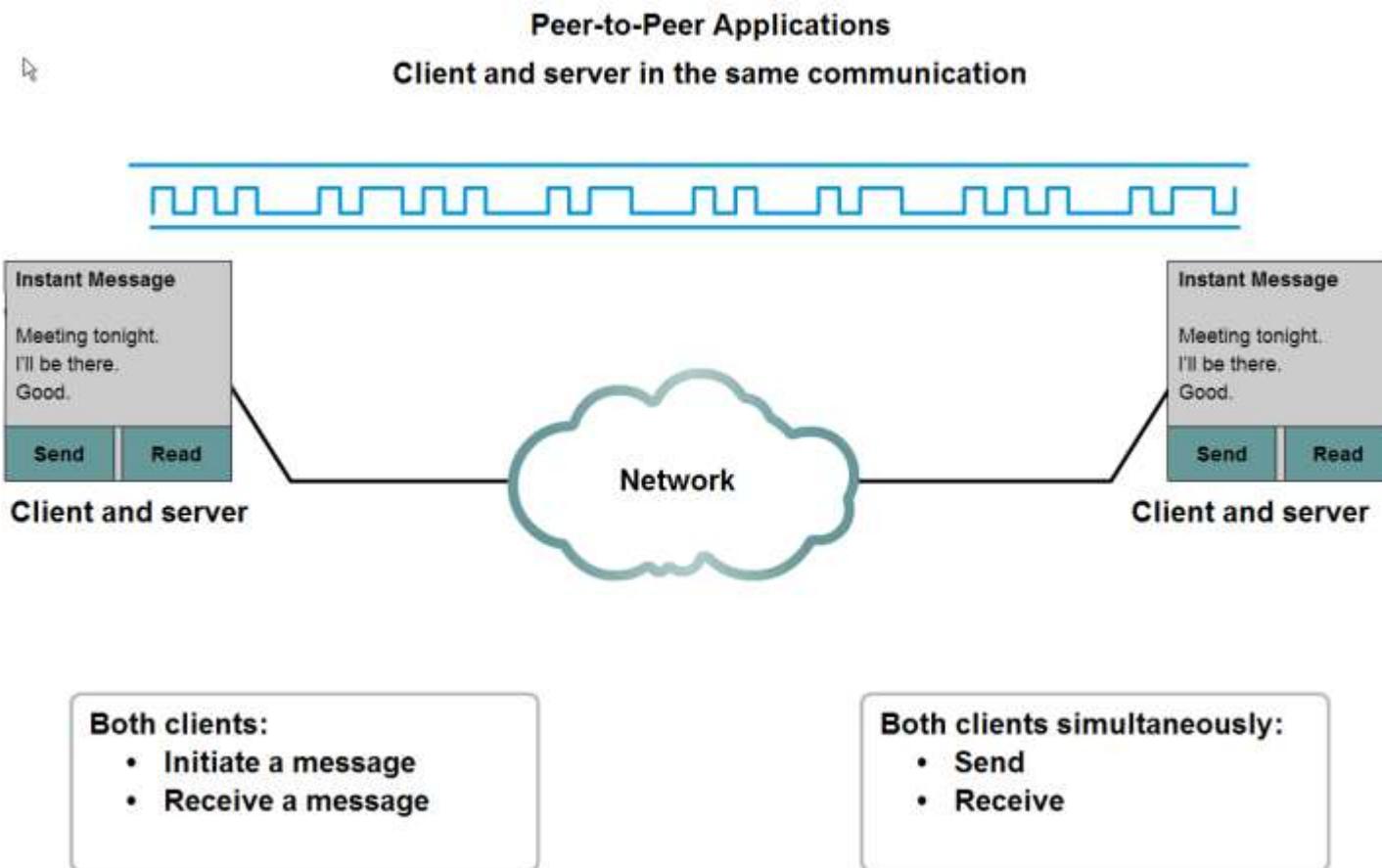
Peer-to-Peer (P2P) Network Model

- Two or more computers are connected and are able to share resources **without having a dedicated server**
- Every end device can function as a client or server on a ‘per request’ basis
- Resources are **decentralized** (information can be located anywhere)
- Difficult to enforce security and policies
- User accounts and access rights have to be set individually on each peer device

P2P Applications

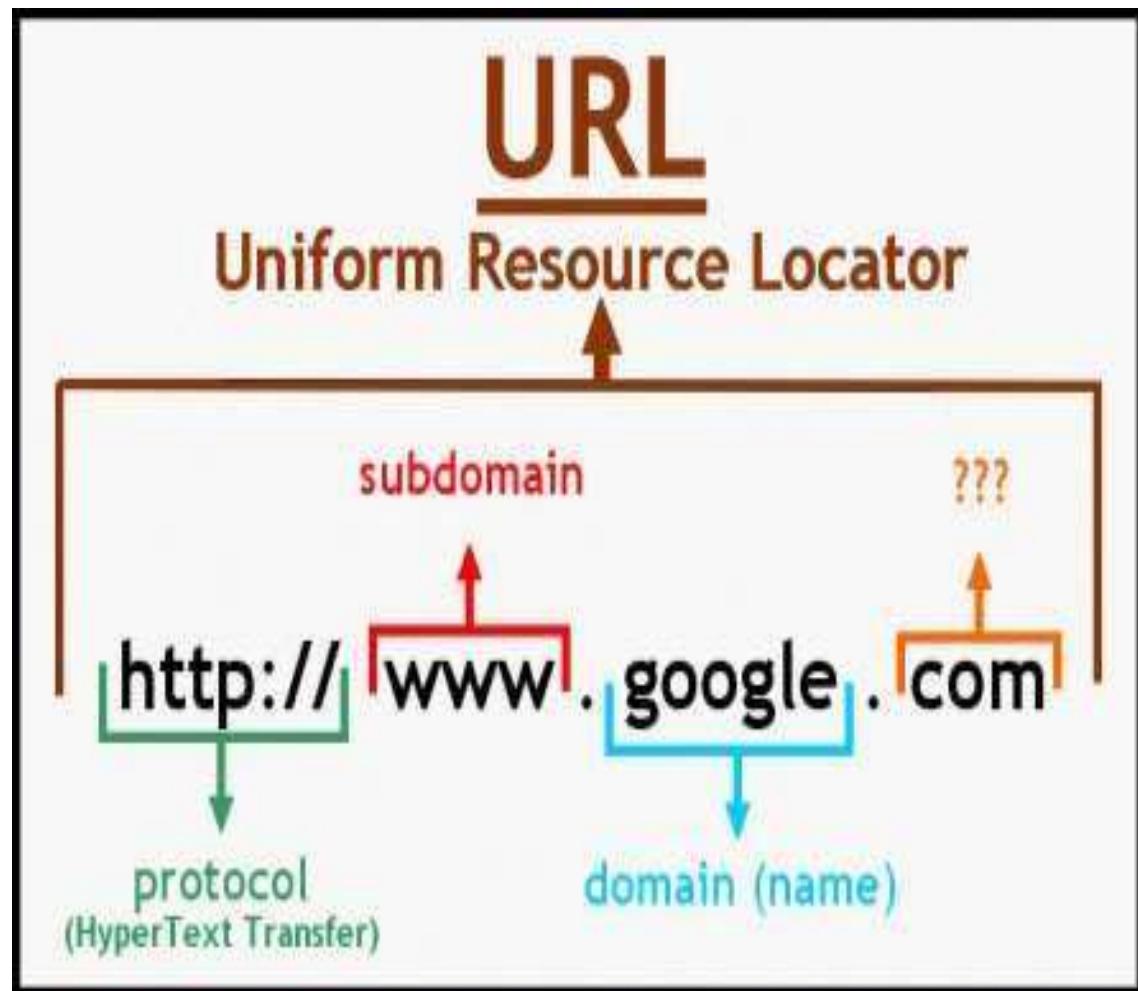
- Running applications in **hybrid** mode allows for a centralized directory of files even though the files themselves may be on multiple machines
- Unlike P2P networks, a device can act as both the client and server within the same communication
- Each device must provide a user interface and run a background service.
- Can be used on P2P networks, client/server networks and across the internet.

P2P Applications Example



WWW and HTTP

What is
URL?



What is HTTP ?

HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Response.

HTTP is an application layer protocol

Connectionless protocol

stateless

HTTP connections

Nonpersistent HTTP

- At most one object is sent over a TCP connection.

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.

HTTP session consist of three phases:



The client establishes a TCP connection



The client sends its request and then waits for the answer



The server processes the request and sends back its answer, containing a status code and the appropriate data

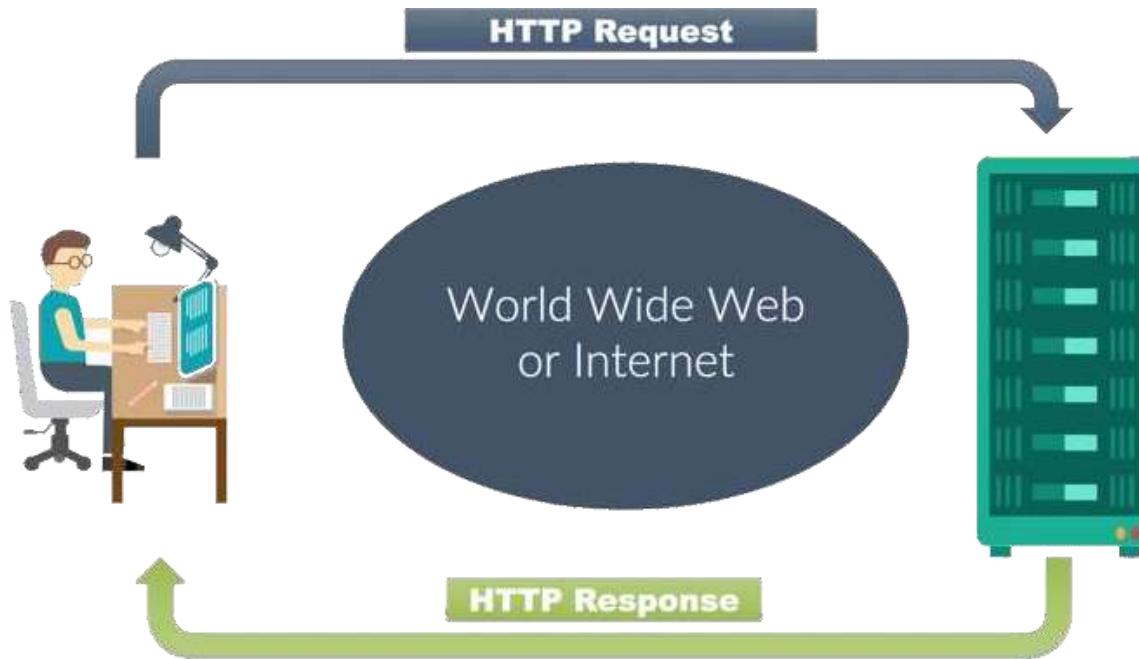


The connection is closed.

HTTP Methods

- GET**> Request for a web page or an object from server
- PUT**> For sending a document to the server
- POST**> For sending data or information about client to the server
- DELETE**> Request to Delete an object on the server
- HEAD**> Request for information about a web page or a document
- TRACE**> Used to trace the proxies and tunnels in the path from client to server
- OPTION**> Used to determine server's capabilities

HTTP Request / Response



Communication between clients and servers is done by requests and responses:

1. A client (a browser) sends an HTTP request to the web
2. An web server receives the request
3. The server runs an application to process the request
4. The server returns an HTTP response (output) to the browser
5. The client (the browser) receives the response

The HTTP Request Circle

A typical HTTP request/response circle:

The browser requests an HTML page. The server
~~returns an HTML file~~

The browser requests a style sheet. The server returns a
~~css file~~

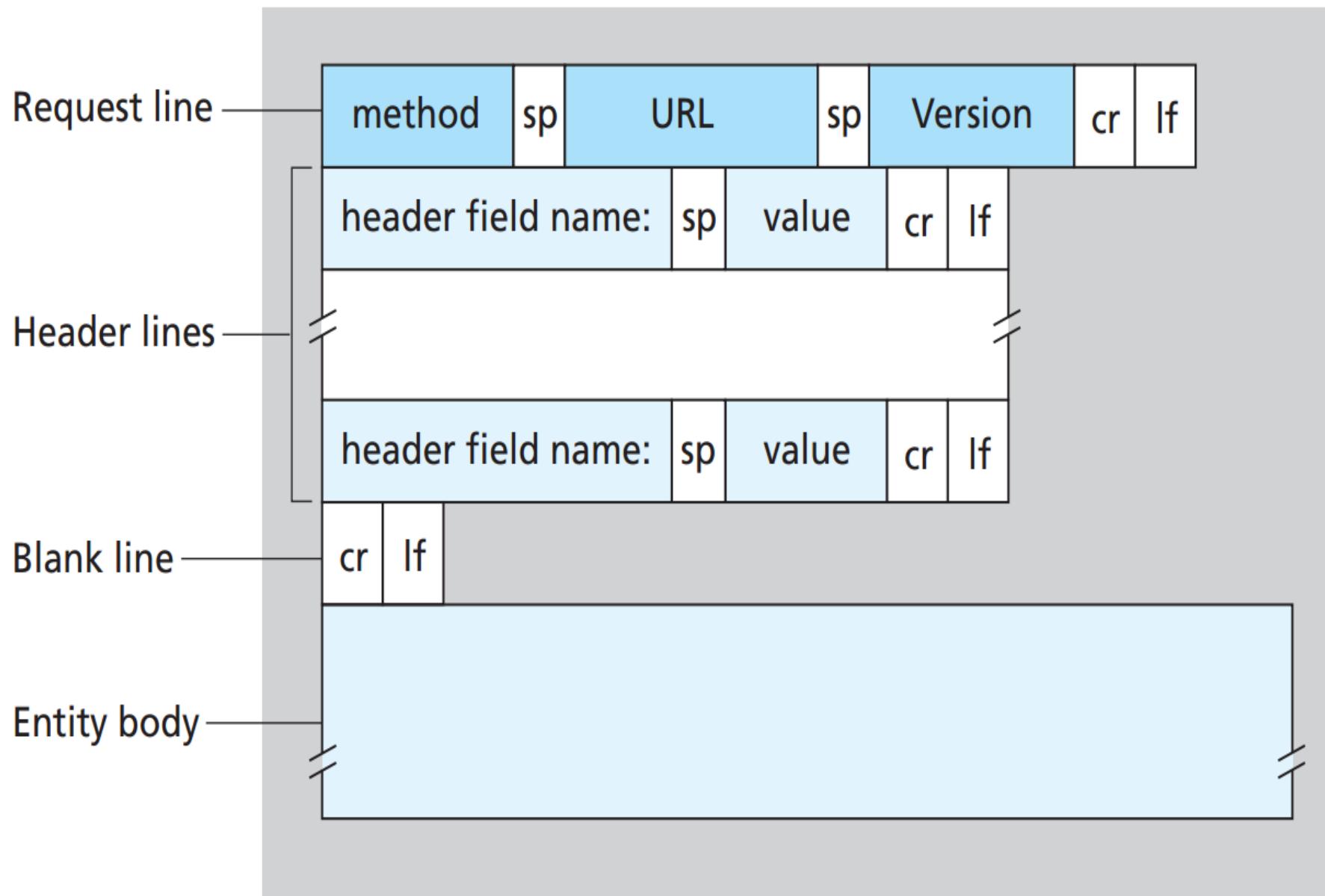
The browser requests an JPG image. The server
~~returns a JPG file~~

The message format consists of the following 3 items:

Start-line

Header Fields

Message Body



Headers

HTTP message headers are used to precisely describe the resource being fetched or the behaviour of the server or the client. There are four types of HTTP message headers:

1. General headers

Table 27.3 General headers

Header	Description
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

2. Request headers

Table 27.4 Request headers

<i>Header</i>	<i>Description</i>
Accept	Shows the medium format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the server
If-modified-since	Sends the document if newer than specified date
If-match	Sends the document only if it matches given tag
If-non-match	Sends the document only if it does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

3. Response headers

Table 27.5 *Response headers*

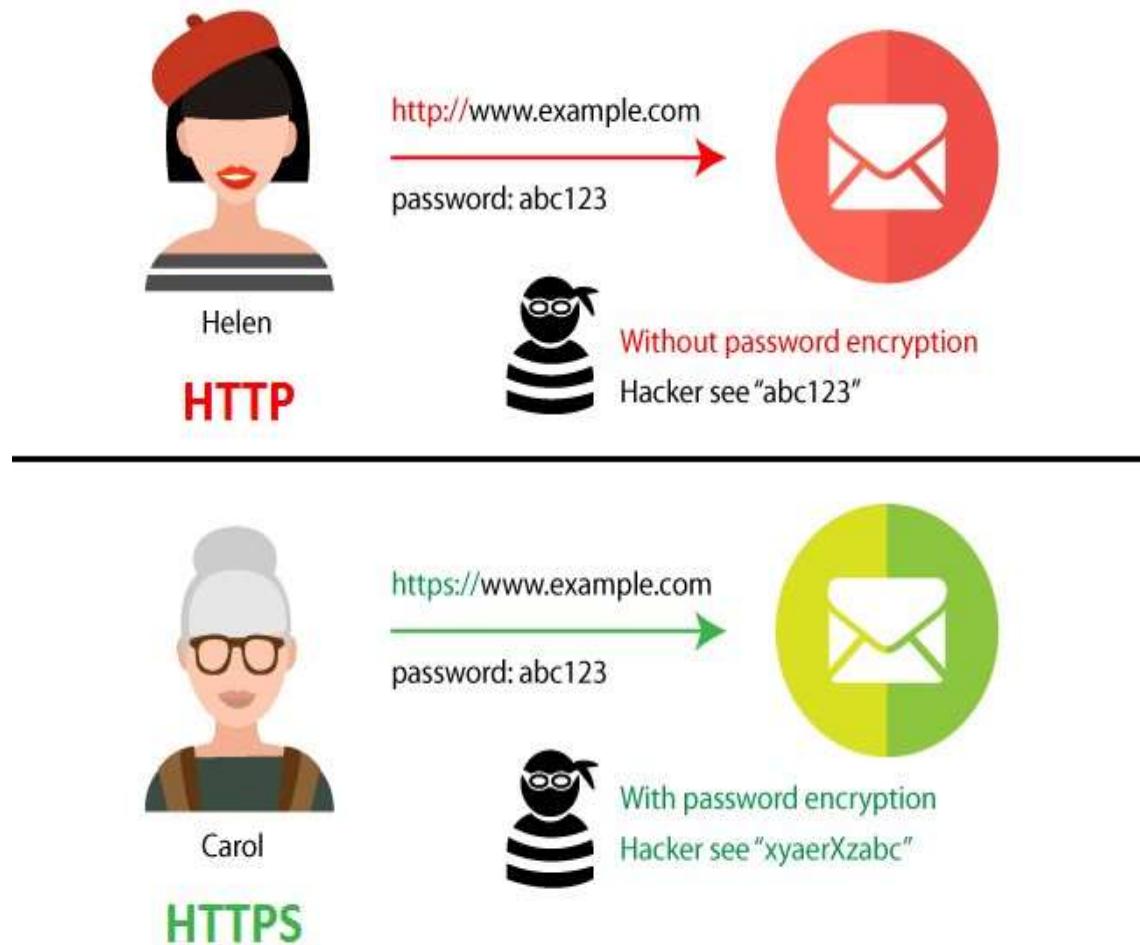
<i>Header</i>	<i>Description</i>
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

4. Entity headers

Table 27.6 *Entity headers*

<i>Header</i>	<i>Description</i>
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

HTTP VS HTTPS



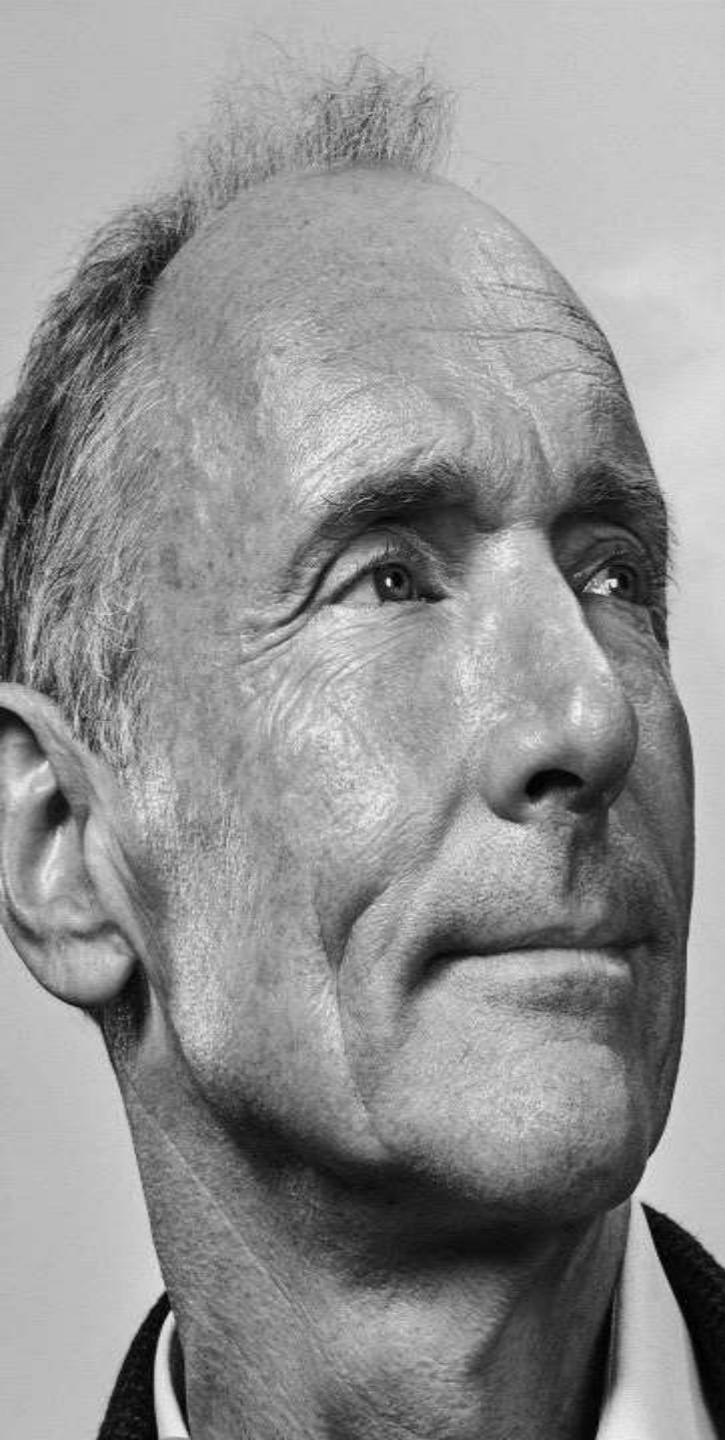
What is World Wide Web ?



- The World Wide Web is a way of exchanging information between computers on the Internet.
- The World Wide Web is the network of pages of images, texts and sounds on the Internet which can be viewed using browser software .

The World Wide Web

- *The World Wide Web, or Web, consists of a worldwide collection of electronic documents (Web pages)*
- *A Web site is a collection of related Web pages and associated items*
- *A Web server is a computer that delivers requested Web pages to your computer*
- *Web 2.0 refers to Web sites that provide a means for users to interact*



History



(www) was Invented between 1989 – 1990,
by Tim Berners Lee at Cern .



Published in
1992

- Means for transferring
text and graphics



Using Text mark up
language (HTML)

- invented by Bernes-Lee
- Simple and easy to use

WWW Components

1. Structural Components:

- Clients/browsers
- Servers
- Caches
- Internet

2. Semantic Components:

- Hyper Text Transfer Protocol (HTTP)
- Hyper Text Markup Language (HTML)
- Extensible Markup Language (XML)
- Uniform Resource Identifiers (URIs)

-The Fundamental Concept of World Wide Web-

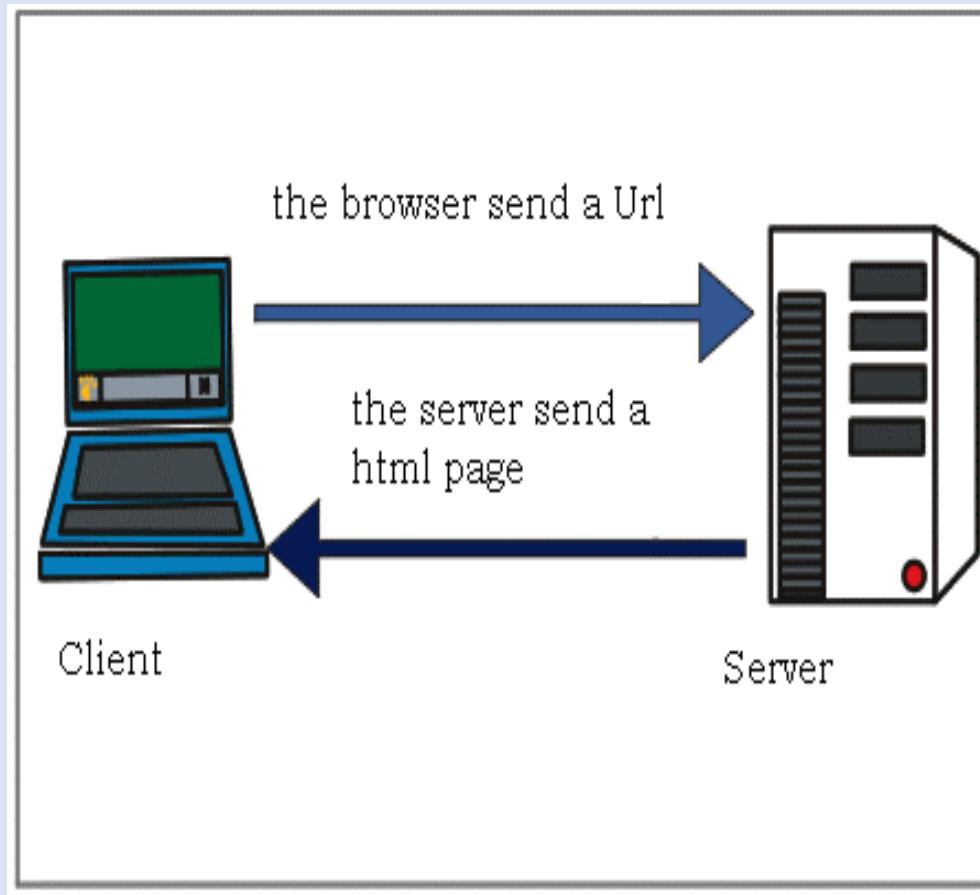
1. The Hypertext concept

Hypertext is text which contains links to other texts.

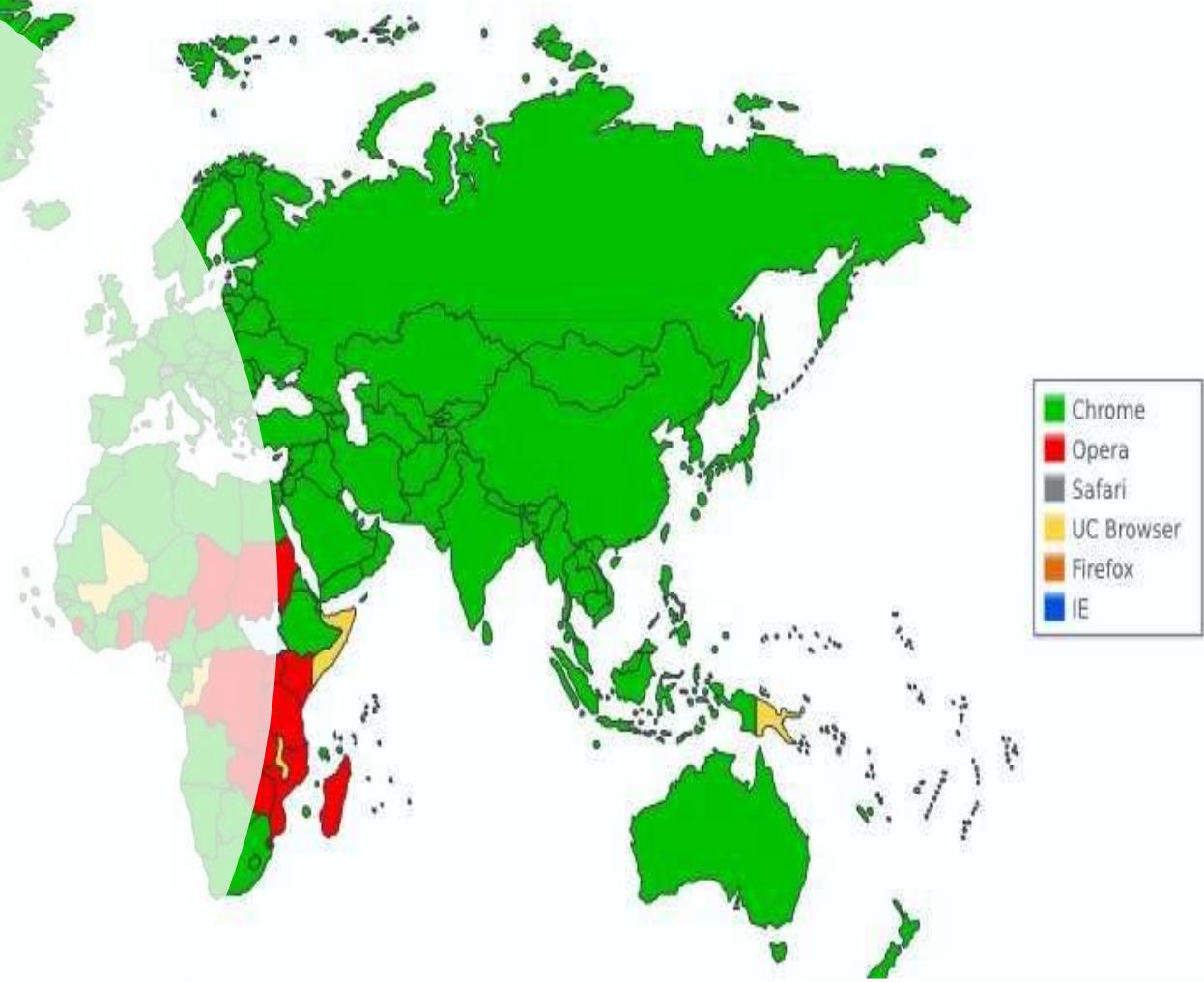
2. The Hypermedia concept

Hypermedia is a term used for hypertext which is not constrained to be text: it can include graphics, video and sound

- **Web Browser:**
A web browser displays a web document and enables users to access web documents.
- **Web Server:**
This is a program that waits patiently for the browser to request a web page. The servers looks for the requested information, retrieves it and send it to the browser or sends an error message if the file is not found
- **URL (Uniform Resource Locator):**
These are the web addresses. The resource locator is an addressing system .



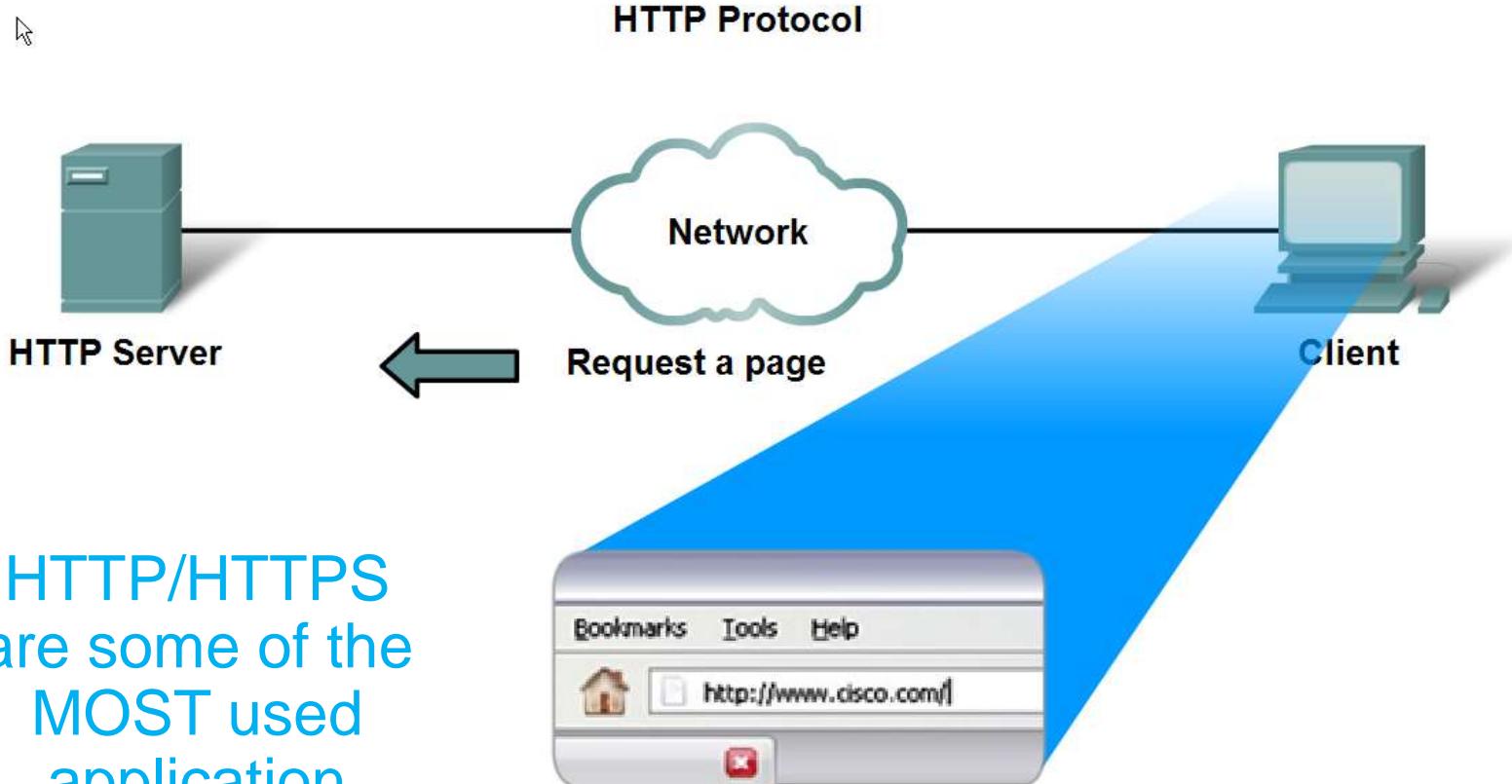
Most
popular
browser
—rs



WWW Service and HTTP

- Steps:
 - 1) URL is typed in the address bar.
 - 2) Browser checks with DNS server to convert it to an IP address
 - 3) Connects to the server requested
 - 4) Using **HTTP** or **HTTPS** protocol requirements, the browser sends a GET request to the server to ask for the desired html document (usually index.html)
 - 5) The server sends the HTML code for the web page to the browser.
 - 6) The browser interprets the HTML code and formats the page to fit the browser window.
 - 7) See the next slide for an example.

WWW Service and HTTP



HTTP/HTTPS
are some of the
MOST used
application
protocols!

FTP

File Transfer Protocol

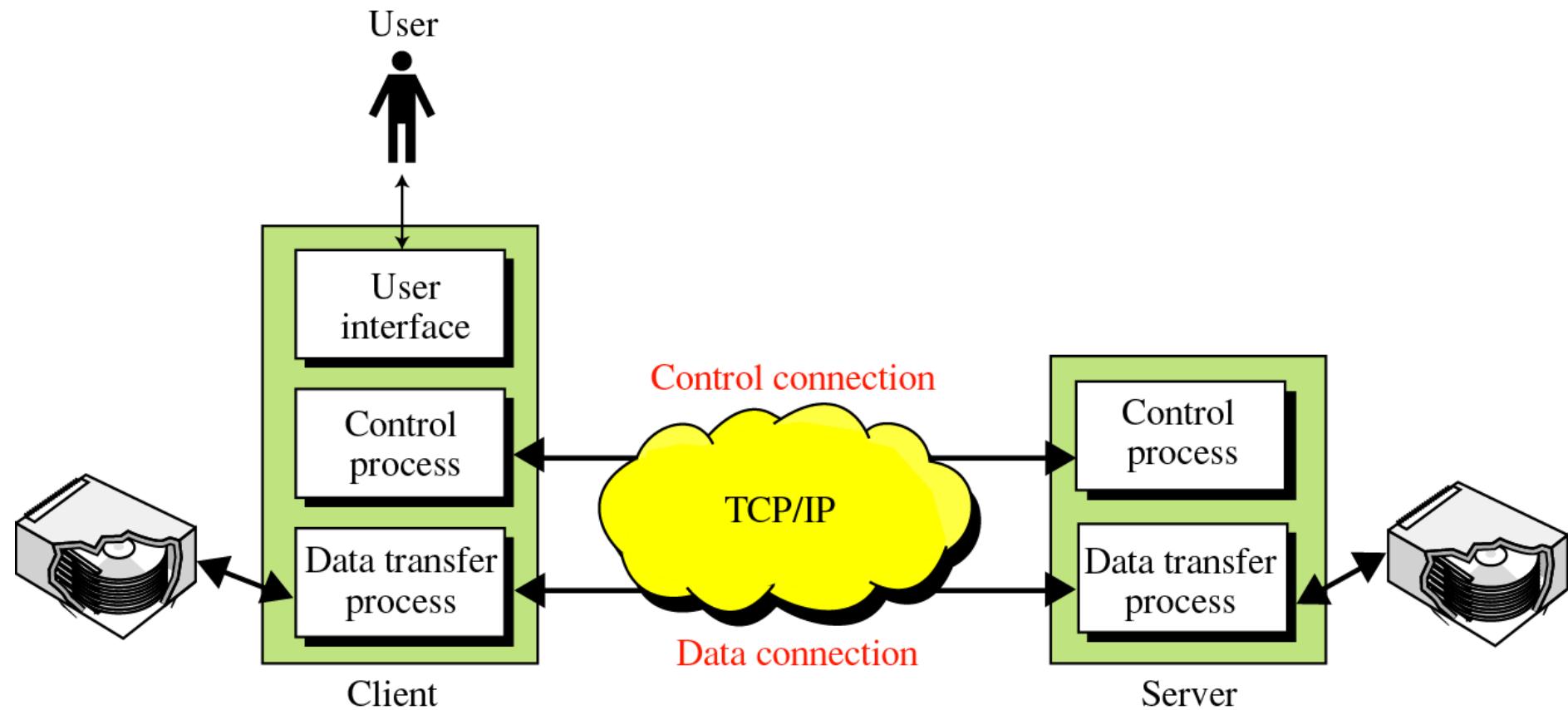
- CONNECTIONS
- COMMUNICATION
- COMMAND PROCESSING
- FILE TRANSFER
- USER INTERFACE
- ANONYMOUS FTP

FTP

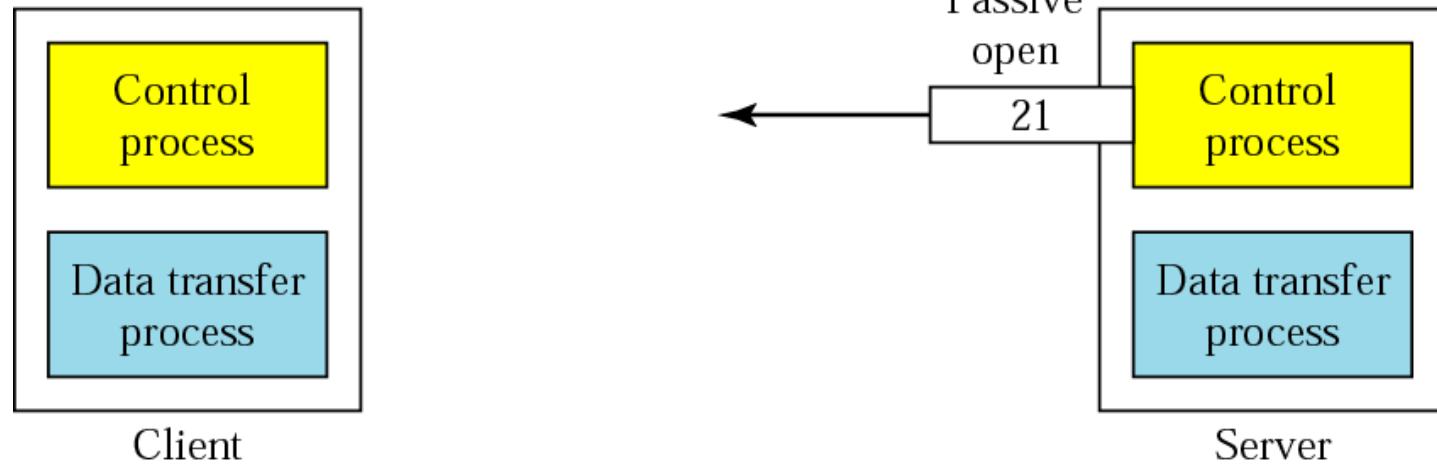
- Commonly used application layer protocol
- Allows for the transfer of files between clients/servers.
- Requires 2 connections to the server
 - 1) Commands – uses TCP port 21
 - 2) Actual data – uses TCP port 20

*FTP uses the services of TCP.
It needs two TCP connections.
The well-known port 21 is used
for the control connection
and the well-known
port 20 for the data connection.*

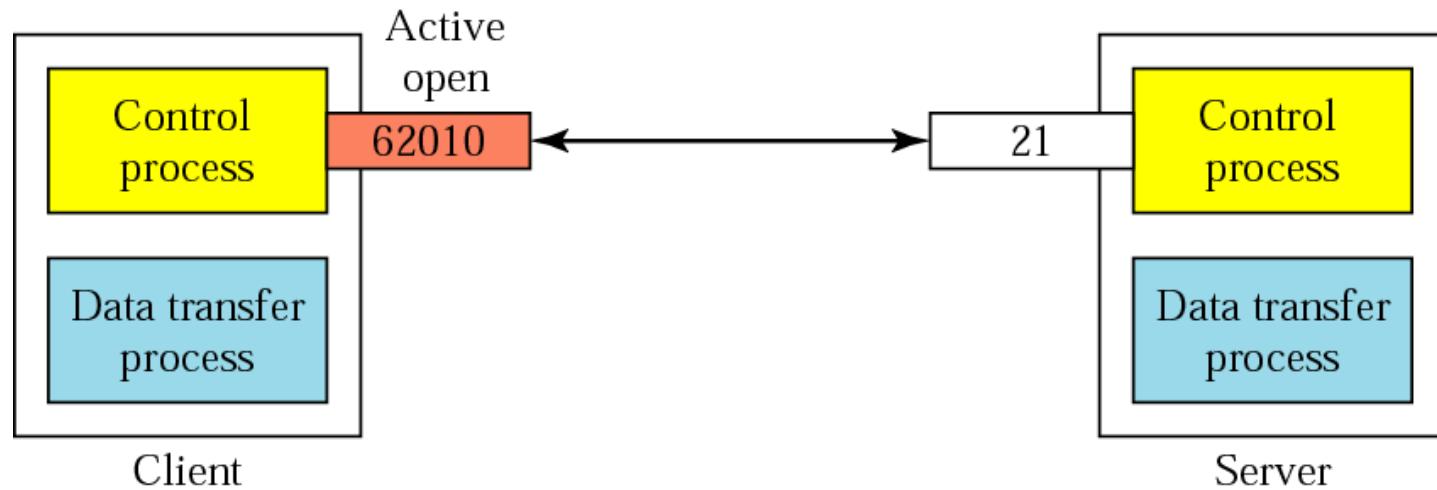
FTP



Connections: The control connection



a. Passive open by server

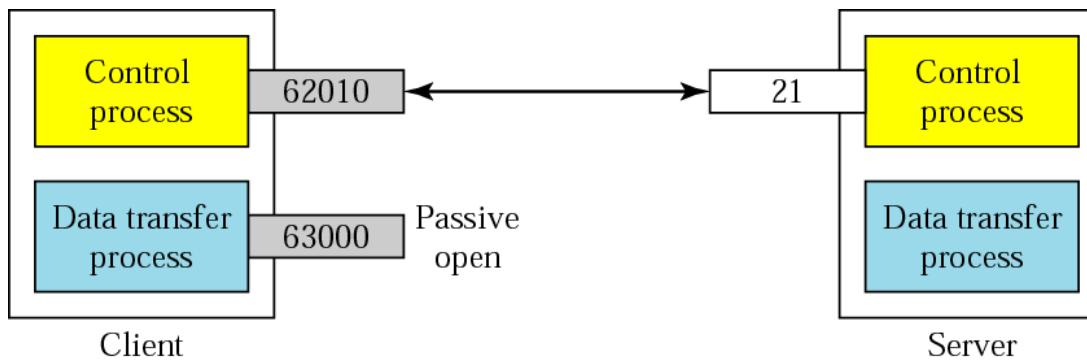


b. Active open by client

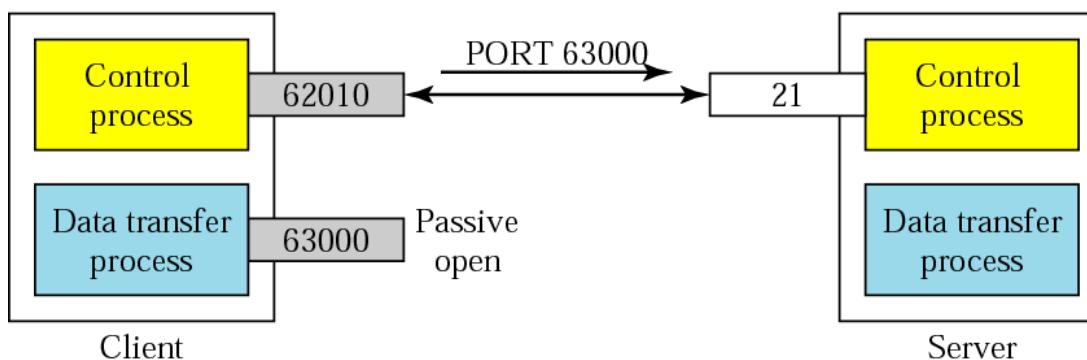
The Data Connection

- Uses Server's well-known port 20
1. Client issues a passive open on an ephemeral port, say x .
 2. Client uses PORT command to tell the server about the port number x .
 3. Server issues an active open from port 20 to port x .
 4. Server creates a child server/ephemeral port number to serve the client

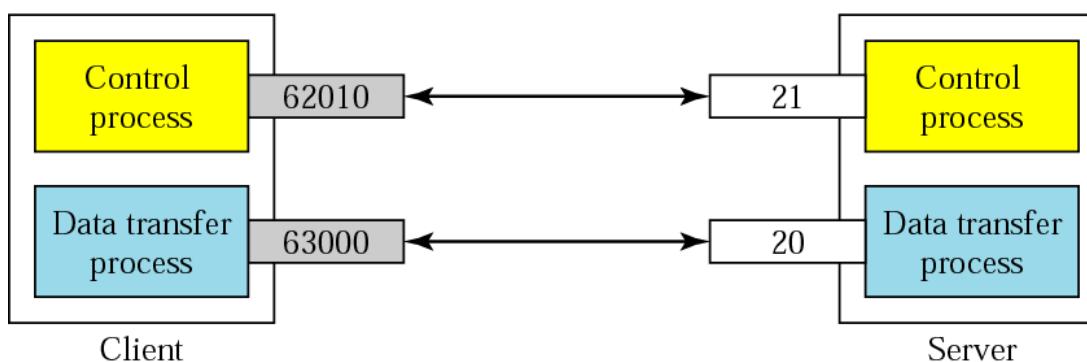
Creating the data connection



a. Passive open by client

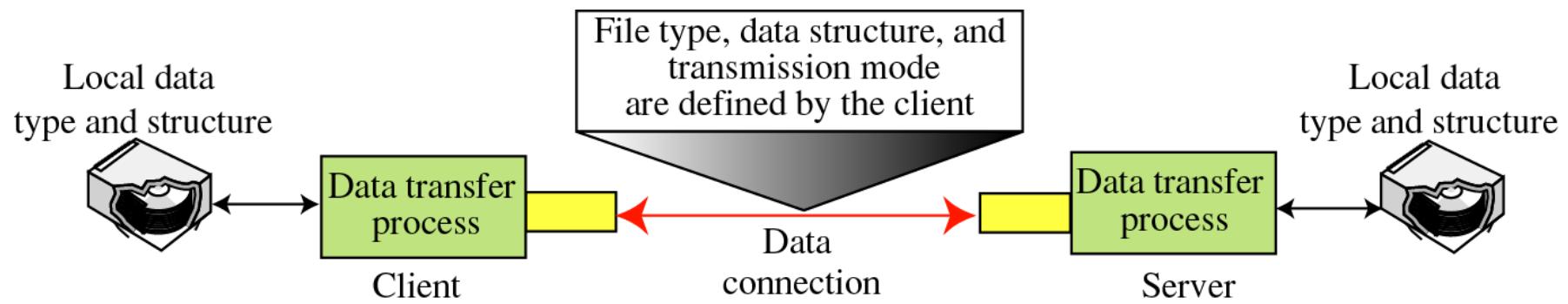


b. Sending ephemeral port number to server



c. Active open by server

Using the data connection



File Type

- ASCII or EBCDIC
 - Nonprint
 - TELNET
- Image

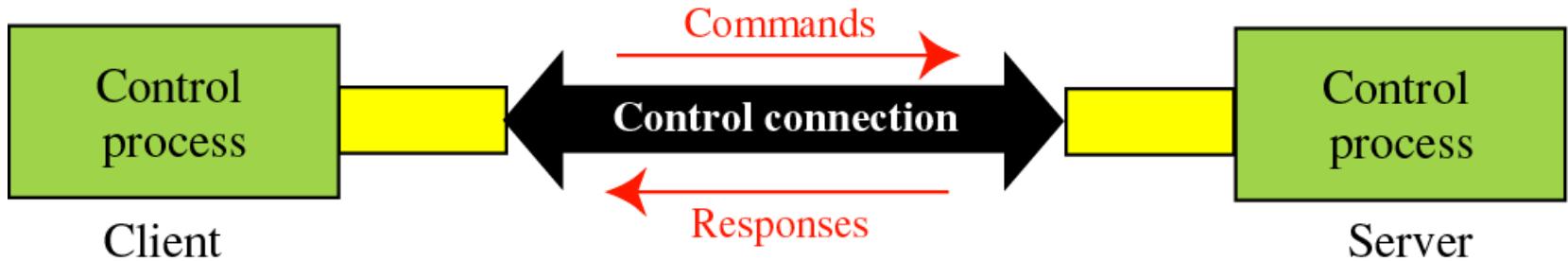
Data Structure

- File Structure
- Record Structure
- Page Structure

Transmission Mode

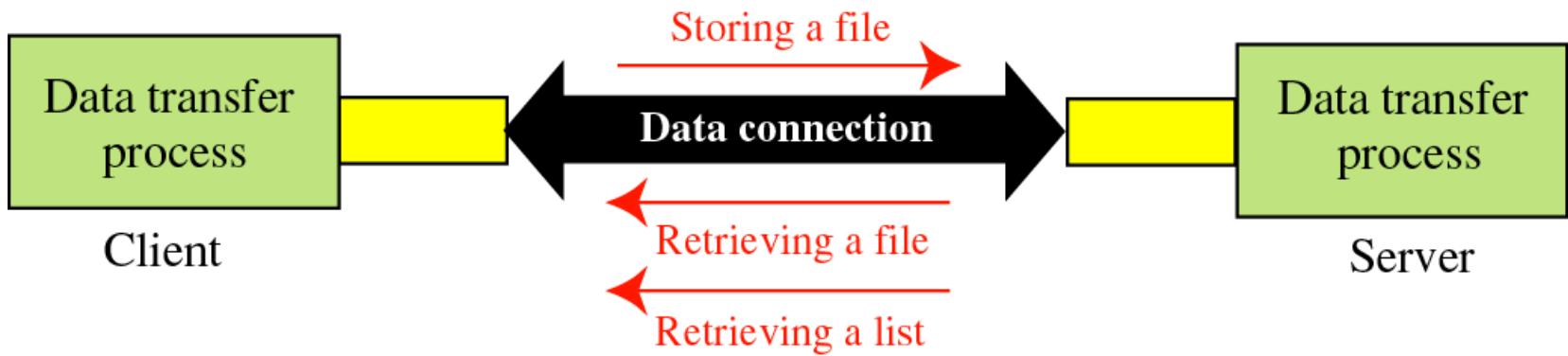
- Stream mode
- Block mode
- Compressed mode

Command processing

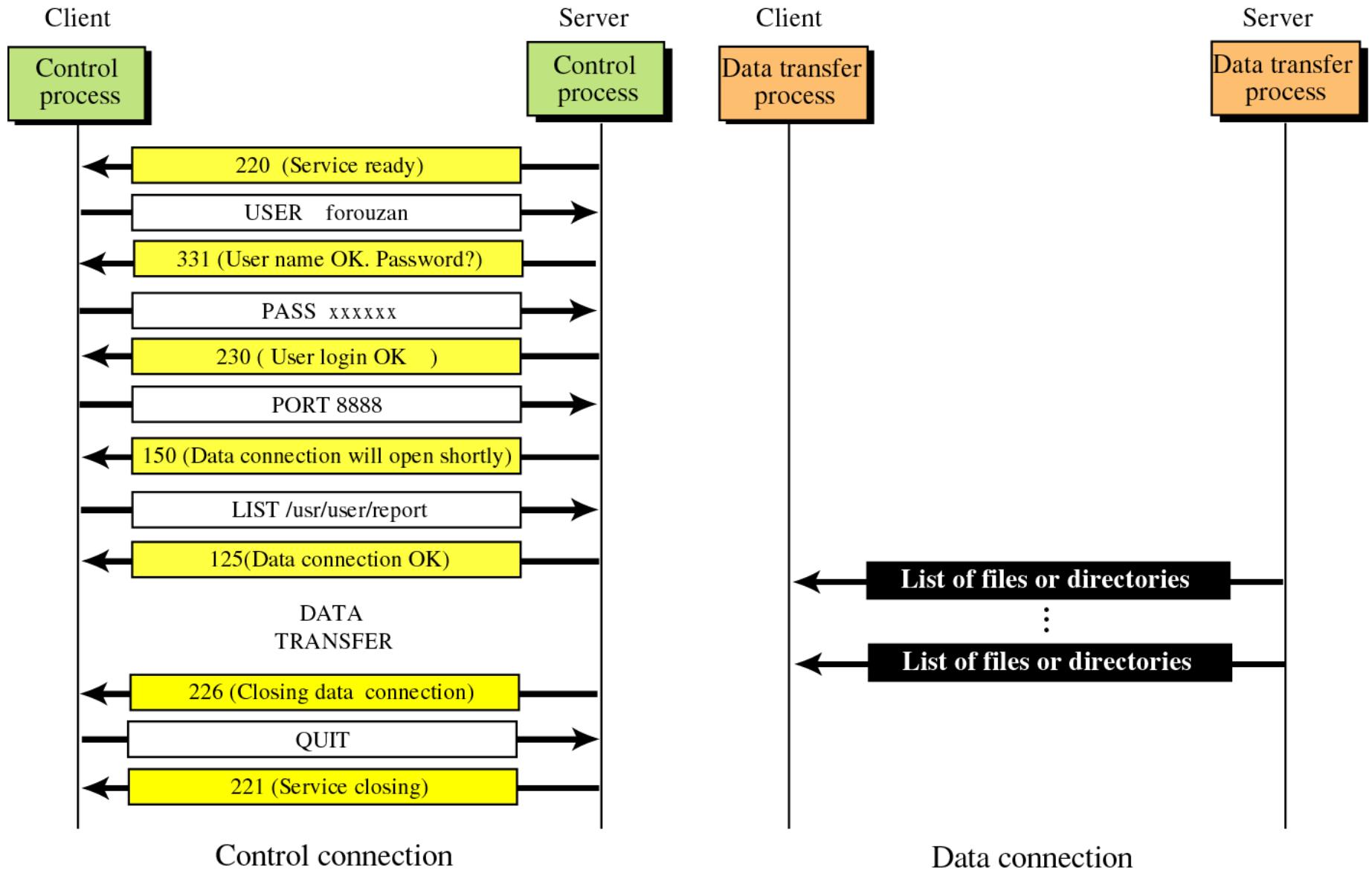


- Access Commands
- File Management
- Data Formatting
- Port defining
- File transfer
- Miscellaneous

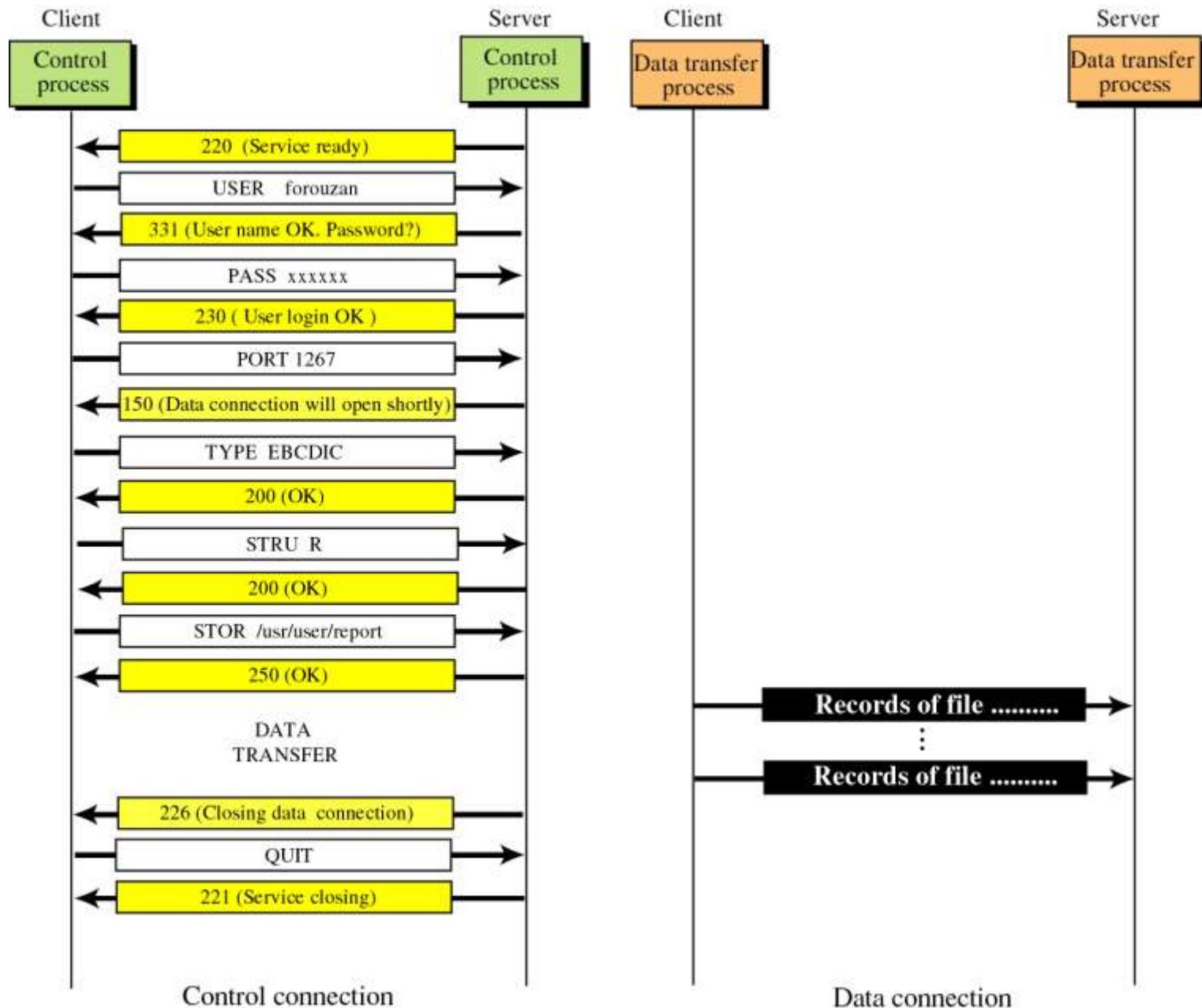
File transfer



Example 1



Example 2



E-Mail

- Email Basics
- What Makes Up An Email
- How Email Works
- What Are TCP/IP Ports

Email Basics

- What is an Email – an electronic message transmitted over a network from one user to another.
- Can be as simple as a few lines of text, or include attachments such as pictures or documents.
- Email made up 75% of network traffic soon after the introduction of the internet.

What Makes Up An Email

- The Header
 - Who sent the email.
 - To whom the mail is sent.
 - When the email was sent.
 - The email subject.
 - The size of the email.

What Makes Up An Email

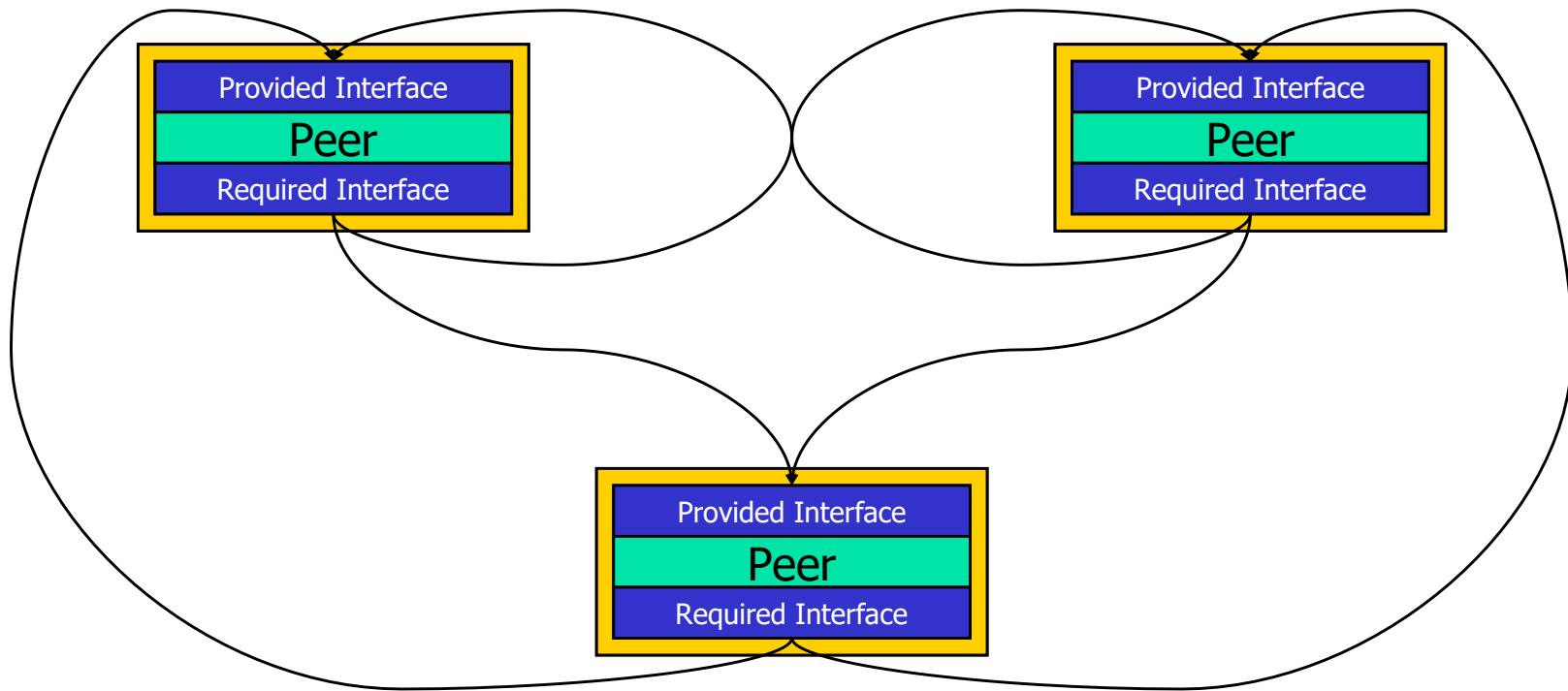
- The Body
 - Contains the message.
 - May also contain an attachment.
- Attachments
 - If not embedded within the body, attachments are sent along with the email.

How Email Works

- Different Architectural Models exist for constructing computer systems.
- Some models include:
 - Peer-Peer
 - Pipe and Filter
 - Implicit Invocation
 - Client-Server

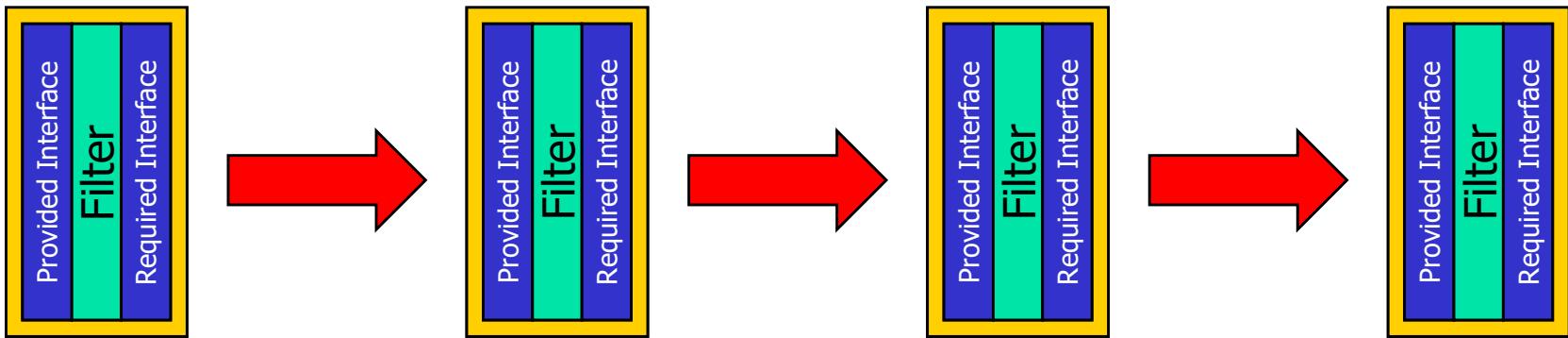
How Email Works

Peer-Peer Model



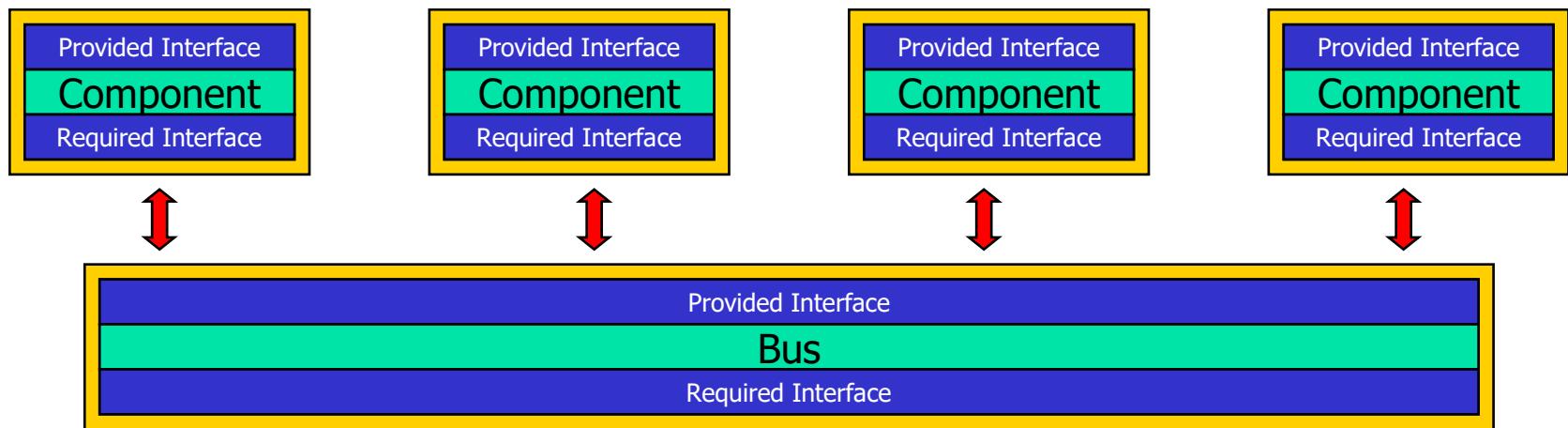
How Email Works

Pipe and Filter Model



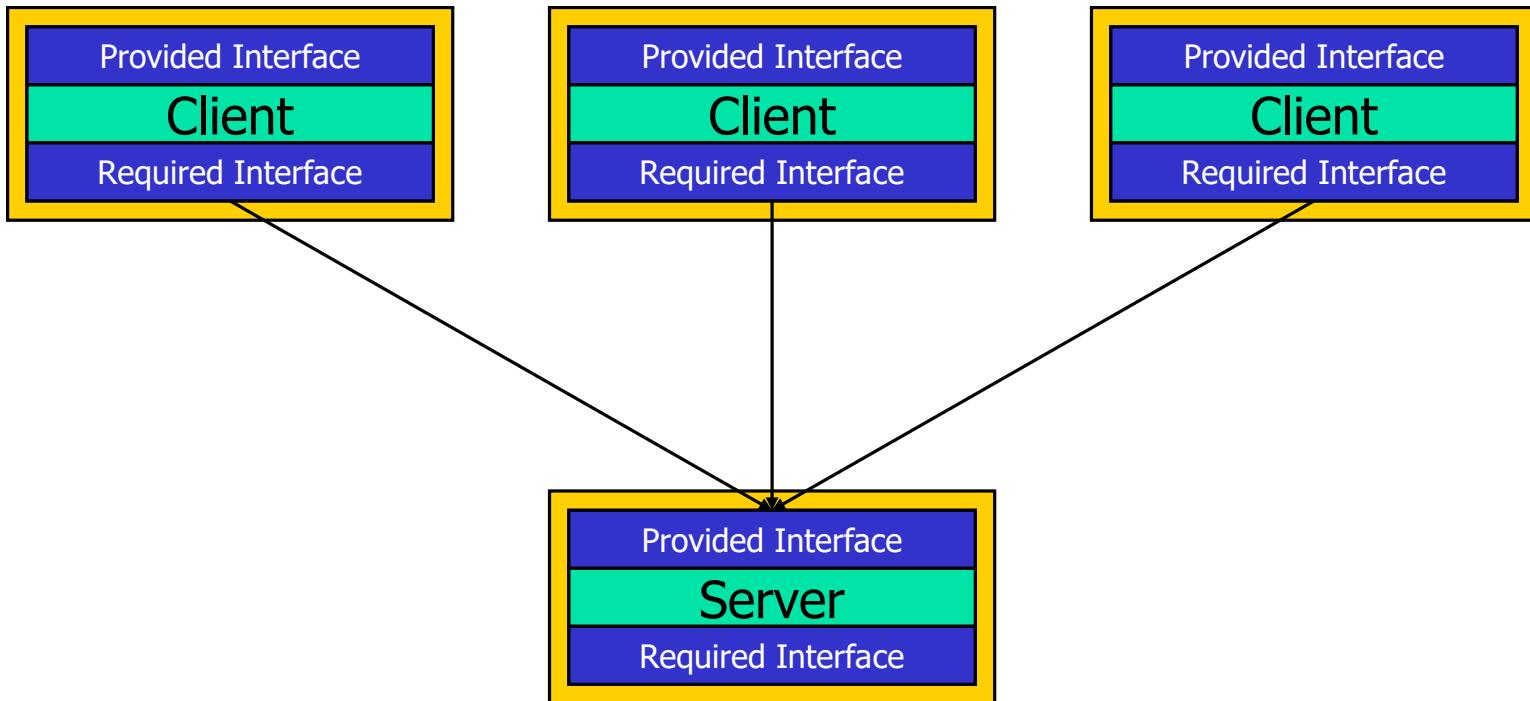
How Email Works

Implicit Invocation “Bus” Model



How Email Works

Client-Server Model



How Email Works

- The model that works best for email is the Client-Server model.
- Clients carry out user interactions with the email server.

How Email Works

Clients

- Forms in which clients appear:
 - Application based - these are installed onto user's machines and include Microsoft Outlook and the freely available Outlook Express and Eudora.
 - Web based - these appear in a web browser's window and include Hotmail, Yahoo and Outlook web client.

How Email Works

Clients

- Clients vary greatly in functionality, but all provide a basic level of functionality that assists the user.
- Basic functions include:
 - Ability to create new emails.
 - Display and store received emails.
 - Hold address lists of contacts, a calendar, journal and other extra functions that help organize the user's working day.
 - The client is also configured with the account information and names or IP addresses of the email servers with which it will be communicating.

How Email Works

Servers

- An email server is typically a combination of processes running on a server with a large storage capacity – a list of users and rules, and the capability to receive, send and store emails and attachments.
- These servers are designed to operate without constant user intervention.
- Should process emails for months as sending, receiving and maintenance tasks are carried out at scheduled times. The client only has to connect to the email server when it sends and checks/receives new email.
- Sometimes it may be permanently connected to the server to allow access to shared address books or calendar information – this is typical of a LAN-based email server.

How Email Works

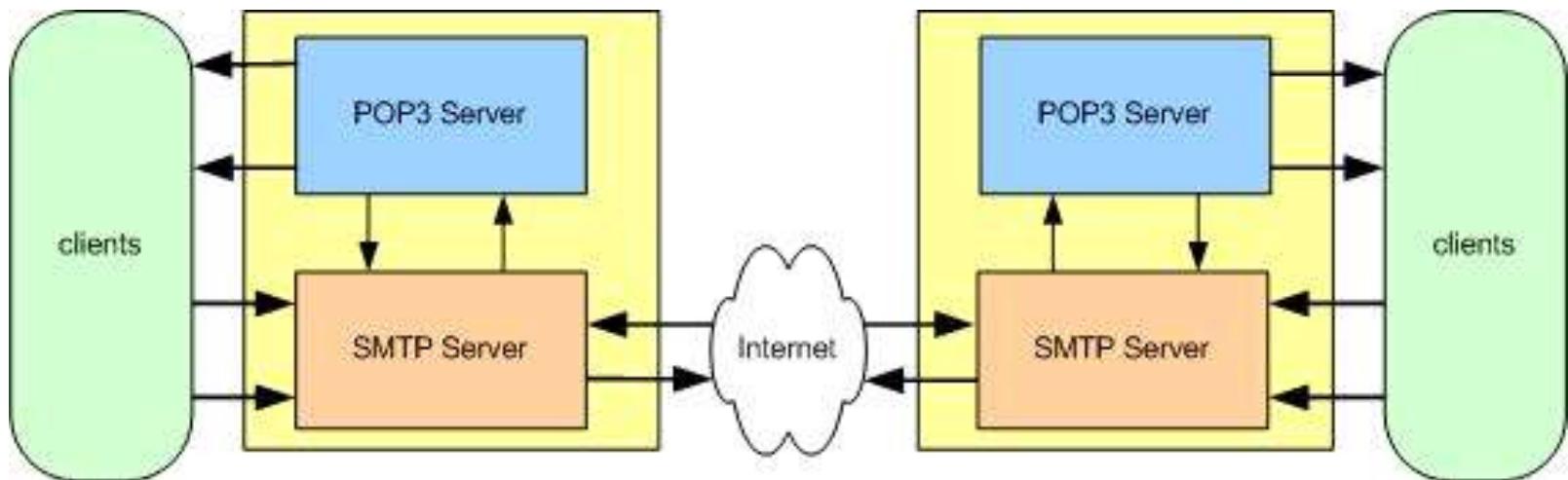
Servers

- Most email servers conduct email services by running two separate processes on the same machine.
- One process is the POP3 (Post Office protocol 3) server, which holds emails in a queue and delivers emails to the client when they are requested.
- The other is the SMTP (simple mail transfer protocol) server that receives outgoing emails from clients and sends and receives email from other SMTP servers.
- These two processes are linked by an internal mail delivery mechanism that moves mail between the POP3 and SMTP servers.

How Email Works

Servers

- When the client calls the email server to send or check for mail it connects to the server on certain TCP/IP ports:
 - SMTP on port 25
 - POP3 on port 110.



How Email Works

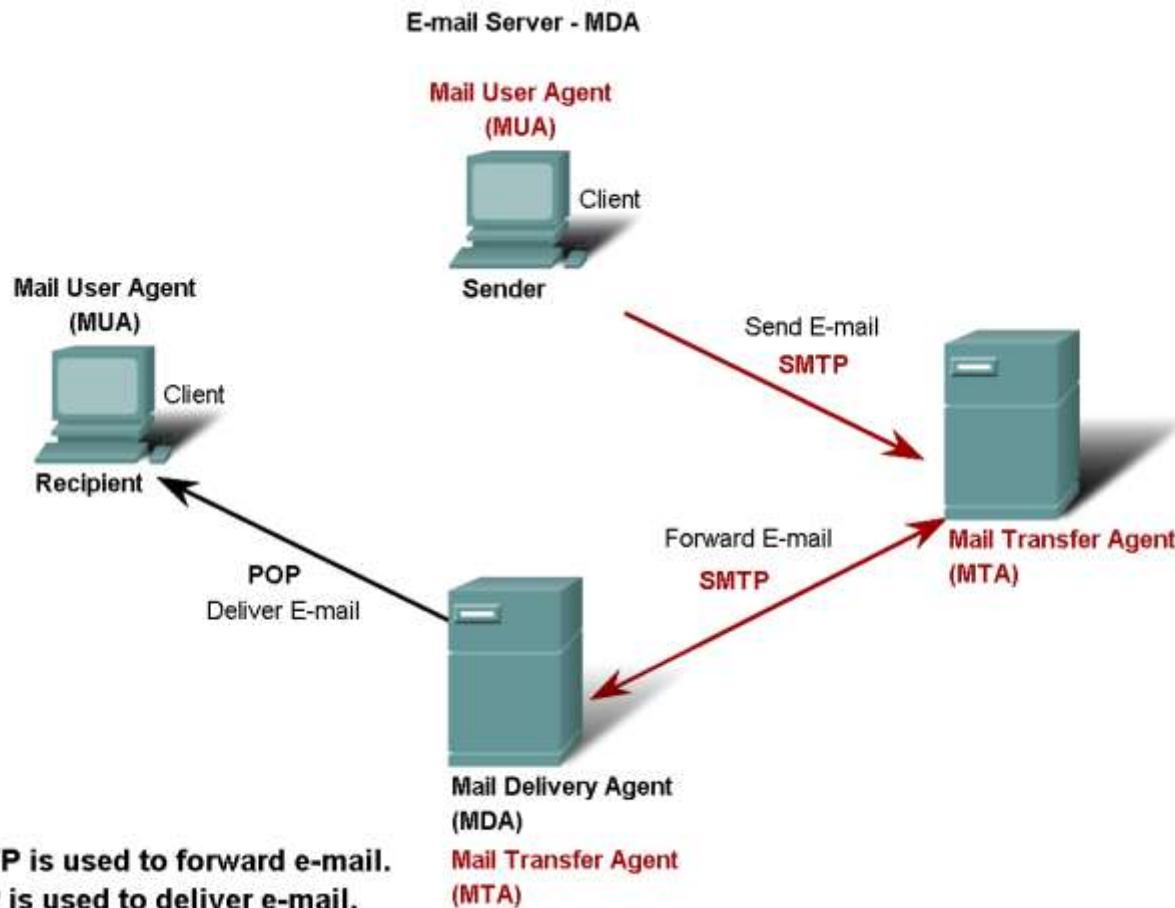
Servers

- Email systems come in various formats, but the most common rely on a single server that provides both POP3 and SMTP services.
- Sometimes, in large organizations, these services are separated onto different servers.
- Currently, the majority of email servers use Windows NT or 2000.

E-mail services and SMTP/POP protocols

- E-mail is the most popular network service.
- E-mail client (when people compose e-mail) is called Mail User Agent (MUA)
- MUA allows messages to be sent/retrieved to and from your mailbox
- Requires several applications and services
 - POP or POP3 – deliver email from server to client (incoming messages)
 - SMTP – handles outbound messages from clients

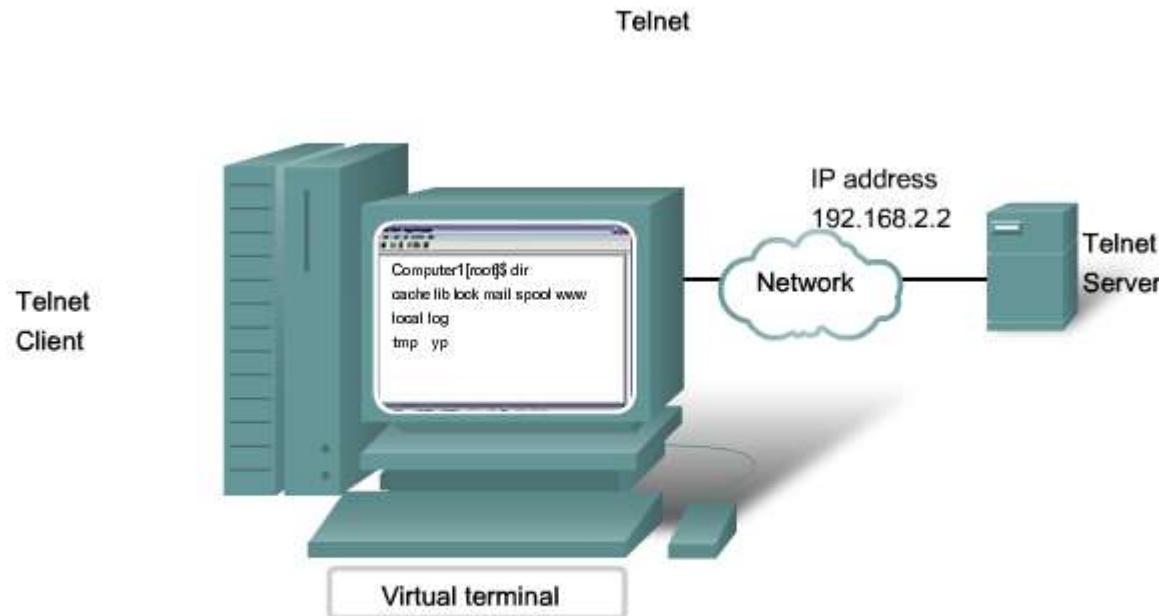
E-mail services and SMTP/POP protocols



Telnet

- Developed in the early 1970's – among the oldest of the application layer protocols and services in the TCP/IP protocol suite.
- Allows users to emulate text-based terminal devices over the network using software.
- A connection is known as a 'virtual terminal (vty)' session.
- Can be run from the command prompt on a PC.
- You can use the device as if you were sitting there with all the rights and priorities that your username will offer you.
- Disadvantages: Doesn't support encryption like SSH. All data is transferred as plain text. It can be easily intercepted and understood.
- If security is a concern, you should use Secure Shell (SSH) protocol. Provides for remote logins with stronger authentication than telnet.
- Network Professionals should always use SSH whenever possible.

Telnet



Telnet provides a way to use a computer, connected via the network, to access a network device as if the keyboard and monitor were directly connected to the device.

TELNET vs. telnet

- TELNET is a *protocol* that provides “a general, bi-directional, eight-bit byte oriented communications facility”.
- telnet is a *program* that supports the TELNET protocol over TCP.
- Many application protocols are built upon the TELNET protocol.

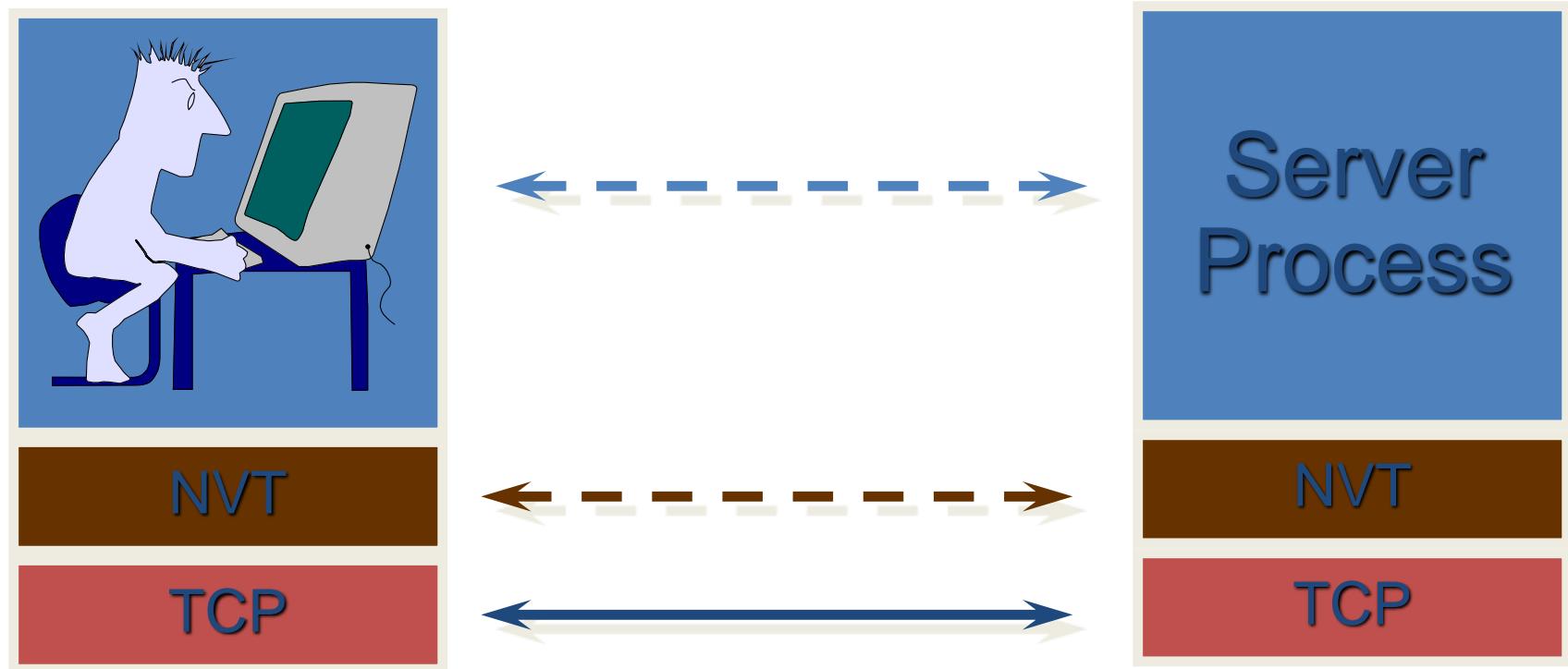
The TELNET Protocol

- TCP connection
- data and control over the same connection.
- Network Virtual Terminal
- negotiated options

Network Virtual Terminal

- Intermediate representation of a generic terminal.
- Provides a standard language for communication of terminal control functions.

Network Virtual Terminal



Negotiated Options

- All NVTs support a minimal set of capabilities.
- Some terminals have more capabilities than the minimal set.
- The 2 endpoints negotiate a set of mutually acceptable options (character set, echo mode, etc).

Negotiated Options

- The protocol for requesting optional features is well defined and includes rules for eliminating possible negotiation “loops”.
- The set of options is not part of the TELNET protocol, so that new terminal features can be incorporated without changing the TELNET protocol.

Option examples

- Line mode vs. character mode
- echo modes
- character set (EBCDIC vs. ASCII)

Control Functions

- TELNET includes support for a series of control functions commonly supported by servers.
- This provides a uniform mechanism for communication of (the supported) control functions.

Control Functions

- Interrupt Process (IP)
 - suspend/abort process.
- Abort Output (AO)
 - process can complete, but send no more output to user's terminal.
- Are You There (AYT)
 - check to see if system is still running.

More Control Functions

- Erase Character (EC)
 - delete last character sent
 - typically used to edit keyboard input.
- Erase Line (EL)
 - delete all input in current line.

Command Structure

- All TELNET commands and data flow through the same TCP connection.
- Commands start with a special character called the Interpret as Command escape character (IAC).
- The IAC code is 255.
- If a 255 is sent as data - it must be followed by another 255.

Looking for Commands

- Each receiver must look at each byte that arrives and look for IAC.
- If IAC is found and the next byte is IAC - a single byte is presented to the application/terminal.
- If IAC is followed by any other code - the TELNET layer interprets this as a command.

SECURE SHELL

- ‘*Secure shell is a de facto standard for remote logins and encrypted file transfers.*’ [SSH communications inc.]
- Founded in 1995 by Tatu Ylonen, a researcher at Helsinki University of Technology, Finland
- It provides authentication and encryption for business critical applications to work securely over the internet.
- Originally introduced for UNIX terminals as a replacement for the insecure remote access “Berkeley services” , viz. rsh, rlogin, rcp, telnet, etc.
- It can also be used for port forwarding of arbitrary TCP/IP or X11 connections (interactive terminals)
- It is a layer over TCP/IP and runs on the port 22.

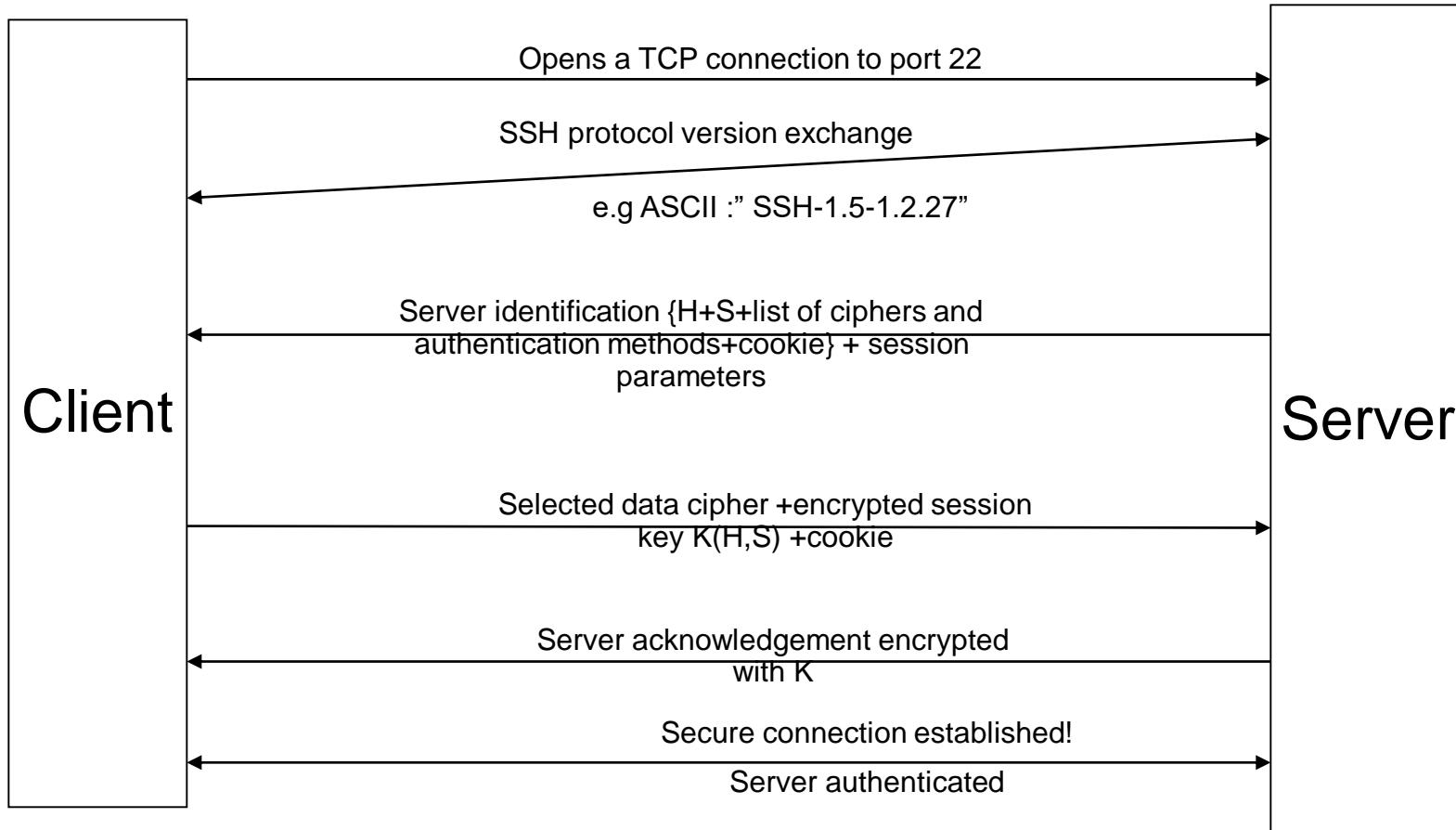
Why SSH?

- The three core security requirements for a remote access technology – **confidentiality, integrity** and authentication
- Most of the earlier technologies lack **confidentiality** and **integrity**. For e.g Telnet and FTP transmit username and passwords in cleartext.
- They are vulnerable to attacks such as IP spoofing, DoS, MITM and eavesdropping.
- Secure shell satisfies all the three requirements by using:
 - ❖ Data Encryption to provide confidentiality
 - ❖ Host-based and (or) client-based authentication
 - ❖ Data integrity using MACs and hashes

Flavors of SSH

- There are two incompatible versions of the SSH protocol: SSH-1 and SSH-2
- SSH-2 was introduced in order to fix several bugs in SSH-1 and also includes several new features
- Both versions have technical glitches and are vulnerable to known attacks (discussed later)
- SSH-1 is more popular nowadays due to licensing issues with SSH-2
- OpenSSH is a free version of the SSH with open source code development. It supports both the versions and mainly used in UNIX environment.

The SSH-1 protocol exchange



H – host key ; S- Server Key ; cookie- sequence of 8 random bytes; K- session key

The SSH-1 protocol exchange (contd.)

- Once secure connection is established, client attempts to authenticate itself to server.
- Some of the authentication methods used are:
 - Kerberos
 - RHosts and RHostsRSA
 - Public key
 - Password based authentication (OTP)
- Integrity checking is provided by means of a weak CRC -32
- Compression is provided using the “deflate” algorithm of GNU *gzip* utility. It is beneficial for file transfer utilities such as *ftp*, *scp*, etc.

The SSH-2 protocol

- SSH-1 is monolithic, encompassing multiple functions into a single protocol whereas SSH-2 has been separated into modules and consists of 3 protocols:
 - **SSH Transport layer protocol (SSH-TRANS)** : Provides the initial connection, packet protocol, server authentication and basic encryption and integrity services.
 - **SSH Authentication protocol (SSH-AUTH)** : Verifies the client's identity at the beginning of an SSH-2 session, by three authentication methods: Public key, host based and password.
 - **SSH Connection protocol (SSH-CONN)** : It permits a number of different services to exchange data through the secure channel provided by SSH-TRANS.
- A fourth protocol SSH protocol architecture (SSH-ARCH) describes the overall architecture.
- All the protocols are still in the draft stage.

SSH-1 vs SSH-2

SSH-2	SSH-1
Separate transport, authentication and connection protocols	One monolithic protocol
Strong cryptographic integrity check	Weak CRC-32 integrity check
No server keys needed due to Diffie Hellman Key exchange	Server key used for forward secrecy on the session key
Supports public key certificates	N/A
Algorithms used: DSA, DH, SHA-1, MD5, 3DES, Blowfish, Twofish, CAST-128, IDEA, ARCFour	RSA, MD5, CRC-32, 3DES, IDEA, ARCFour, DES
Periodic replacement of session keys	N/A

Is SSH really secure?

- Secure shell does counter certain types of attacks such as:
 - Eavesdropping
 - Name service and IP spoofing
 - Connection hijacking
 - MITM
 - Insertion attack
- However it fails against these attacks:
 - Password cracking
 - IP and TCP attacks
 - *SYN flooding*
 - *TCP RST, bogus ICMP*
 - *TCP desynchronization and hijacking*
 - Traffic analysis
 - Covert channels

Some known vulnerabilities in SSH

- OpenSSH Challenge-Response Authentication Buffer Overflow: A hostile modification to the SSH client floods the server with authentication responses and causes a buffer overflow. [SSH-2]
- Passive analysis of SSH traffic: It lets the attacker obtain sensitive information by passively monitoring encrypted SSH sessions. The information can later be used to speed up brute-force attacks on passwords, including the initial login password and other passwords appearing in interactive SSH sessions. [SSH-1 & SSH-2]
- Key recovery in SSH protocol 1.5 : This vulnerability may lead to the compromise of the session key. Once the session key is determined, the attacker can proceed to decrypt the stored session using any implementation of the crypto algorithm used. This will reveal all information in an unencrypted form. [SSH-1]
- CRC-32 integrity check vulnerability : It is based on the weakness of the CRC-32 integrity that becomes exploitable due to CBC and CFB feedback modes [SSH-1]

SSH tools

- There are several commercial and freeware SSH tools for UNIX and windows platforms
- Windows –
 - PuTTY
 - TeraTerm
 - SecureCRT
- UNIX –
 - The RedHat Linux package and OpenBSD come alongwith tools such as ssh, sshd(daemon), scp(secure copy), sftp(secure ftp), etc.
 - OpenSSH

DNS

Domain Name System

- How do we efficiently locate resources?
 - DNS: name → IP address
- Challenge
 - How do we scale these to the wide area?

Obvious Solutions (1)

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Single point of update
- Doesn't *scale!*

Obvious Solutions (2)

Why not use /etc/hosts?

- Original Name to Address Mapping
 - Flat namespace
 - /etc/hosts
 - SRI kept main copy
 - Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
 - Many more downloads
 - Many more updates

Domain Name System Goals

- Basically a wide-area distributed database
- Scalability
- Decentralized maintenance
- Robustness
- Global scope
 - Names mean the same thing everywhere
- Don't need
 - Atomicity
 - Strong consistency

Programmer's View of DNS

- Conceptually, programmers can view the DNS database as a collection of millions of *host entry structures*:

```
/* DNS host entry structure */
struct hostent {
    char    *h_name;          /* official domain name of host */
    char    **h_aliases;      /* null-terminated array of domain names */
    int     h_addrtype;       /* host address type (AF_INET) */
    int     h_length;         /* length of an address, in bytes */
    char    **h_addr_list;    /* null-terminated array of in_addr structs */
};
```

- `in_addr` is a struct consisting of 4-byte IP address
- Functions for retrieving host entries from DNS:
 - `gethostbyname`: query key is a DNS host name.
 - `gethostbyaddr`: query key is an IP address.

DNS Message Format

12 bytes	Identification	Flags
	No. of Questions	No. of Answer RRs
	No. of Authority RRs	No. of Additional RRs
Name, type fields for a query	Questions (variable number of answers)	
RRs in response to query	Answers (variable number of resource records)	
Records for authoritative servers	Authority (variable number of resource records)	
Additional “helpful info that may be used”	Additional Info (variable number of resource records)	

DNS Header Fields

- Identification
 - Used to match up request/response
- Flags
 - 1-bit to mark query or response
 - 1-bit to mark authoritative or not
 - 1-bit to request recursive resolution
 - 1-bit to indicate support for recursive resolution

DNS Records

RR format: (**class**, **name**, **value**, **type**, **ttl**)

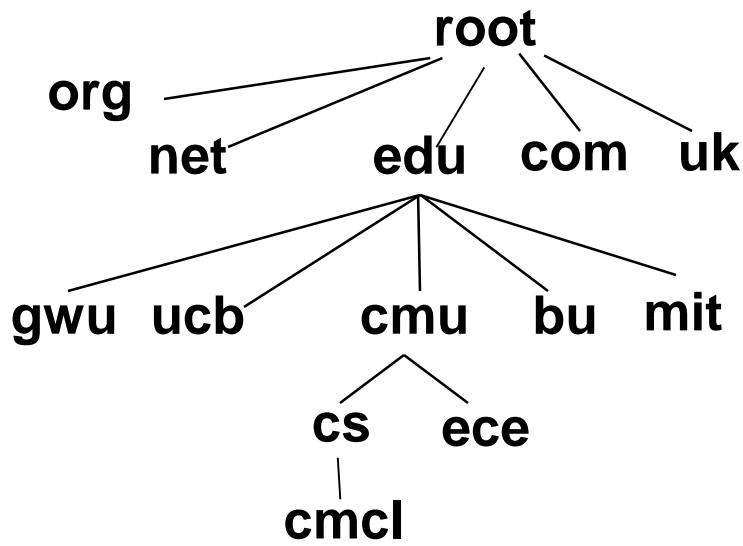
- DB contains tuples called resource records (RRs)
 - Classes = Internet (IN), Chaosnet (CH), etc.
 - Each class defines value associated with type

- Type=CNAME
- **name** is an alias name for some “canonical” (the real) name
 - **value** is canonical name
 - Type=MX
- **value** is hostname of mailserver associated with **name**

Properties of DNS Host Entries

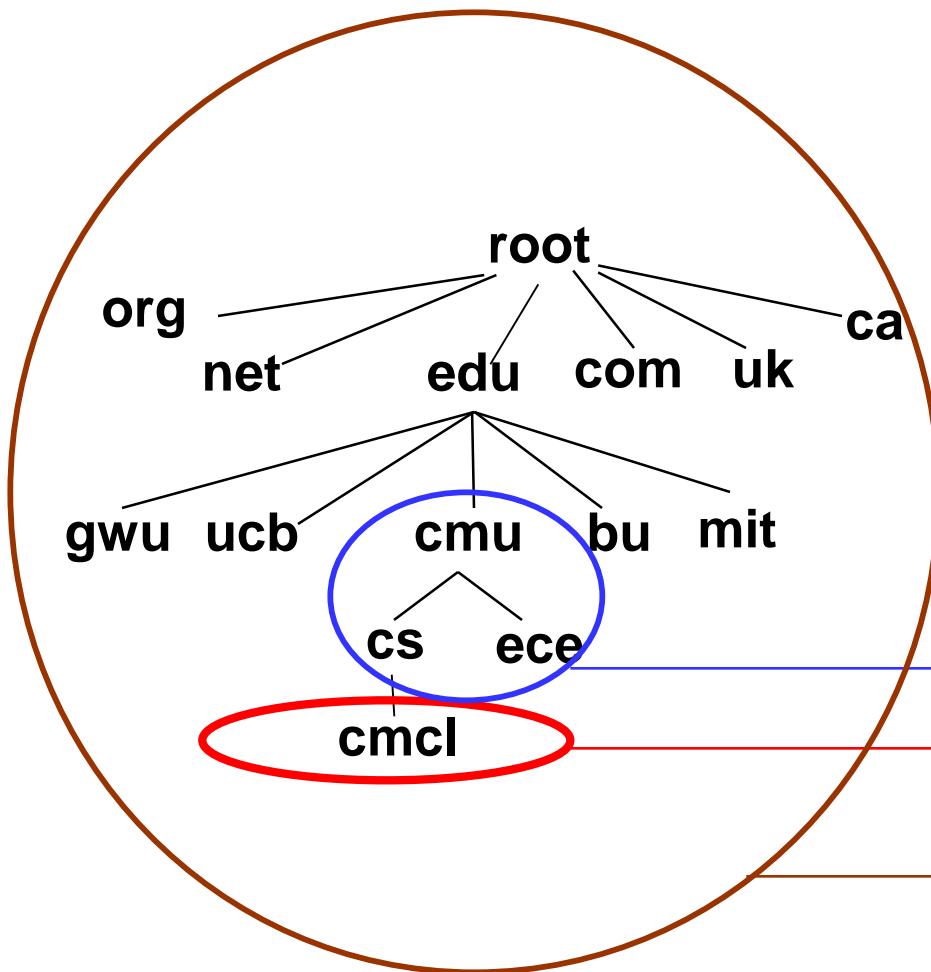
- Different kinds of mappings are possible:
 - Simple case: 1-1 mapping between domain name and IP addr:
 - kittyhawk.cmcl.cs.cmu.edu maps to 128.2.194.242
 - Multiple domain names maps to the same IP address:
 - eecs.mit.edu and cs.mit.edu both map to 18.62.1.6
 - Single domain name maps to multiple IP addresses:
 - aol.com and www.aol.com map to multiple IP addrs.
 - Some valid domain names don't map to any IP address:
 - for example: cmcl.cs.cmu.edu

DNS Design: Hierarchy Definitions



- Each node in hierarchy stores a list of names that end with same suffix
 - Suffix = path up tree
- E.g., given this tree, where would following be stored:
 - Fred.com
 - Fred.edu
 - Fred.cmu.edu
 - Fred.cmcl.cs.cmu.edu
 - Fred.cs.mit.edu

DNS Design: Zone Definitions



- Zone = contiguous section of name space
 - E.g., Complete tree, single node or subtree
- A zone has an associated set of name servers
 - Must store list of names and tree links

Subtree

Single node

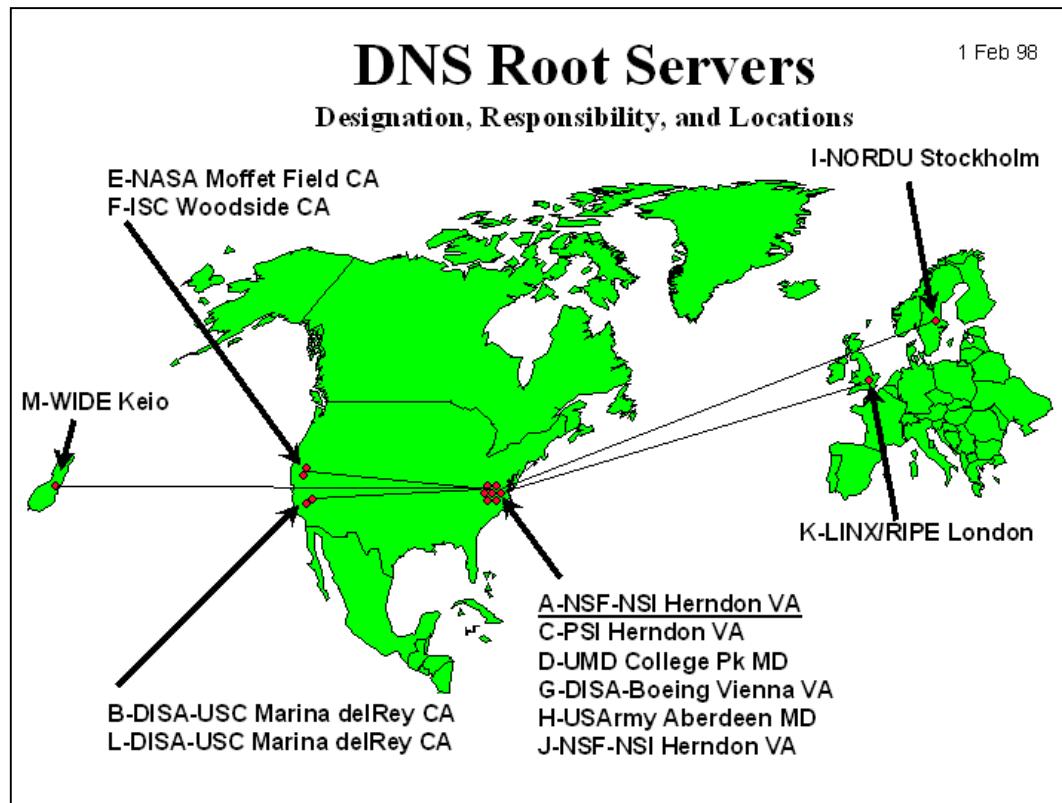
Complete
Tree

DNS Design: Cont.

- Zones are created by convincing owner node to create/delegate a subzone
 - Records within zone stored multiple redundant name servers
 - Primary/master name server updated manually
 - Secondary/redundant servers updated by zone transfer of name space
 - Zone transfer is a bulk transfer of the “configuration” of a DNS server – uses TCP to ensure reliability
- Example:
 - CS.CMU.EDU created by CMU.EDU administrators
 - Who creates CMU.EDU or .EDU?

DNS: Root Name Servers

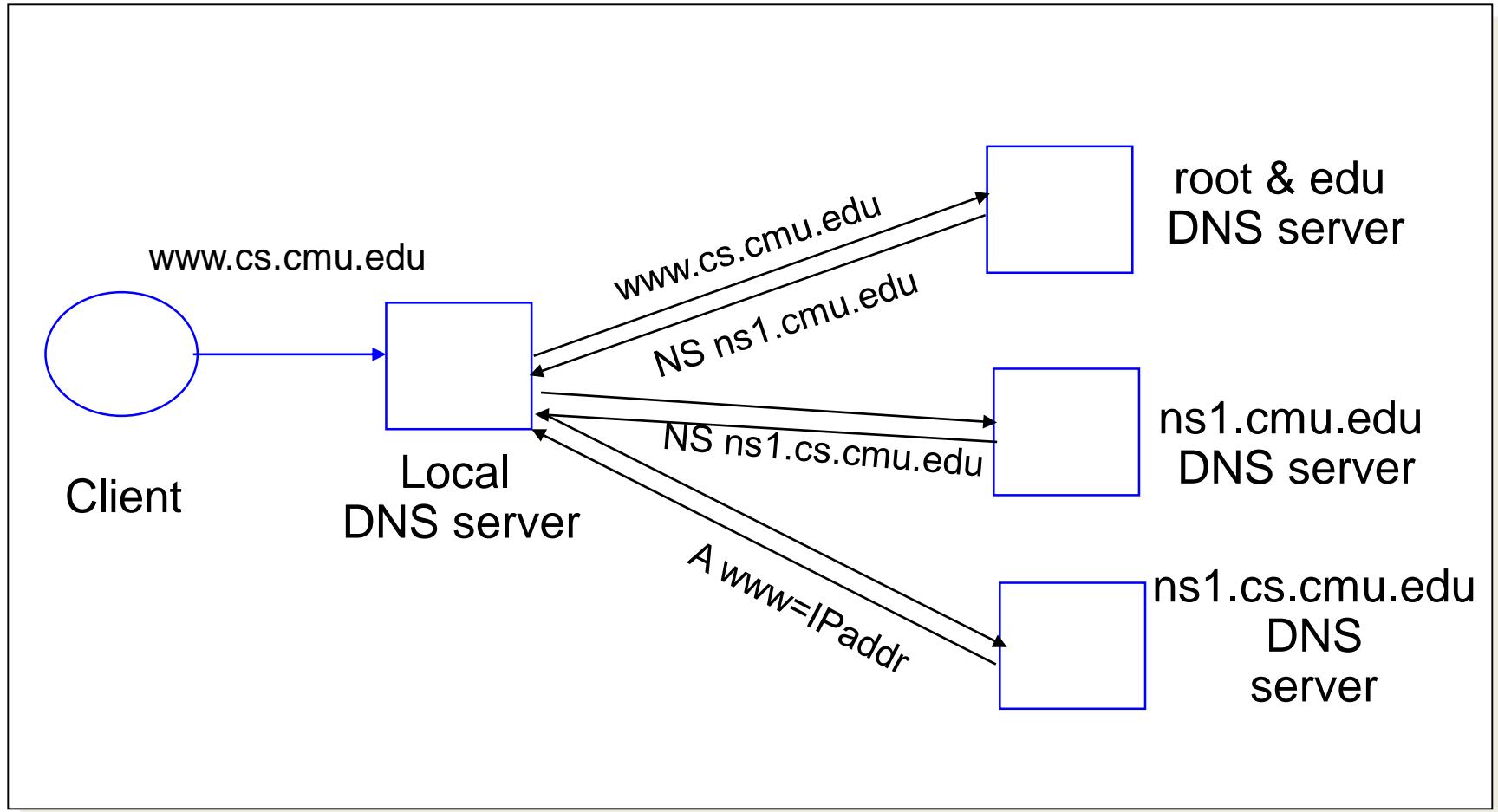
- Responsible for “root” zone
- Approx. 13 root name servers worldwide
 - Currently {a-m}.root-servers.net
- Local name servers contact root servers when they cannot resolve a name
 - Configured with well-known root servers



Servers/Resolvers

- Each host has a resolver
 - Typically a library that applications can link to
 - Local name servers hand-configured (e.g. /etc/resolv.conf)
- Name servers
 - Either responsible for some zone or...
 - Local servers
 - Do lookup of distant host names for local hosts
 - Typically answer queries about local zone

Typical Resolution



Typical Resolution

- Steps for resolving `www.cmu.edu`
 - Application calls `gethostbyname()` (RESOLVER)
 - Resolver contacts local name server (S_1)
 - S_1 queries root server (S_2) for (www.cmu.edu)
 - S_2 returns NS record for `cmu.edu` (S_3)
 - What about A record for S_3 ?
 - This is what the additional information section is for (PREFETCHING)
 - S_1 queries S_3 for www.cmu.edu
 - S_3 returns A record for www.cmu.edu
- Can return multiple A records → what does this mean?

Lookup Methods

Recursive query:

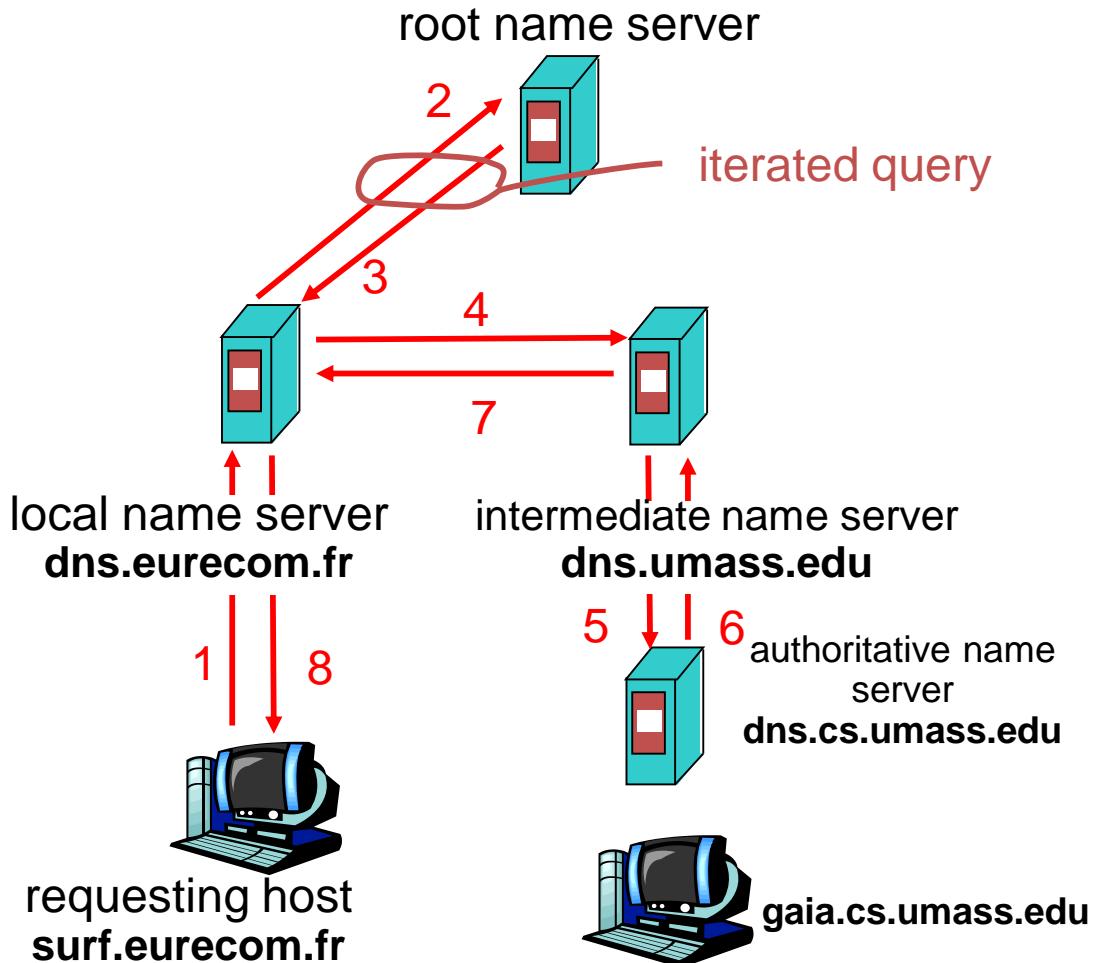
- Server goes out and searches for more info (recursive)
- Only returns final answer or “not found”

Iterative query:

- Server responds with as much as it knows (iterative)
- “I don’t know this name, but ask this server”

Workload impact on choice?

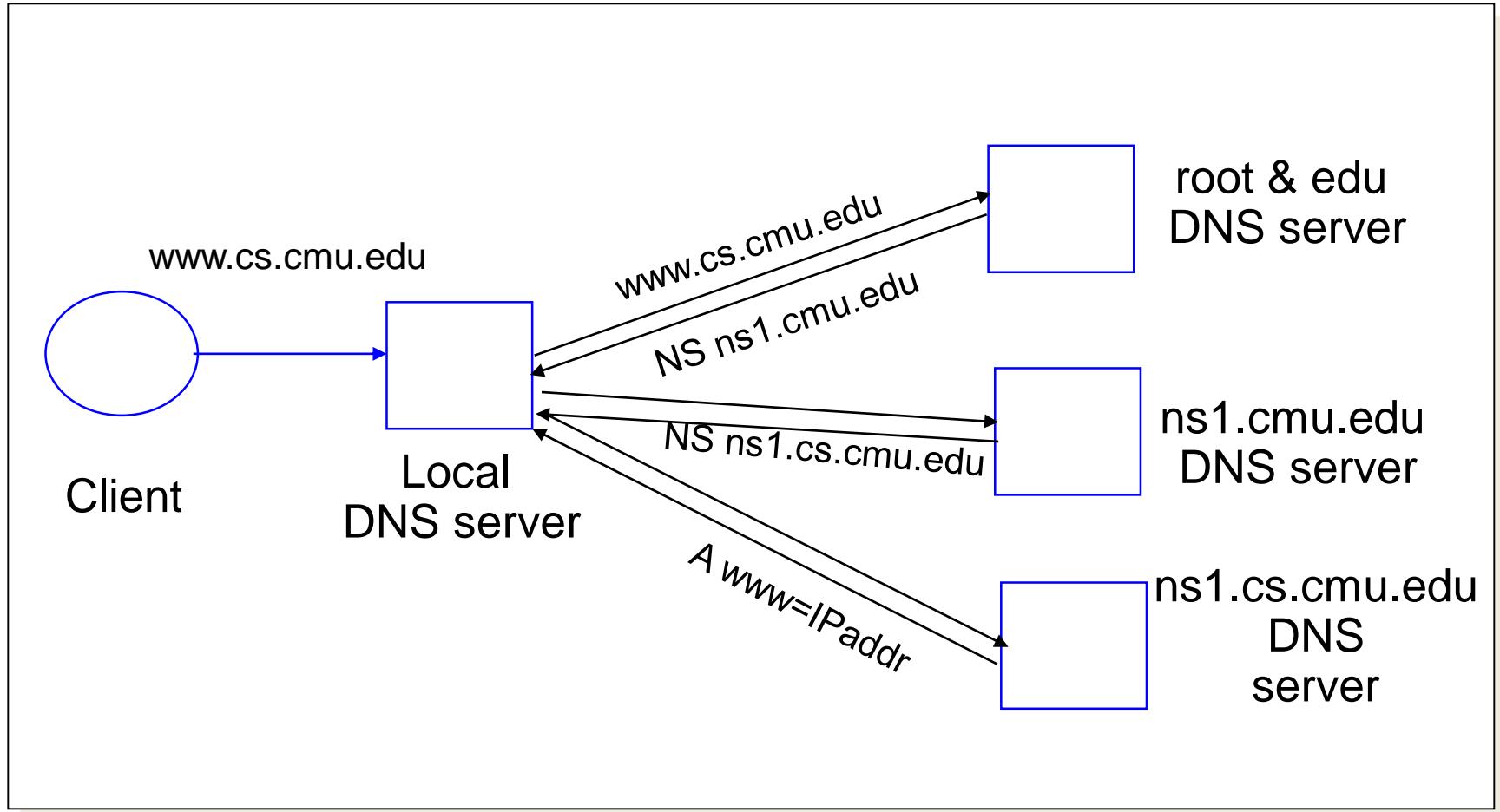
- Local server typically does recursive
- Root/distant server does iterative



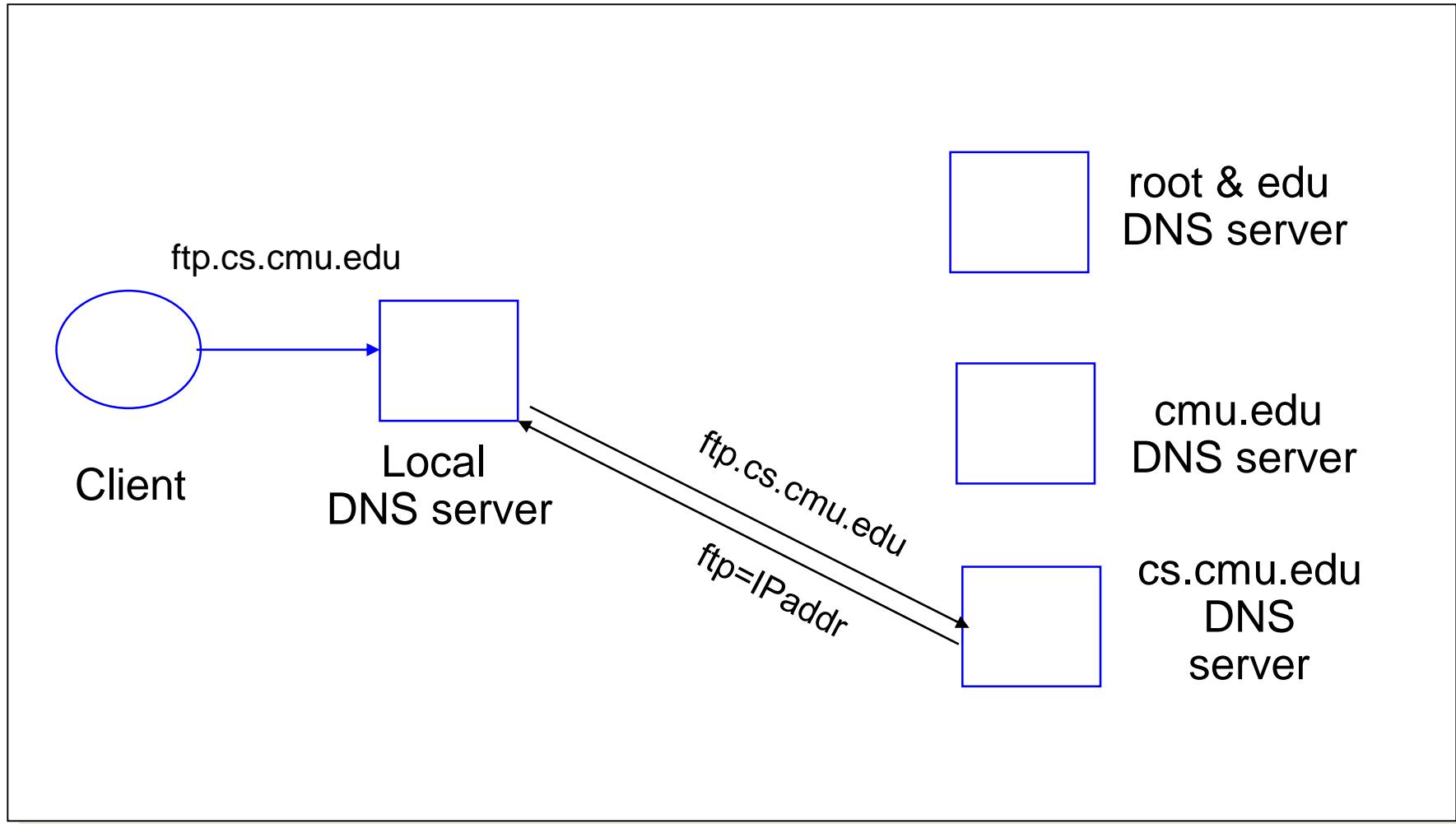
Workload and Caching

- Are all servers/names likely to be equally popular?
 - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
 - Quick response for repeated translations
 - Other queries may reuse some parts of lookup
 - NS records for domains
- DNS negative queries are cached
 - Don't have to repeat past mistakes
 - E.g. misspellings, search strings in resolv.conf
- Cached data periodically times out
 - Lifetime (TTL) of data controlled by owner of data
 - TTL passed with every record

Typical Resolution



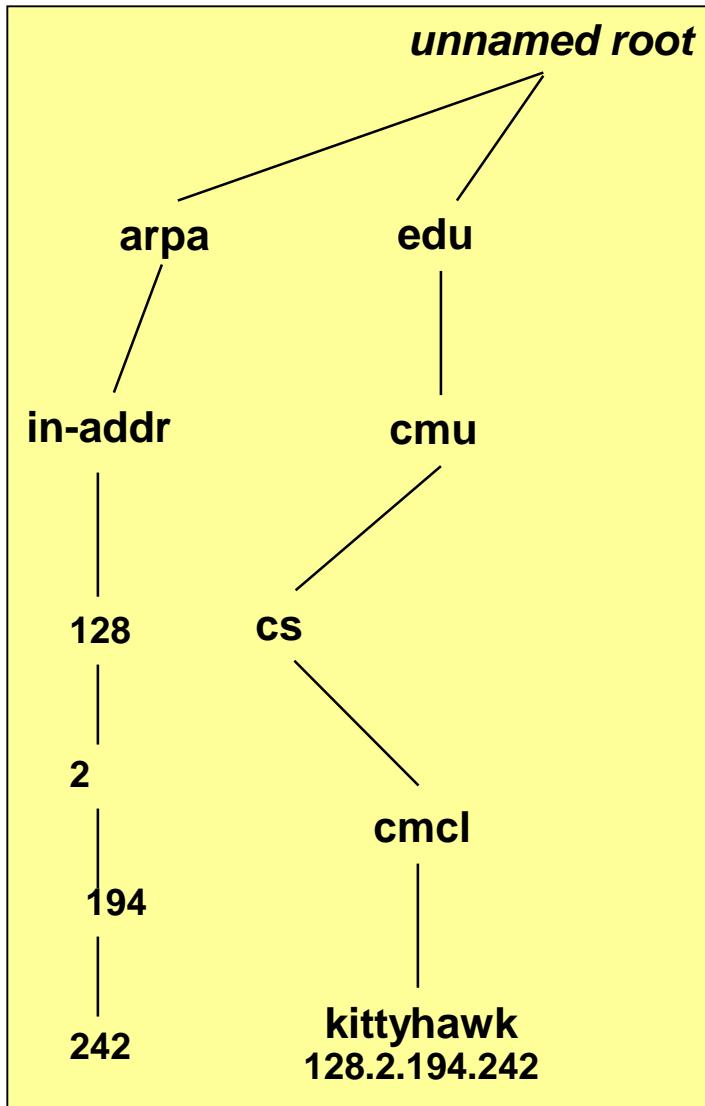
Subsequent Lookup Example



Reliability

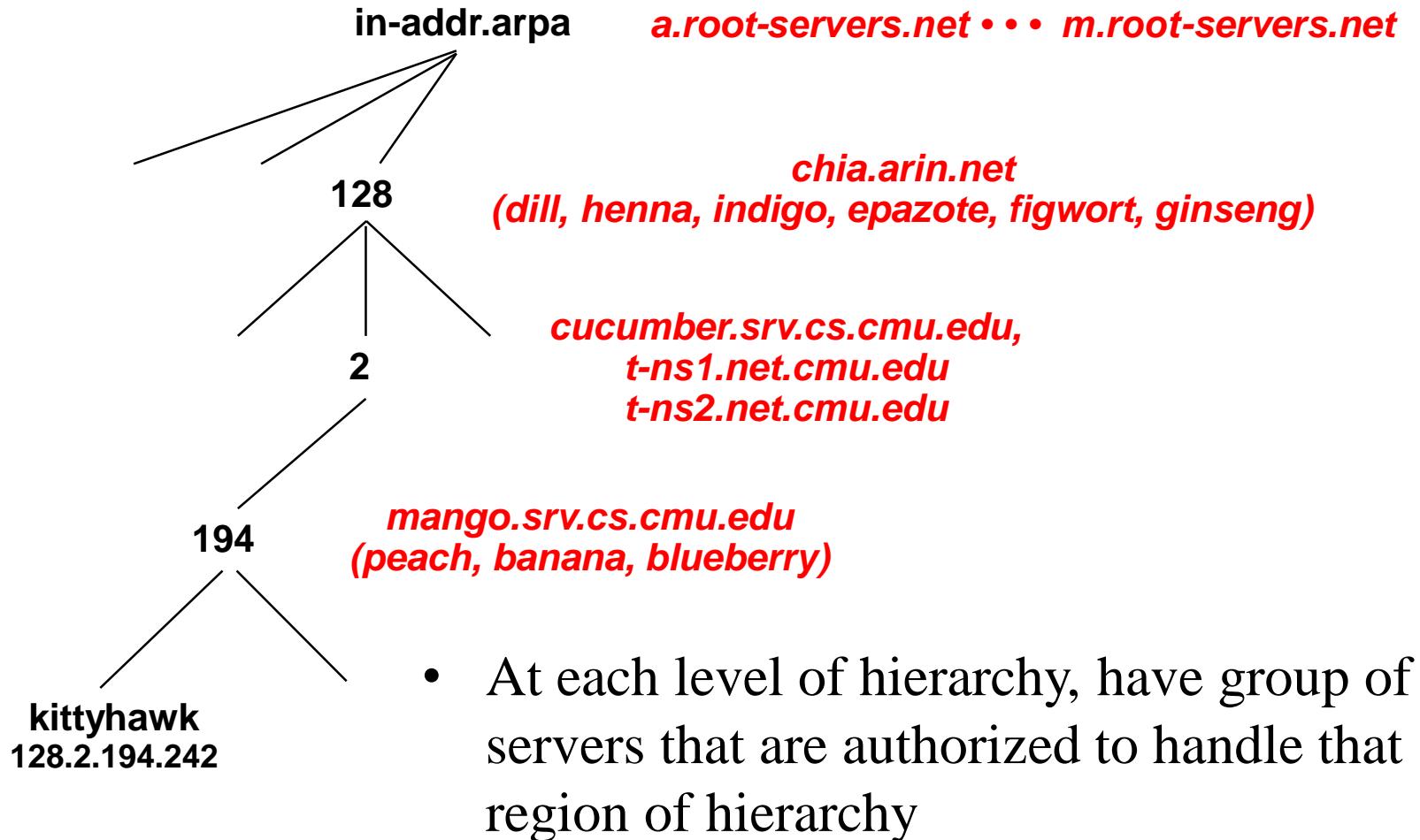
- DNS servers are replicated
 - Name service available if \geq one replica is up
 - Queries can be load balanced between replicas
- UDP used for queries
 - Need reliability → must implement this on top of UDP!
 - Why not just use TCP?
- Try alternate servers on timeout
 - Exponential backoff when retrying same server
- Same identifier for all queries
 - Don't care which server responds

Reverse DNS



- Task
 - Given IP address, find its name
- Method
 - Maintain separate hierarchy based on IP names
 - Write 128.2.194.242 as 242.194.128.2.in-addr.arpa
 - Why is the address reversed?
- Managing
 - Authority manages IP addresses assigned to it
 - E.g., CMU manages name space 128.2.in-addr.arpa

.arpa Name Server Hierarchy



Prefetching

- Name servers can add additional data to response
- Typically used for prefetching
 - CNAME/MX/NS typically point to another host name
 - Responses include address of host referred to in “additional section”

Root Zone

- Generic Top Level Domains (gTLD) = .com, .net, .org, etc...
- Country Code Top Level Domain (ccTLD) = .us, .ca, .fi, .uk, etc...
- Root server ({a-m}.root-servers.net) also used to cover gTLD domains
 - Load on root servers was growing quickly!
 - Moving .com, .net, .org off root servers was clearly necessary to reduce load → done Aug 2000

New gTLDs

- .info → general info
- .biz → businesses
- .aero → air-transport industry
- .coop → business cooperatives
- .name → individuals
- .pro → accountants, lawyers, and physicians
- .museum → museums
- Only new one active so far = .info, .biz, .name

SNMP

INTRODUCTION

- SNMP
- What is SNMP?
- SNMP Architecture
- SNMP Components
- Versions of SNMP
- Trap Event

WHAT IS SNMP?

- **Simple Network Management Protocol (SNMP)** is an Internet standard protocol
- It is defined by IETF, Internet engineering task force. It is application layer protocol.
- It manages devices on IP networks. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more. It is used mostly in network management.
- SNMP is an application program that allows, manager to retrieve value of an object defined in agent, a manager to store value in an object defined in agent , an agent to send alarm information called trap event.

SNMP ARCHITECTURE

- Manager
 - Agent
- NMS(network management Systems)

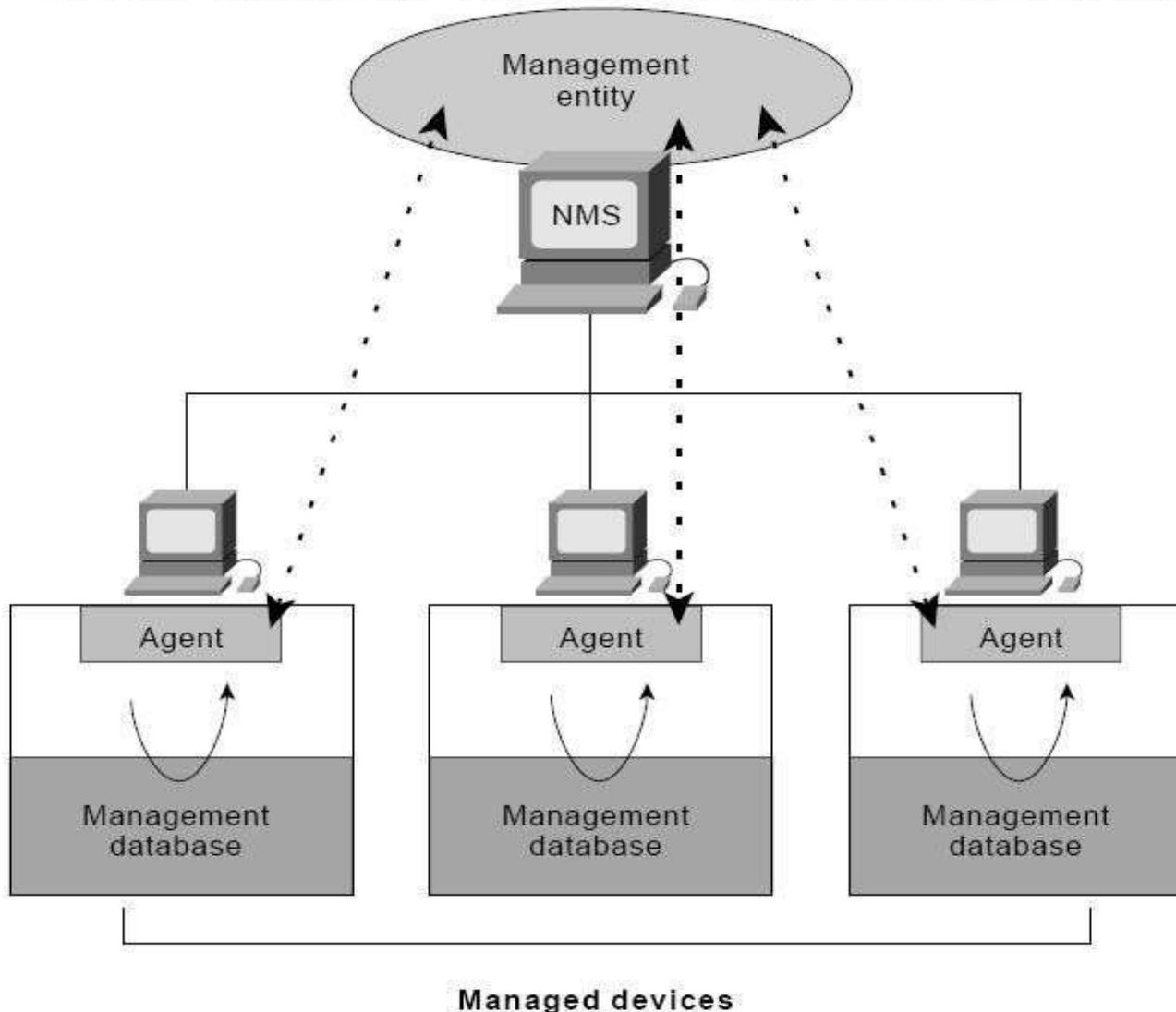
SNMP ARCHITECTURE

- SNMP defines **manager** ,it monitoring and managing devices connected with network.
- Manager sending request to network device called **agent** and **agent** will respond to manager request via SNMP.
- Agents will respond data on managed systems as variables.

SNMP ARCHITECTURE

- NMS, network management systems it executes application which monitor and manage devices.
basically it will run on manager

An SNMP-Managed Network Consists of Managed Devices, Agents, and NMSs

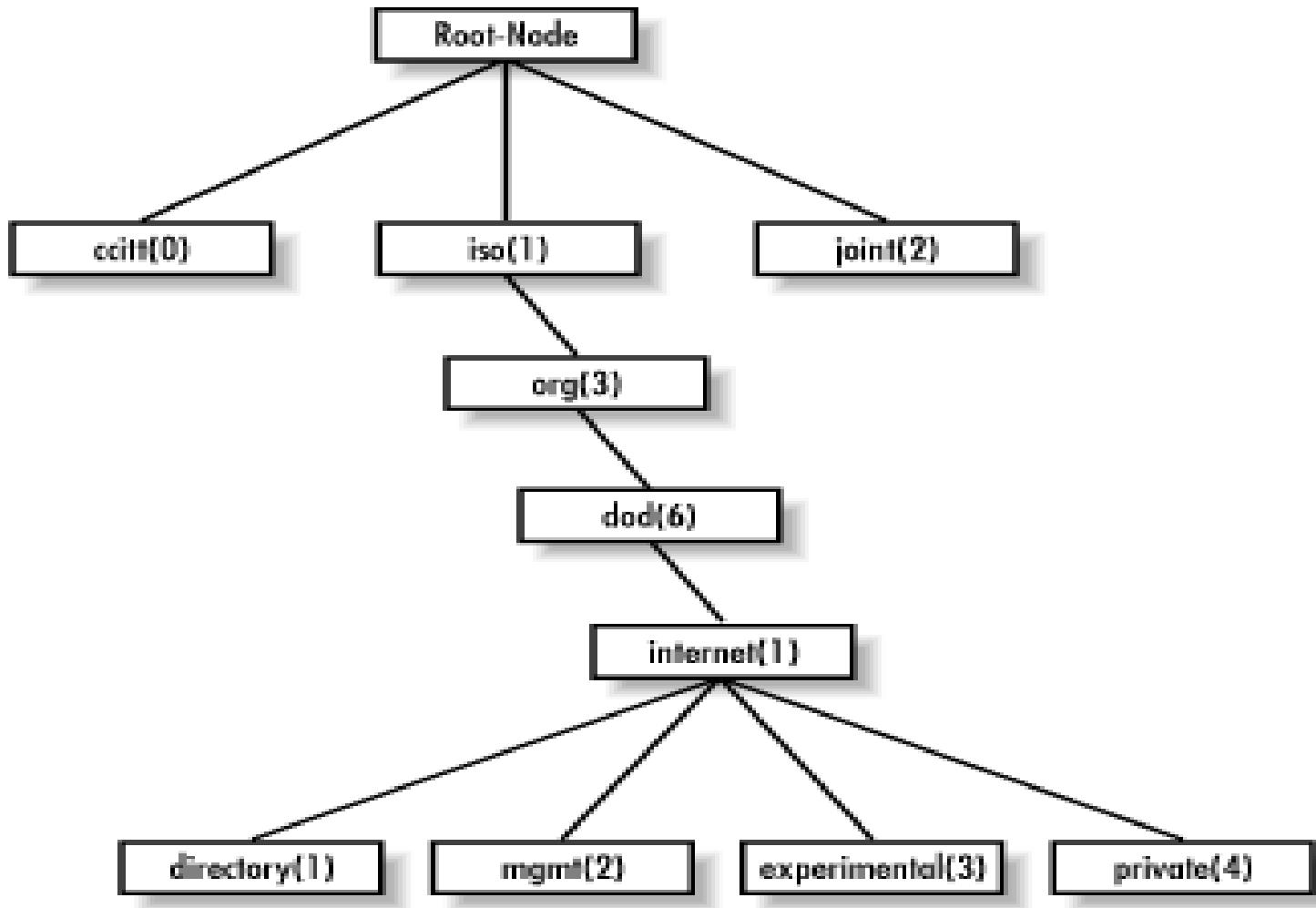


SNMP COMPONENTS

- ‘ Management information base (MIB)
- ‘ Structure of management information (SMI)

MIB

- SNMP protocol provide information about devices and that variables information is provided by management information base(MIB). It is collection of objects and their types in hierachal tree format.



SMI

- SMI defines rule for naming objects, defining object types and showing how to encode objects and data.
- Agent will receives request on UDP port no. 161 and manager will send request on UDP port no. 161
- Manager receives notifications from agent on port no. 162 that called trap event.

SNMP VERSIONS & PDU

- SNMPv1
- SNMPv2 (SNMPv2P,SNMPv2C,SNMPv2u)

SNMP has protocol data units(PDUs),

- GetRequest SetRequest
- Getnextrequest
- GetBulkrequest (SNMPv2)
- Response
- Trap
- InformRequest(SNMPv2)
-

- SNMPv1

V1 is first version of SNMP basically has poor security.

- SNMPv2

V2 is revised version of V1 it improves in area of performance, security, confidentiality. It introduces Getbulkrequest & inform Request

SNMP MESSAGE FIELDS

Field	Description
version	SNMP version;
community	A pairing of an SNMP agent with some arbitrary set of SNMP application entities. The name of the community functions as a password to authenticate the SNMP message.
request-id	Used to distinguish among outstanding requests by providing each request with a unique ID.
error-status	Used to indicate that an exception occurred while processing a request. Values are: noError (0), tooBig (1), noSuchName (2), badValue (3), readOnly (4), genErr (5)
error-index	When error-status is nonzero, error-index may provide additional information by indicating which variable in a list caused the exception. A variable is an instance of a managed object.
variable-bindings	A list of variable names and corresponding values. In some cases (e.g., GetRequest-PDU, the values are null.

Get Request

Manager to agent request to retrieve the value of variable. Agent will respond with requested variable with current stored value.

SetRequest

Manager to agent request to change value of variable. Variable bindings are defined in request. Agent will respond with new value.

GetNextRequest

Manager to agent request to discover available variables and their values. Agent will respond with value of next variable.

GetBulkRequest

it request multiple iterations of GetNextRequest. Returns the response with multiple variable bindings in request.

Response

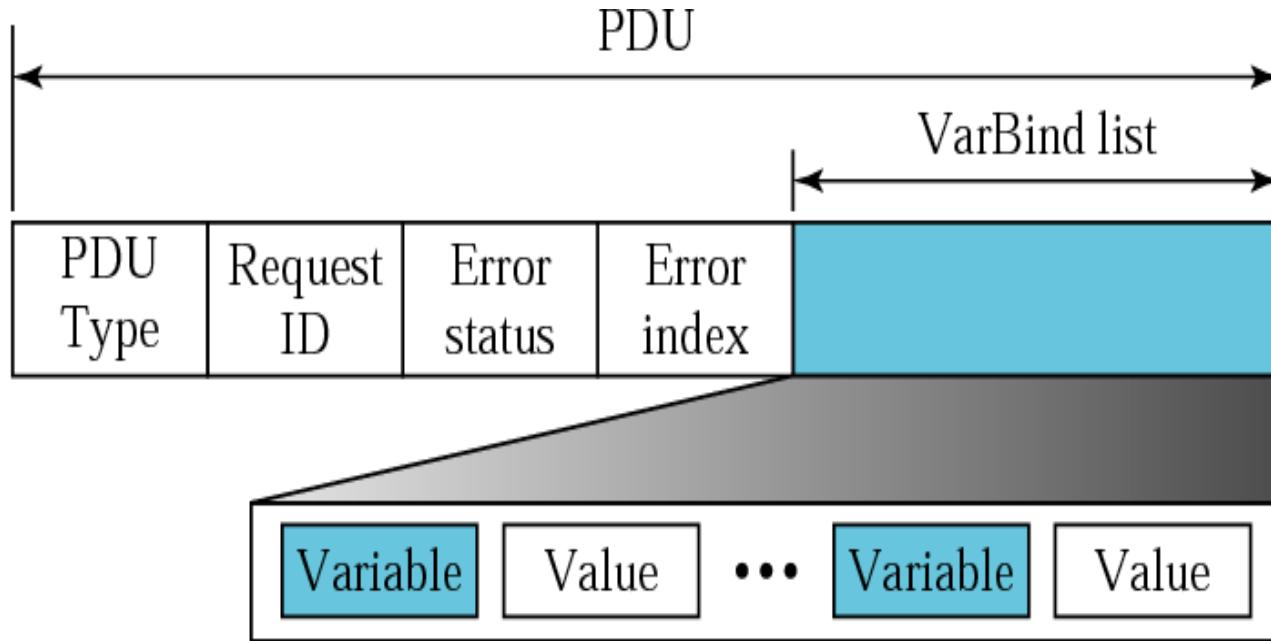
returns value as requested from agent to manager. It used as response to set and get request.

Trap

It is an notification event , agent to manager which is not requested by manager. Agent itself informing to manager.

Inform request

It is manager to manager communication, one manager can send some information to another manager using informRequest PDU receiving manager response with Response PDU to manager confirming receipt of manager



Differences:

1. Error status and error index values are zeros for all request messages except GetBulkRequest.
2. Error status field is replaced by non-repeater field and error index field is replaced by max-repetitions field in GetBulkRequest.

SNMPv2 error-status	SNMPv1 error-status
noError	noError
tooBig	tooBig
genErr	genErr
wrongValue	badValue
wrongEncoding	badValue
wrongType	badValue
wrongLength	noSuchName
inconsistentValue	noSuchName
noAccess	noSuchName
notWritable	noSuchName
noCreation	genErr
inconsistentName	genErr
resourceUnavailable	noSuchName
commitFailed	
undoFailed	
authorizationError	

TRAP-PDU

enterprise	Type of object generating trap; based on sysObjectID.
agent-addr	Address of object generating trap.
generic-trap	Generic trap type. Values are: coldStart (0), warmStart (1), linkDown (2), linkUp (3), authenticationFailure (4), egpNeighborLoss (5), enterpriseSpecific (6).
specific-trap	Specific trap code.
time-stamp	Time elapsed between the last (re)initialization of the network entity and the generation of the trap; contains the value of sysUpTime.

- A sysUpTime parameter (TimeTicks). This appears in the first variable-binding in an SNMPv2-Trap-PDU or InformRequest-PDU.
- An snmpTrapOID parameter (OBJECT IDENTIFIER).
- This appears in the second variable-binding in an SNMPv2-Trap-PDU or InformRequest-PDU.
- A list of variable-bindings (VarBindList). This refers to all but the first two variable-bindings in an SNMPv2-Trap- PDU or InformRequest-PDU

- **TRAPV2**
 1. Contains agent address.
 2. It has information about specific trap and generic trap value.
 3. It does not have error index and status
- **TRAPV2**
 1. Does not contain agent address.
 2. It has Trap OID in the second varbind.
 3. It has error index and status.