

# **COMPUTER NETWORKS**

**Course NOTES**

**With ZeroCost Academy**

## **Computer Networks**

### **Objectives:**

- To introduce the fundamental types of computer networks.
- To demonstrate the TCP/IP & OSI model merits & demerits.
- To know the role of various protocols in Networking

### **UNIT - I:**

Introduction: Network, Uses of Networks, Types of Networks, Reference Models: TCP/IP Model, The OSI Model, Comparison of the OSI and TCP/IP reference model. Architecture of Internet.

Physical Layer: Guided transmission media, Wireless transmission media, Switching

### **UNIT - II:**

Data Link Layer - Design issues, Error Detection & Correction, Elementary Data Link Layer Protocols, Sliding window protocols

Multiple Access Protocols - ALOHA, CSMA,CSMA/CD, CSMA/CA, Collision free protocols, Ethernet- Physical Layer, Ethernet Mac Sub layer, Data link layer switching: Use of bridges, learning bridges, spanning tree bridges, repeaters, hubs, bridges, switches, routers and gateways.

### **UNIT - III:**

Network Layer: Network Layer Design issues, store and forward packet switching connection less and connection oriented networks-routing algorithms-optimality principle, shortest path, flooding, Distance Vector Routing, Count to Infinity Problem, Link State Routing, Path Vector Routing, Hierarchical Routing; Congestion control algorithms, IP addresses, CIDR, Subnetting, SuperNetting, IPv4, Packet Fragmentation, IPv6 Protocol, Transition from IPv4 to IPv6, ARP, RARP.

### **UNIT - IV:**

Transport Layer: Services provided to the upper layers elements of transport protocol

**REFERENCES BOOKS:**

1. An Engineering Approach to Computer Networks - S. Keshav, 2nd Edition, Pearson Education.
2. Understanding communications and Networks, 3rd Edition, W. A. Shay, Cengage Learning.
3. Computer Networking: A Top-Down Approach Featuring the Internet, James F. Kurose, K. W. Ross, 3rd Edition, Pearson Education.

**Outcomes:**

- Students should be understand and explore the basics of Computer Networks and Various Protocols.
- Student will be in a position to understand the World Wide Web concepts.
- Students will be in a position to administrate a network and flow of information further
- Student can understand easily the concepts of network security, Mobile

## INDEX

UNIT NO	TOPIC	PAGE NO
I	INTRODUCTION TO NETOWRKS	1
	TYPES OF NETWORKS,	4
	INTRODUCTION TO PHYSICAL LAYER	18
	COMPARISON OF OSI AND TCP/IP PROTOCOLS	25
II	DATA LINK LAYER DESIGN ISSUES	35
	SLIDING WINDOW PROTOCOLS	41
	MULTIPLE ACCESS PROTOCOLS	49
III	NETWORK LAYER DESIGN ISSUES	78
	CONNECTION LESS AND CONNECTION ORIENTED PROTOCOLS	80
	ROUTING PROTOCOLS,IP ADDRESS	81
IV	TRANSPORT LAYER SERVICES PROVIDED	101
	THE INTERNET TRANSPORT PROTOCOLS	130
V	APPLICATION LAYER SERVICES	249
	APPLICATIONS LAYER PARADIGMS	256

# **UNIT - I**

## **NETWORKS**

A network is a set of devices (often referred to as *nodes*) connected by communication links. A node can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network.

“Computer network” to mean a collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information.

The connection need not be via a copper wire; fiber optics, microwaves, infrared, and communication satellites can also be used.

Networks come in many sizes, shapes and forms, as we will see later. They are usually connected together to make larger networks, with the **Internet** being the most well-known example of a network of networks.

There is considerable confusion in the literature between a **computer network** and a **distributed system**. The key distinction is that in a distributed system, a collection of independent computers appears to its users as a single coherent system. Usually, it has a single model or paradigm that it presents to the users. Often a layer of software on top of the operating system, called **middleware**, is responsible for implementing this model. A well-known example of a distributed system is the **World Wide Web**. It runs on top of the Internet and presents a model in which everything looks like a document (Web page).

## **USES OF COMPUTER NETWORKS**

### **1. Business Applications**

- to distribute information throughout the company (**resource sharing**). sharing physical resources such as printers, and tape backup systems, is sharing information
- **client-server model**. It is widely used and forms the basis of much network

- electronic commerce
- entertainment.(game playing,)

### **3 Mobile Users**

- Text messaging or texting
- Smart phones,
- GPS (Global Positioning System)
- m-commerce
- NFC (Near Field Communication)

### **4 Social Issues**

With the good comes the bad, as this new-found freedom brings with it many unsolved social, political, and ethical issues.

Social networks, message boards, content sharing sites, and a host of other applications allow people to share their views with like-minded individuals. As long as the subjects are restricted to technical topics or hobbies like gardening, not too many problems will arise.

The trouble comes with topics that people actually care about, like politics, religion, or sex. Views that are publicly posted may be deeply offensive to some people. Worse yet, they may not be politically correct. Furthermore, opinions need not be limited to text; high-resolution color photographs and video clips are easily shared over computer networks. Some people take a live-and-let-live view, but others feel that posting certain material (e.g., verbal attacks on particular countries or religions, pornography, etc.) is simply unacceptable and that such content must be censored. Different countries have different and conflicting laws in this area. Thus, the debate rages.

Computer networks make it very easy to communicate. They also make it easy for the people who run the network to snoop on the traffic. This sets up conflicts over issues such as **employee rights versus employer rights**. Many people read and write email at work. Many employers have claimed the right to read and possibly censor employee messages, including messages sent

**BOTNET ATTACK:** Botnets can be used to perform [distributed denial-of-service attack](#) (DDoS attack), steal data, send spam, and allows the attacker to access the device and its connection.

---

The effectiveness of a data communications system depends on four fundamental characteristics: delivery, accuracy, timeliness, and jitter.

1. **Delivery.** The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.

2 **Accuracy.** The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable.

3. **Timeliness.** The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced, and without significant delay. This kind of delivery is called *real-time* transmission.

4. **Jitter.** Jitter refers to the variation in the packet arrival time. It is the uneven delay in the delivery of audio or video packets. For example, let us assume that video packets are sent every 30 ms. If some of the packets arrive with 30-ms delay and others with 40-ms delay, an uneven quality in the video is the result.

---

A data communications system has five components

1. **Message.** The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.

2 **Sender.** The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.

3. **Receiver.** The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.

4. **Transmission medium.** The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.

5. **Protocol.** A protocol is a set of rules that govern data communications. It

Text  
Numbers  
Images  
Audio  
Video

---

## Data Flow

Communication between two devices can be simplex, half-duplex, or full-duplex as shown in Figure.



**Simplex** In simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive (Figure a). Keyboards and traditional monitors are examples of simplex devices.

## Half-Duplex

In half-duplex mode, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa (Figure b). Walkie-talkies and CB (citizens band) radios are both half-duplex systems.

an inquiry and a response. The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

Performance is often evaluated by two networking metrics: **throughput and delay**. We often need more throughput and less delay. However, these two criteria are often contradictory. If we try to send more data to the network, we may increase throughput but we increase the delay because of traffic congestion in the network.

**Reliability:** In addition to accuracy of delivery, network reliability is measured by the frequency of failure, the time it takes a link to recover from a failure, and the network's robustness in a catastrophe.

**Security:** Network security issues include protecting data from unauthorized access, protecting data from damage and development, and implementing policies and procedures for recovery from breaches and data losses.

---

## **Physical Structures**

Before discussing networks, we need to define some network attributes.

### **Type of Connection**

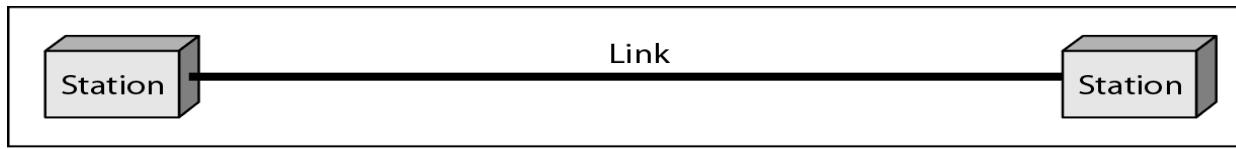
A network is two or more devices connected through links. A link is a communications pathway that transfers data from one device to another.

There are two possible types of connections: point-to-point and multipoint.

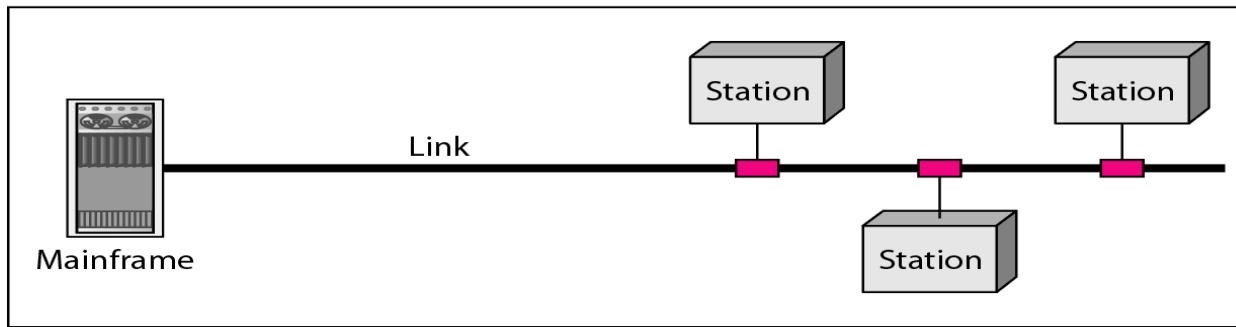
**Point-to-Point** A point-to-point connection provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices. Most point-to-point connections use an actual length of wire or cable to connect the two ends, but other options, such as microwave or satellite links, are also possible.

When you change television channels by infrared remote control, you are establishing a point-to-point connection between the remote control and the television's control system.

**Multipoint** A multipoint (also called multi-drop) connection is one in which more than two specific devices share a single link.



a. Point-to-point



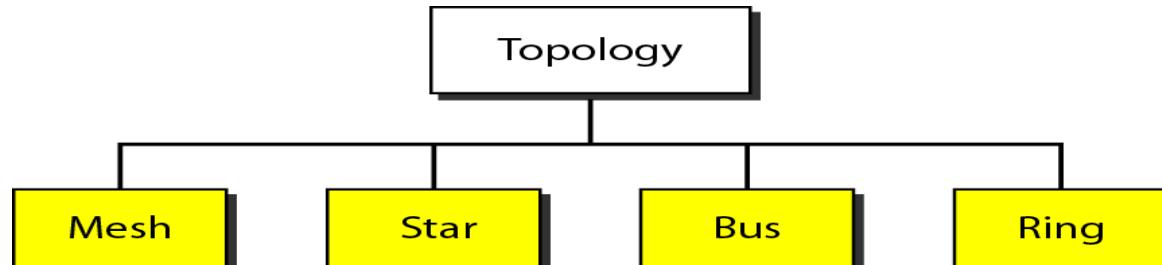
b. Multipoint

## ***Physical Topology***

The term *physical topology* refers to the way in which a network is laid out physically.

Two or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another.

There are four basic topologies possible: mesh, star, bus, and ring



### **MESH:**

A mesh topology is the one where every node is connected to every other node in the network.

network can be calculated using the following formula ( $n$  is the number of computers in the network):  $n(n-1)/2$

In a *partially connected mesh topology*, at least two of the computers in the network have connections to multiple other computers in that network. It is an inexpensive way to implement redundancy in a network. In the event that one of the primary computers or connections in the network fails, the rest of the network continues to operate normally.

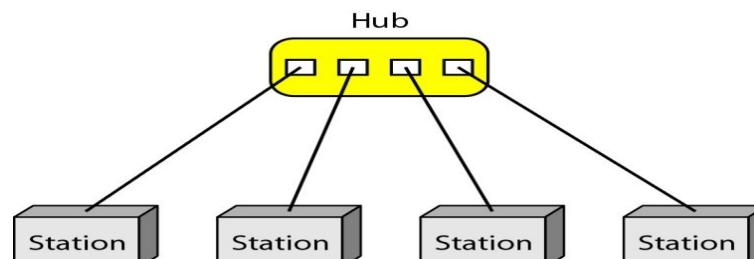
### Advantages of a mesh topology

- Can handle high amounts of traffic, because multiple devices can transmit data simultaneously.
- A failure of one device does not cause a break in the network or transmission of data.
- Adding additional devices does not disrupt data transmission between other devices.

### Disadvantages of a mesh topology

- The cost to implement is higher than other network topologies, making it a less desirable option.
- Building and maintaining the topology is difficult and time consuming.
- The chance of redundant connections is high, which adds to the high costs and potential for reduced efficiency.

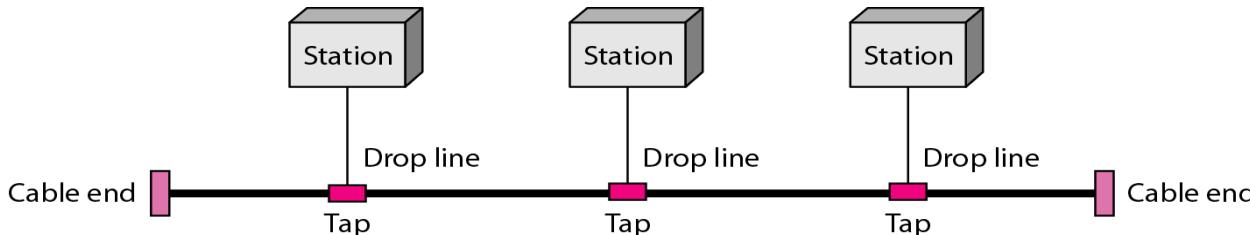
### STAR:



A **star network, star topology** is one of the most common network setups. In this configuration, every node connects to a central network device, like a hub, switch, or computer. The central network device acts as a server and the

- Can have a higher cost to implement, especially when using a switch or router as the central network device.
- The central network device determines the performance and number of nodes the network can handle.
- If the central computer, hub, or switch fails, the entire network goes down and all computers are disconnected from the network

## BUS:



a **line topology**, a **bus topology** is a network setup in which each computer and network device are connected to a single cable or [backbone](#).

### Advantages of bus topology

- It works well when you have a small network.
- It's the easiest network topology for connecting computers or peripherals in a linear fashion.
- It requires less cable length than a star topology.

### Disadvantages of bus topology

- It can be difficult to identify the problems if the whole network goes down.
- It can be hard to troubleshoot individual device issues.
- Bus topology is not great for large networks.
- Terminators are required for both ends of the main cable.
- Additional devices slow the network down.
- If a main cable is damaged, the network fails or splits into two.

A **ring topology** is a [network](#) configuration in which device connections create a circular [data](#) path. In a ring network, [packets](#) of data travel from one device to the next until they reach their destination. Most ring topologies allow packets to travel only in one direction, called a **unidirectional** ring network. Others permit data to move in either direction, called **bidirectional**.

The major disadvantage of a ring topology is that if any individual connection in the ring is broken, the entire network is affected.

Ring topologies may be used in either local area networks ([LANs](#)) or wide area networks ([WANs](#)).

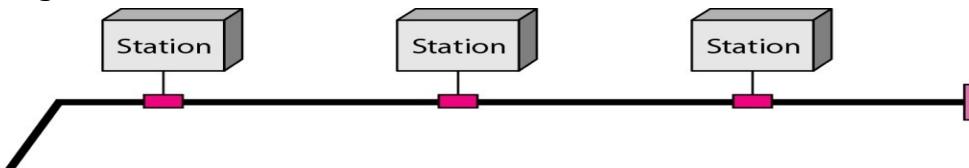
### Advantages of ring topology

- All data flows in one direction, reducing the chance of packet collisions.
- A network server is not needed to control network connectivity between each workstation.
- Data can transfer between workstations at high speeds.
- Additional workstations can be added without impacting performance of the network.

### Disadvantages of ring topology

- All data being transferred over the network must pass through each workstation on the network, which can make it slower than a [star topology](#).
- The entire network will be impacted if one workstation shuts down.
- The hardware needed to connect each workstation to the network is more expensive than Ethernet cards and hubs/switches.

**Hybrid Topology** A network can be hybrid. For example, we can have a main star topology with each branch connecting several stations in a bus topology as shown in Figure



## **LAN (Local Area Network)**

Group of interconnected computers within a small area. (room, building, campus)

Two or more pc's can from a LAN to share files, folders, printers, applications and other devices.

Coaxial or CAT 5 cables are normally used for connections.

Due to short distances, errors and noise are minimum.

Data transfer rate is 10 to 100 mbps.

Example: A computer lab in a school.

## **MAN (Metropolitan Area Network)**

Design to extend over a large area.

Connecting number of LAN's to form larger network, so that resources can be shared.

Networks can be up to 5 to 50 km.

Owned by organization or individual.

Data transfer rate is low compare to LAN.

Example: Organization with different branches located in the city.

## **WAN (Wide Area Network)**

Are country and worldwide network.

Contains multiple LAN's and MAN's.

Distinguished in terms of geographical range.

Uses satellites and microwave relays.

Data transfer rate depends upon the ISP provider and varies over the location.

Best example is the internet.

## **Other types**

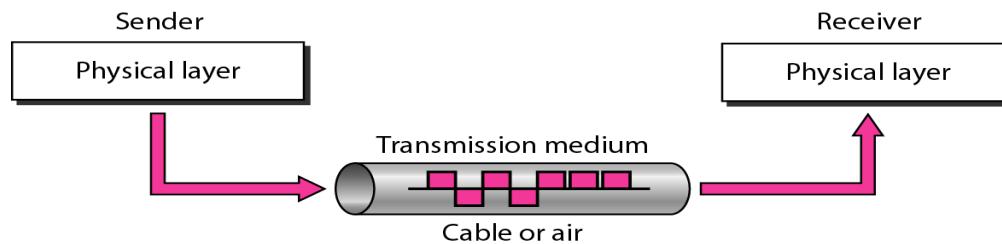
### **WLAN (Wireless LAN)**

A LAN that uses high frequency radio waves for communication.

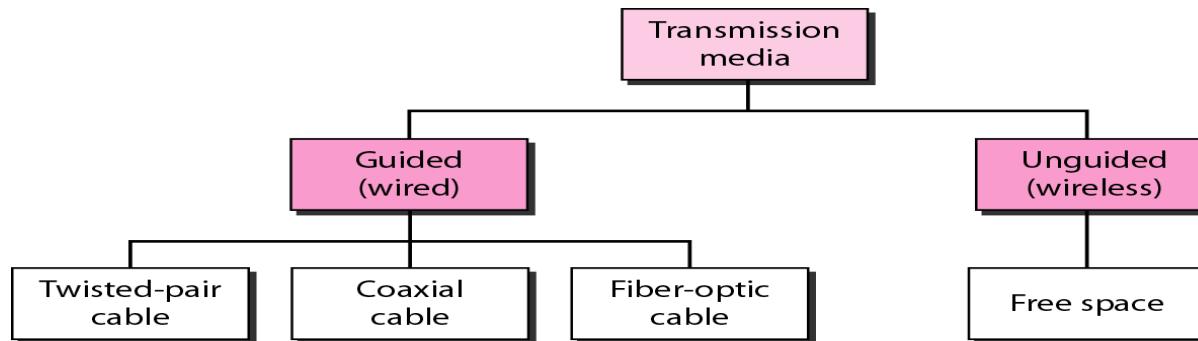
Provides short range connectivity with high speed data transmission.

### **PAN (Personal Area Network)**

Network designed for individual user's facility.

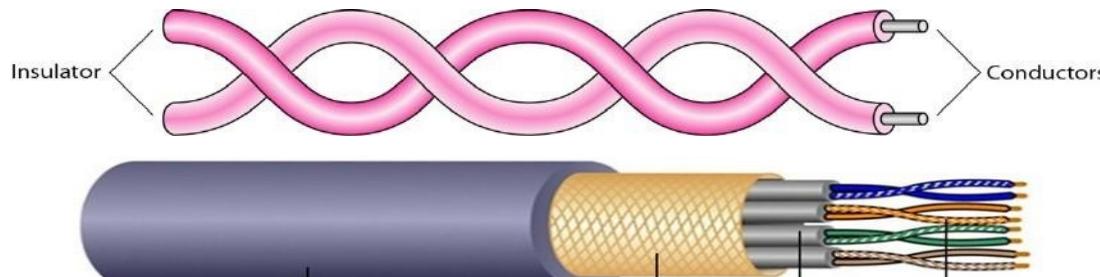


## Classes of transmission media



**Guided Media:** Guided media, which are those that provide a medium from one device to another, include twisted-pair cable, coaxial cable, and fiber-optic cable.

**Twisted-Pair Cable:** A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together. One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference.



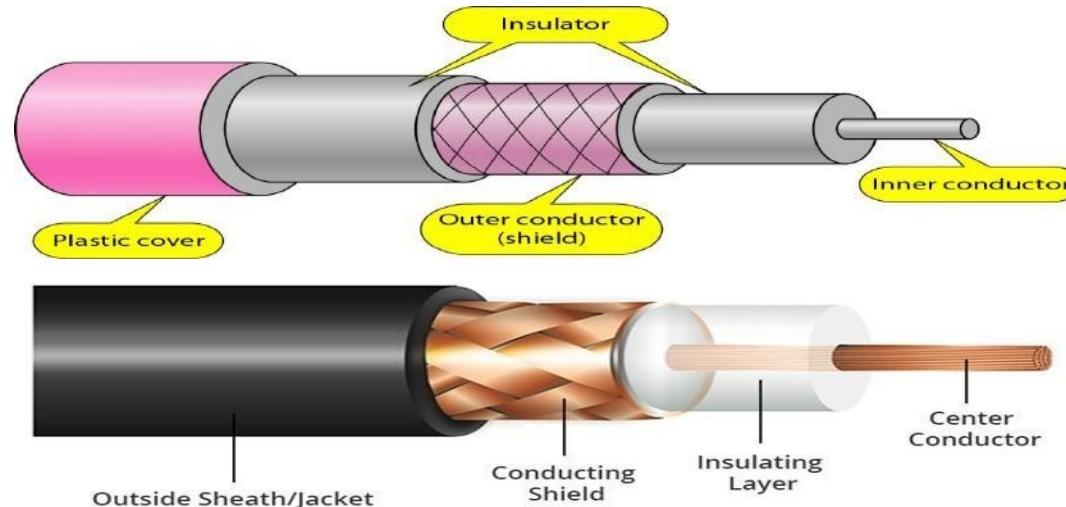
## Applications

Twisted-pair cables are used in telephone lines to provide voice and data channels.

Local-area networks, such as 10Base-T and 100Base-T, also use twisted-pair cables.

## Coaxial Cable

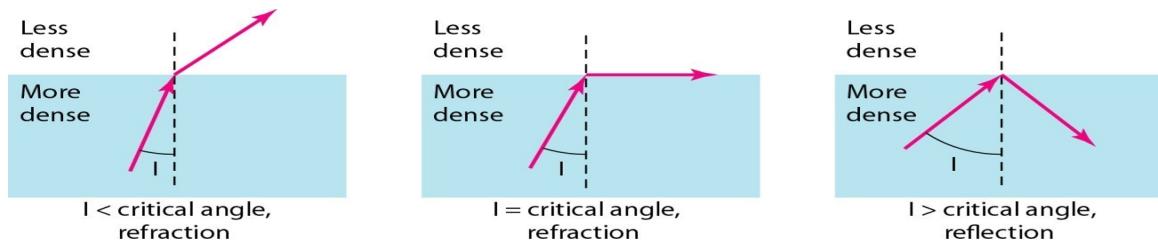
Coaxial cable (or coax) carries signals of higher frequency ranges than those in twisted pair cable. coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover.



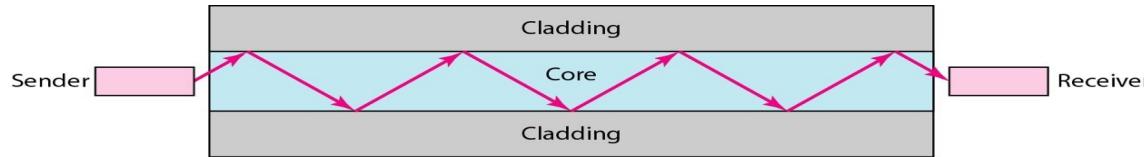
The most common type of connector used today is the Bayone-Neill-Concelman (BNC), connector.

## Applications

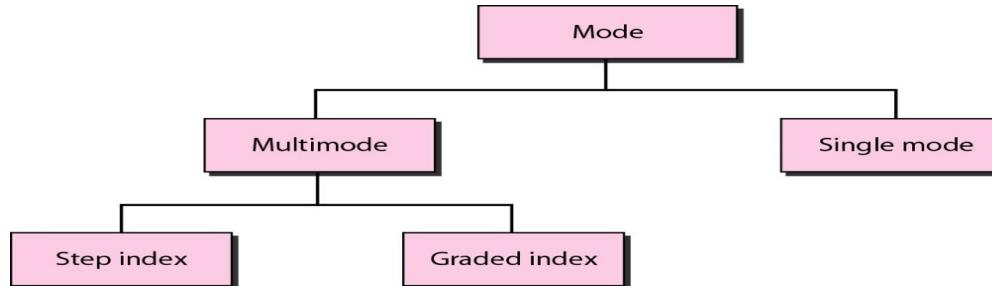
Coaxial cable was widely used in analog telephone networks, digital telephone



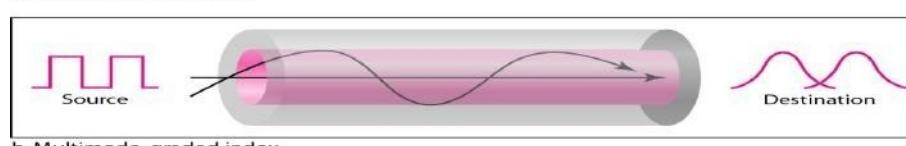
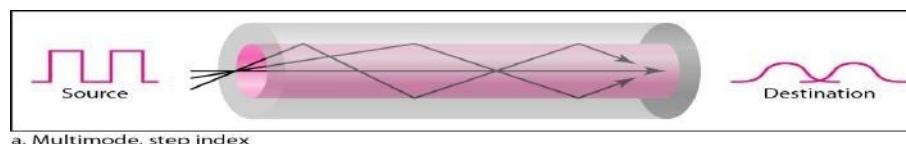
Optical fibers use reflection to guide light through a channel. A glass or plastic core is surrounded by a cladding of less dense glass or plastic.



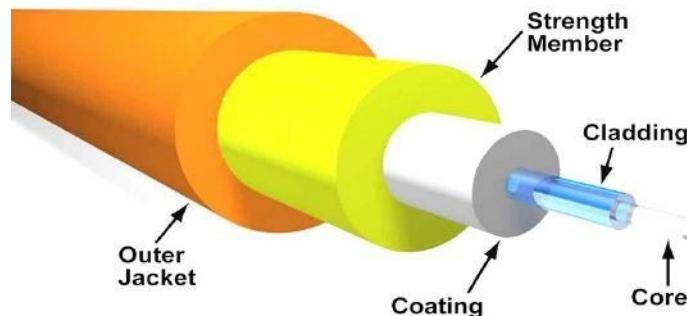
## Propagation Modes



Multimode is so named because multiple beams from a light source move through the core in different paths. How these beams move within the cable depends on the structure of the core, as shown in Figure.



## Fiber Construction



The **subscriber channel (SC) connector**, The **straight-tip (ST) connector**, **MT-RJ(mechanical transfer registered jack)** is a connector

### Applications

Fiber-optic cable is often found in backbone networks because its wide bandwidth is cost-effective..

Some cable TV companies use a combination of optical fiber and coaxial cable, thus creating a hybrid network.

Local-area networks such as 100Base-FX network (Fast Ethernet) and 1000Base-X also use fiber-optic cable

### Advantages and Disadvantages of Optical Fiber

**Advantages** Fiber-optic cable has several advantages over metallic cable (twisted pair or coaxial).

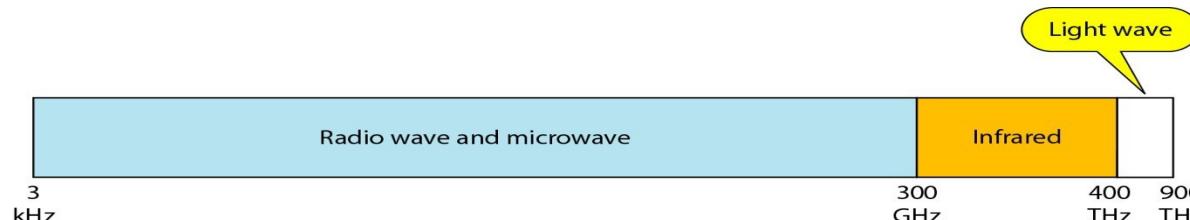
- 1 Higher bandwidth.
- 2 Less signal attenuation. Fiber-optic transmission distance is significantly greater than that of other guided media. A signal can run for 50 km without requiring regeneration. We need repeaters every 5 km for coaxial or twisted-pair cable.
- 3 Immunity to electromagnetic interference. Electromagnetic noise cannot affect fiber-optic cables.
- 4 Resistance to corrosive materials. Glass is more resistant to corrosive materials than copper.
- 5 Light weight. Fiber-optic cables are much lighter than copper cables.

Unguided media transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication.

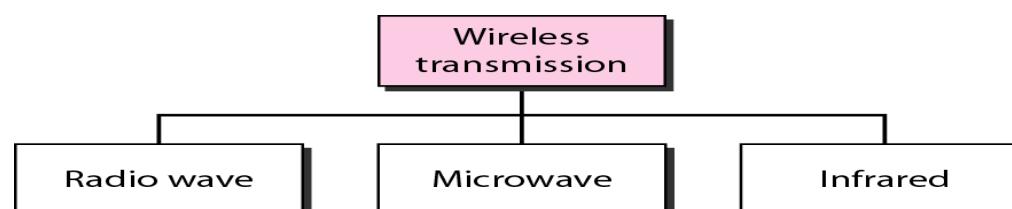
## Radio Waves

## Microwaves

## Infrared

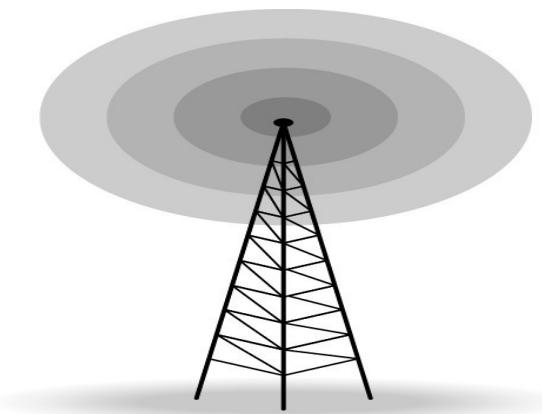


Unguided signals can travel from the source to destination in several ways: ground propagation, sky propagation, and line-of-sight propagation, as shown in Figure



## Radio Waves

Electromagnetic waves ranging in frequencies between 3 kHz and 1 GHz are normally called radio waves. Radio waves are omni directional. When an antenna transmits radio waves they are propagated in all directions. This



### *Applications*

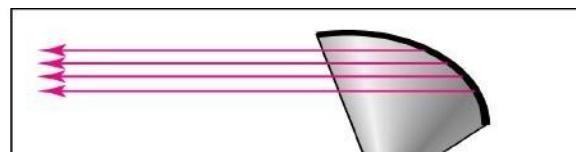
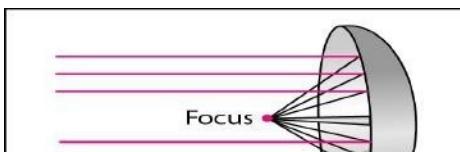
The Omni directional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers. AM and FM radio, television, maritime radio, cordless phones, and paging are examples of multicasting.

## **Microwaves**

Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves. Microwaves are unidirectional. The sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas

### *Unidirectional Antenna*

Microwaves need unidirectional antennas that send out signals in one direction. Two types of antennas are used for microwave communications: the parabolic dish and the horn



characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room.

When we use our infrared remote control, we do not interfere with the use of the remote by our neighbors. Infrared signals useless for long-range communication. In addition, we cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

### **Applications:**

**Infrared signals can be used for short-range communication in a closed area using line-of-sight propagation.**

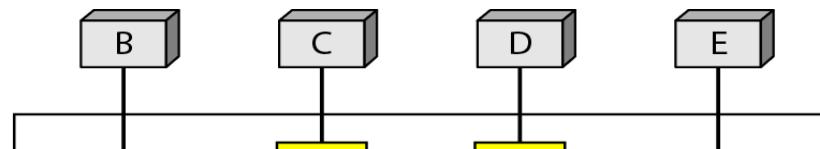
---

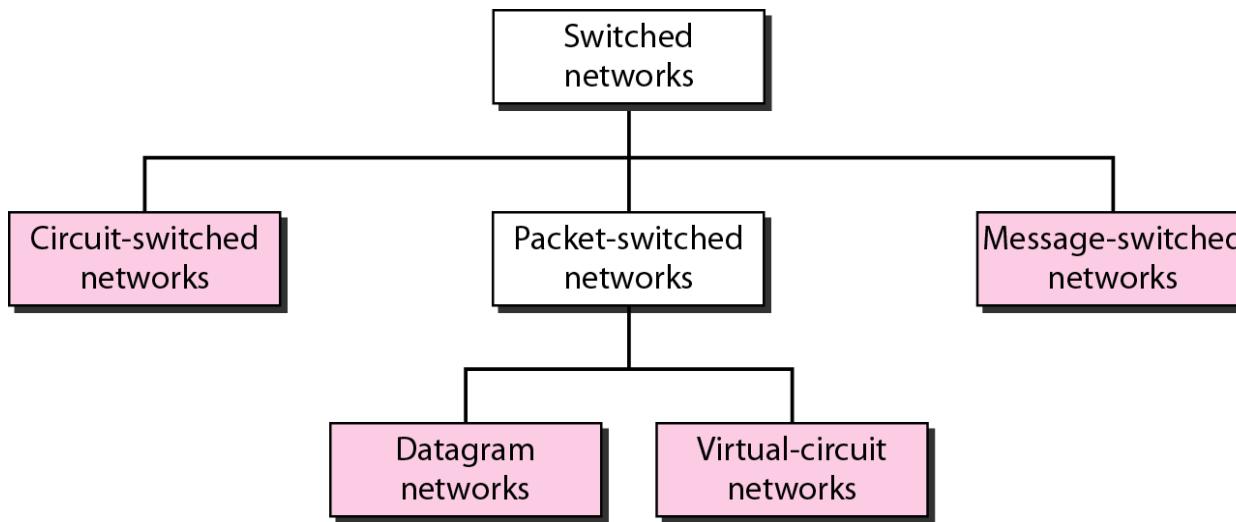
### **Switching**

A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology). These methods, however, are impractical and wasteful when applied to very large networks.

The number and length of the links require too much infrastructure to be cost-efficient, and the majority of those links would be idle most of the time.

A better solution is switching. A switched network consists of a series of interlinked nodes, called switches. Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems (computers or telephones, for example). Others are used only for routing. Figure shows a switched network.



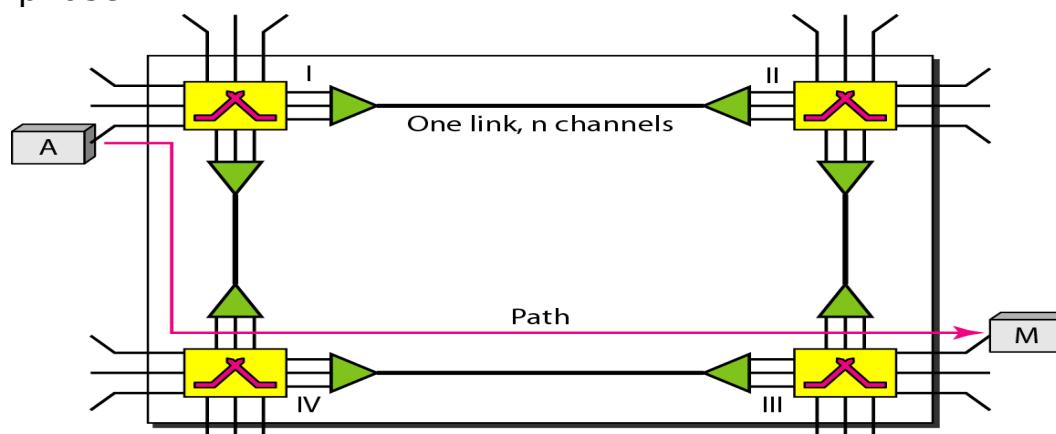


## CIRCUIT-SWITCHED NETWORKS

A circuit-switched network consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links. However, each connection uses only one dedicated channel on each link. Each link is normally divided into  $n$  channels by using FDM or TDM.

In circuit switching, the resources need to be reserved during the setup phase;

the resources remain dedicated for the entire duration of data transfer until the teardown phase



dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time.

In the next step to making a connection, an acknowledgment from system M needs to be sent in the opposite direction to system A. Only after system A receives this acknowledgment is the connection established.

### Data Transfer Phase

After the establishment of the dedicated circuit (channels), the two parties can transfer data.

### Teardown Phase

When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

### **Efficiency**

It can be argued that circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection. These resources are unavailable to other connections.

### **Delay**

Although a circuit-switched network normally has low efficiency, the delay in this type of network is minimal. During data transfer the data are not delayed at each switch; the resources are allocated for the duration of the connection. The total delay is due to the time needed to create the connection, transfer data, and disconnect the circuit.

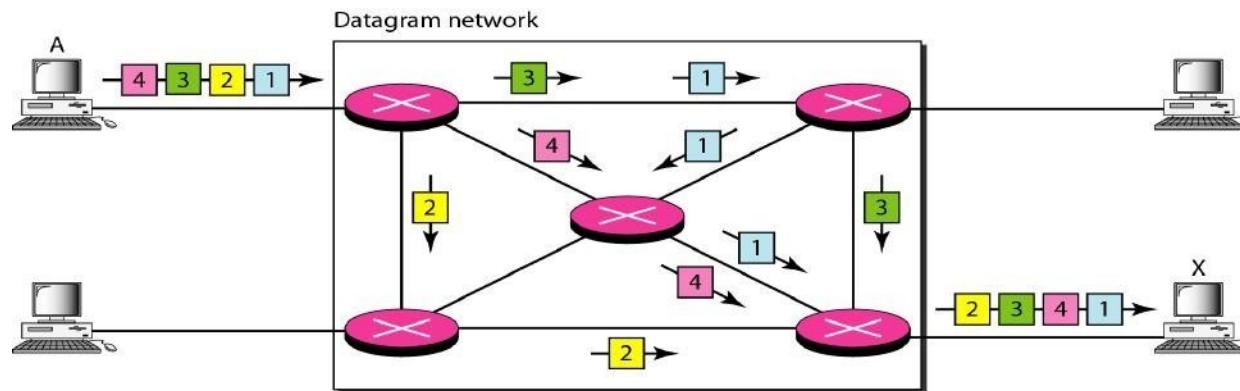
**Switching at the physical layer in the traditional telephone network uses the circuit-switching approach.**

---

## **DATAGRAM NETWORKS**

In a packet-switched network, there is no resource reservation; resources are allocated on demand. The allocation is done on a first come, first-served basis. When a switch receives a packet, no matter what is the source or destination, the packet must wait if there are other packets being processed. This lack of reservation may create delay. For example, if we do not have a

A switch in a datagram network uses a routing table that is based on the destination address. The destination address in the header of a packet in a datagram network remains the same during the entire journey of the packet.



### Efficiency

The efficiency of a datagram network is better than that of a circuit-switched network; resources are allocated only when there are packets to be transferred.

### Delay

There may be greater delay in a datagram network than in a virtual-circuit network. Although there are no setup and teardown phases, each packet may experience a wait at a switch before it is forwarded. In addition, since not all packets in a message necessarily travel through the same switches, the delay is not uniform for the packets of a message.

Switching in the Internet is done by using the datagram approach to packet switching at the network layer.

---

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.
2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
3. As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it defines what should be the next switch and the channel on which the packet is being carried), not end-to-end jurisdiction.
4. As in a circuit-switched network, all packets follow the same path established during the connection.
5. A virtual-circuit network is normally implemented in the data link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer.

### Addressing

In a virtual-circuit network, two types of addressing are involved: global and local (virtual-circuit identifier).

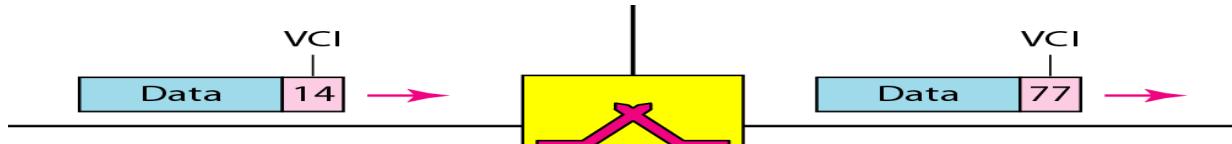
#### Global Addressing

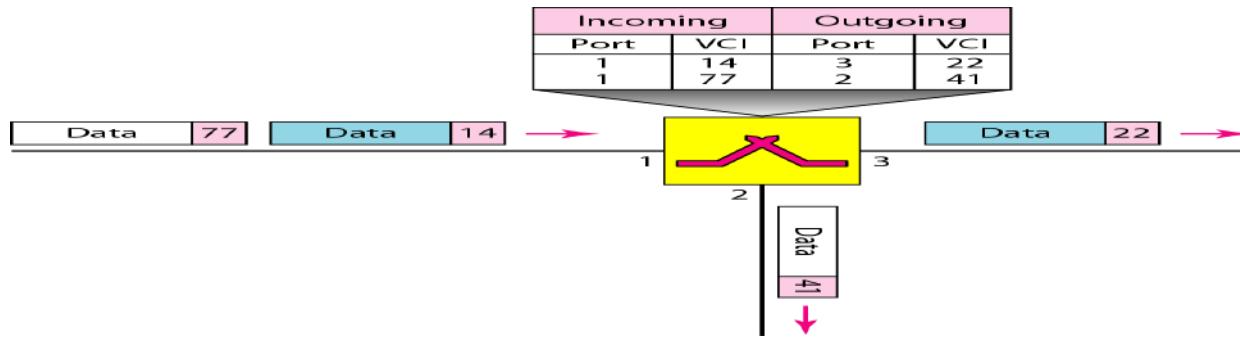
A source or a destination needs to have a global address—an address that can be unique in the scope of the network.

#### Virtual-Circuit Identifier

The identifier that is actually used for data transfer is called the virtual-circuit identifier (VCI). A VCI, unlike a global address, is a small number that has only switch scope; it is used by a frame between two switches. When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCI.

Figure shows how the VCI in a data frame changes from one switch to another. Note that a VCI does not need to be a large number since each switch can use its own unique set of VCIs.



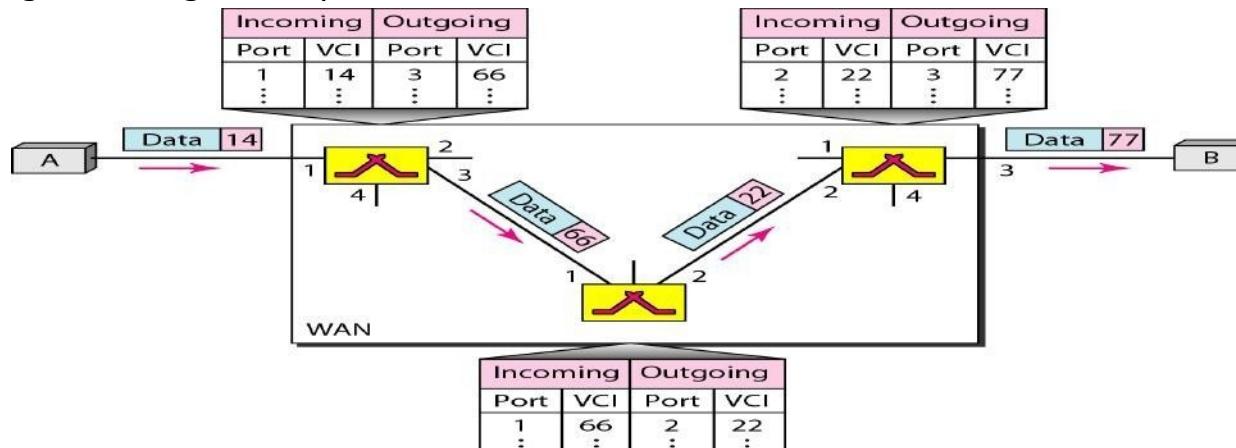


To transfer a frame from a source to its destination, all switches need to have a table entry for this virtual circuit. The table, in its simplest form, has four columns.

We show later how the switches make their table entries, but for the moment we assume that each switch has a table with entries for all active virtual circuits. Figure shows such a switch and its corresponding table.

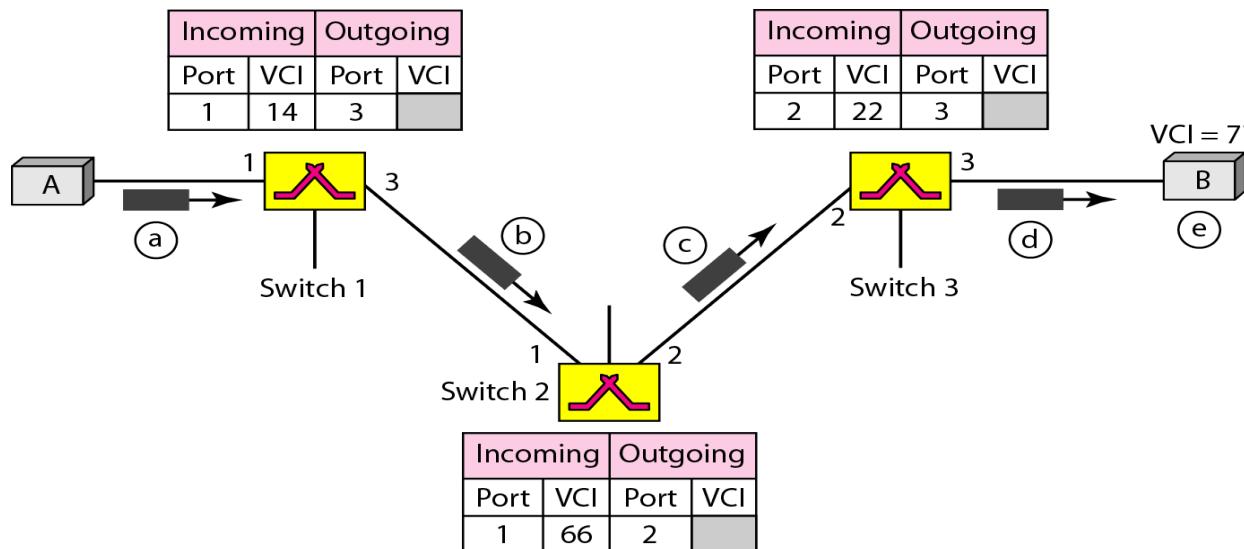
Figure shows a frame arriving at port 1 with a VCI of 14. When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14. When it is found, the switch knows to change the VCI to 22 and send out the frame from port 3.

Figure shows how a frame from source A reaches destination B and how its VCI changes during the trip.



Each switch changes the VCI and routes the frame.

The destination for the frame is the switch connected to the destination computer.



- Source A sends a setup frame to switch 1.
- Switch 1 receives the setup request frame. It knows that a frame going from A to B goes out through port 3. For the moment, assume that it knows the output port. The switch creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The switch assigns the incoming port (1) and chooses an available incoming VCI (14) and the outgoing port (3). It does not yet know the outgoing VCI, which will be found during the acknowledgment step. The switch then forwards the frame through port 3 to switch 2.
- Switch 2 receives the setup request frame. The same events happen here as at switch 1; three columns of the table are completed: in this case, incoming port (1), incoming VCI (66), and outgoing port (2).
- Switch 3 receives the setup request frame. Again, three columns are completed: incoming port (2), incoming VCI (22), and outgoing port (3).
- Destination B receives the setup frame, and if it is ready to receive frames from A, it assigns a VCI to the incoming frames that come from A, in this case 77. This VCI lets the destination know that the frames come from A, and not other sources.

**Acknowledgment** A special frame, called the acknowledgment frame, completes

- a. The destination sends an acknowledgment to switch 3. The acknowledgment carries the global source and destination addresses so the switch knows which entry in the table is to be completed. The frame also carries VCI 77, chosen by the destination as the incoming VCI for frames from A. Switch 3 uses this VCI to complete the outgoing VCI column for this entry. Note that 77 is the incoming VCI for destination B, but the outgoing VCI for switch 3.
- b. Switch 3 sends an acknowledgment to switch 2 that contains its incoming VCI in the table, chosen in the previous step. Switch 2 uses this as the outgoing VCI in the table.
- c. Switch 2 sends an acknowledgment to switch 1 that contains its incoming VCI in the table, chosen in the previous step. Switch 1 uses this as the outgoing VCI in the table.
- d. Finally switch 1 sends an acknowledgment to source A that contains its incoming VCI in the table, chosen in the previous step.
- e. The source uses this as the outgoing VCI for the data frames to be sent to destination B.

### ***Teardown Phase***

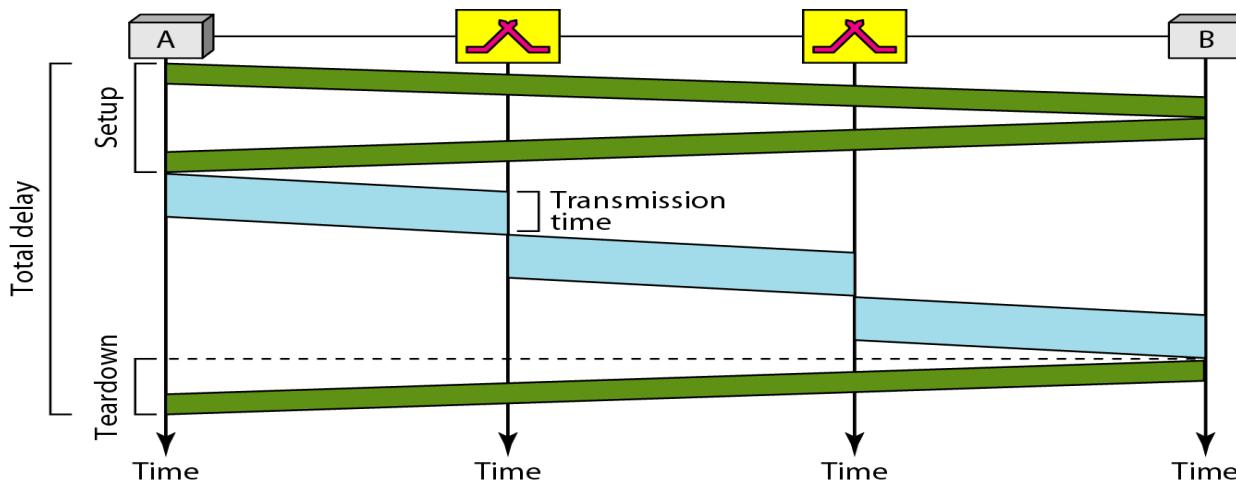
In this phase, source A, after sending all frames to B, sends a special frame called a *teardown request*. Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

### ***Efficiency***

In virtual-circuit switching, all packets belonging to the same source and destination travel the same path; but the packets may arrive at the destination with different delays if resource allocation is on demand.

### ***Delay***

In a virtual-circuit network, there is a one-time delay for setup and a one-time



Switching at the data link layer in a switched WAN is normally implemented by using virtual-circuit techniques.

---

## Comparison

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

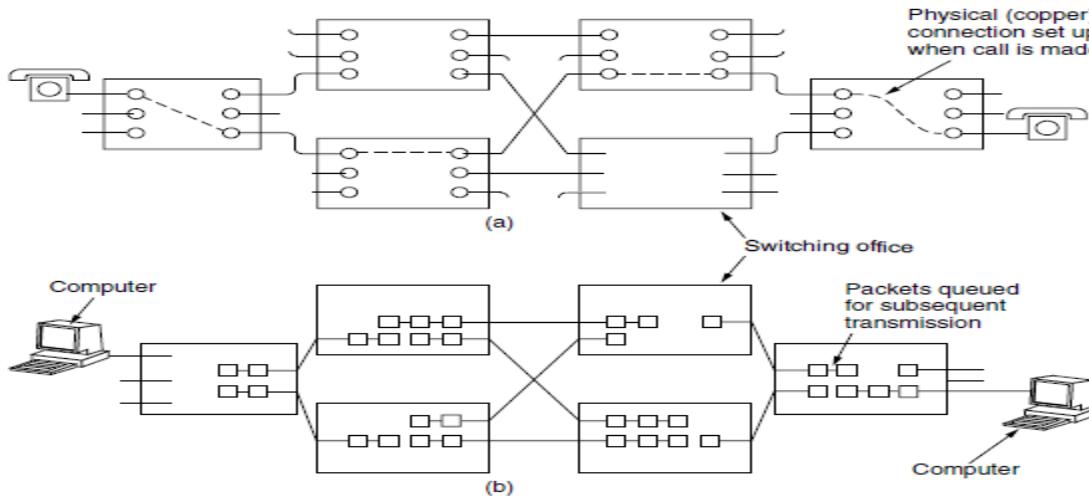


Figure 2-42. (a) Circuit switching. (b) Packet switching.

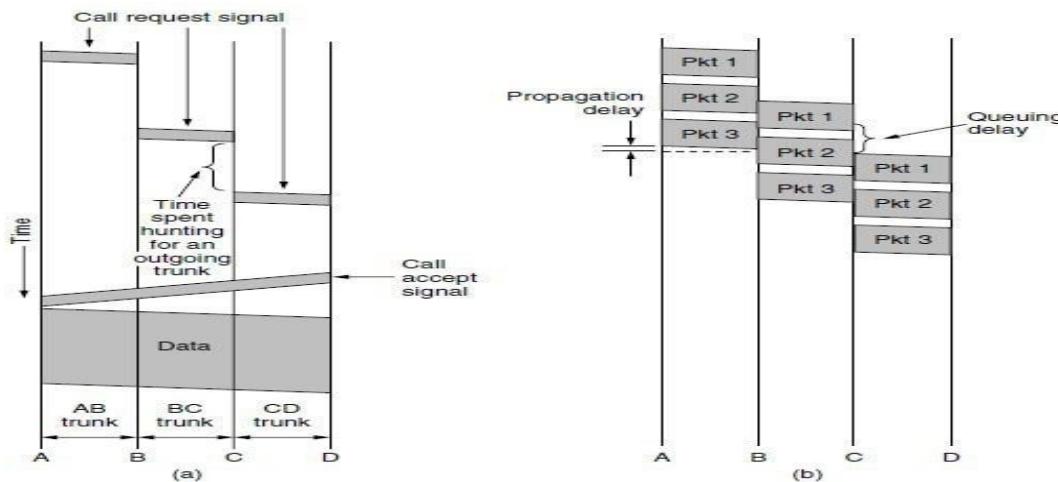


Figure 2-43. Timing of events in (a) circuit switching, (b) packet switching.

## OSI

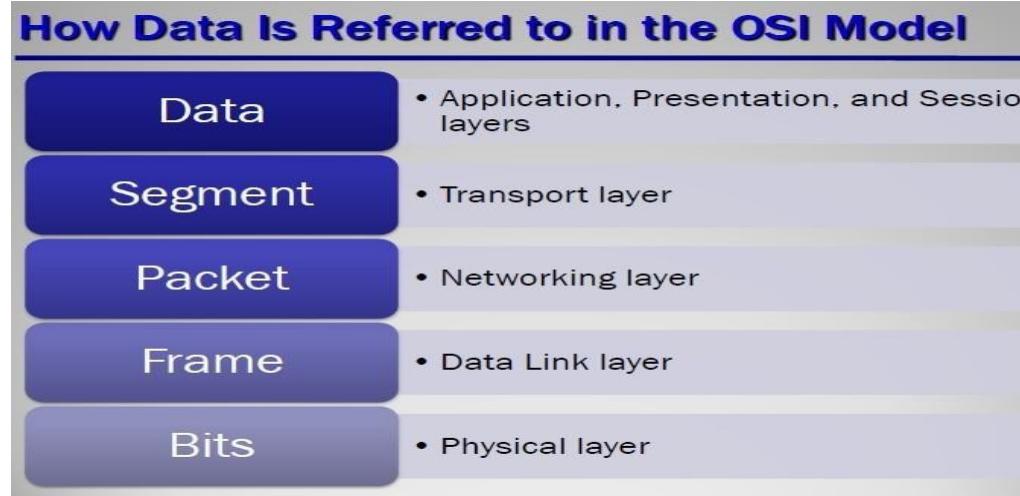
- OSI stands for Open Systems Interconnection
- Created by International Standards Organization (ISO)
- Was created as a framework and reference model to explain how different computing devices can interoperate

Top to bottom

-All People Seem To Need Data Processing

Bottom to top

-Please Do Not Throw Sausage Pizza Away



## **Physical Layer**

- Deals with all aspects of physically moving data from one computer to the next
- Converts data from the upper layers into 1s and 0s for transmission over media
- Defines how data is encoded onto the media to transmit the data
- Defined on this layer: Cable standards, wireless standards, and fiber optic standards.

Copper wiring, fiber optic cable, radio frequencies, anything that can be used to transmit data is defined on the Physical layer of the OSI Model

- Device example: Hub
- Used to transmit data

## **Data Link Layer**

- Is responsible for moving frames from node to node or computer to computer
- Can move frames from one adjacent computer to another, cannot move frames across routers

- Responsible for moving packets (data) from one end of the network to the other, called *end-to-end communications*
- Requires *logical addresses* such as IP addresses
- Device example: Router
- -Routing is the ability of various network devices and their related software to move data packets from source to destination

### **Transport Layer**

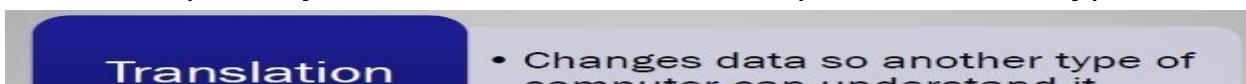
- Takes data from higher levels of OSI Model and breaks it into segments that can be sent to lower-level layers for data transmission
- Conversely, reassembles data segments into data that higher-level protocols and applications can use
- Also puts segments in correct order (called sequencing ) so they can be reassembled in correct order at destination
- Concerned with the reliability of the transport of sent data
- May use a *connection-oriented protocol* such as TCP to ensure destination received segments
- May use a *connectionless protocol* such as UDP to send segments without assurance of delivery
- Uses port addressing

### **Session Layer**

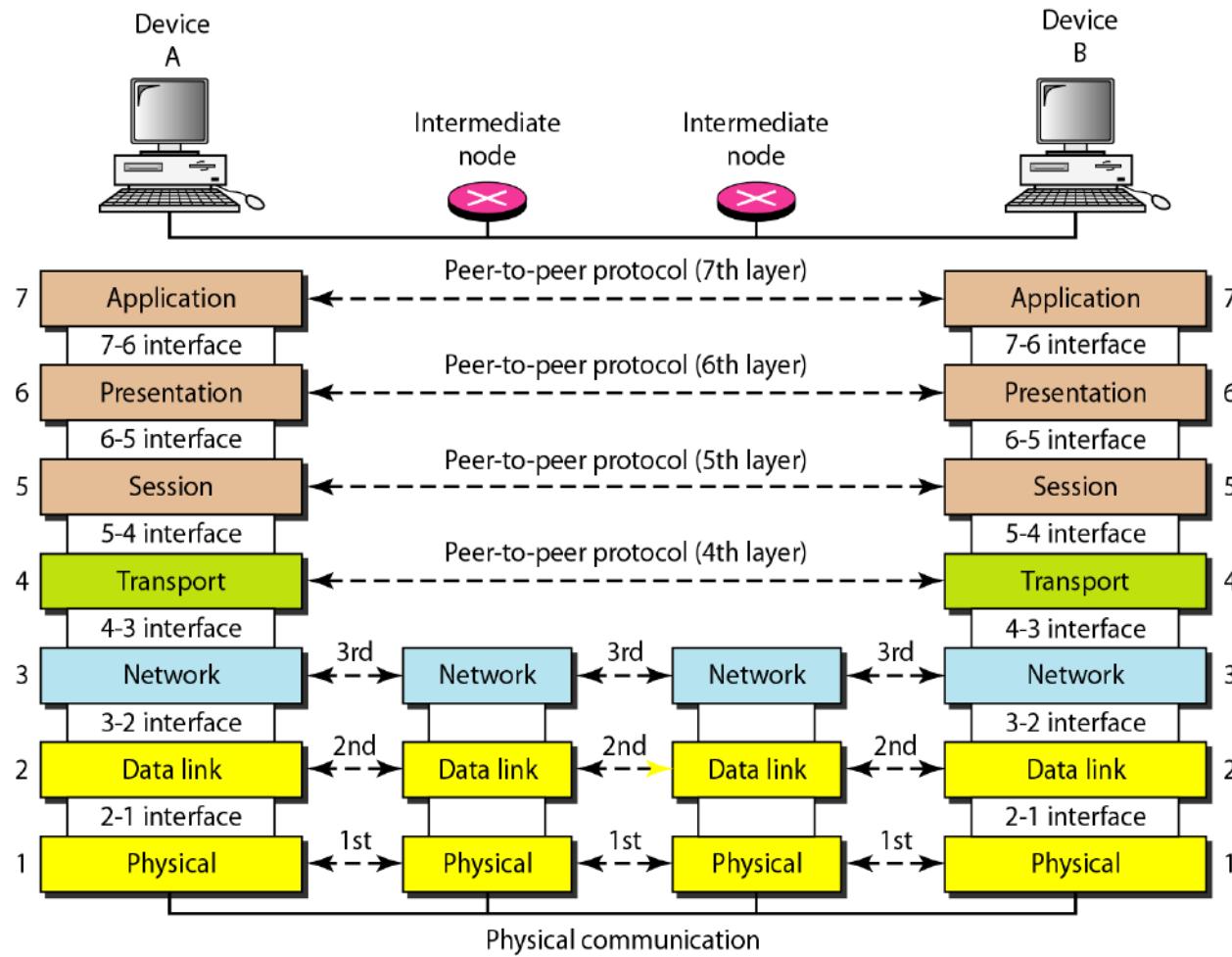
- Responsible for managing the dialog between networked devices
- Establishes, manages, and terminates connections
- Provides duplex, half-duplex, or simplex communications between devices
- Provides procedures for establishing checkpoints, adjournment, termination, and restart or recovery procedures

### **Presentation Layer**

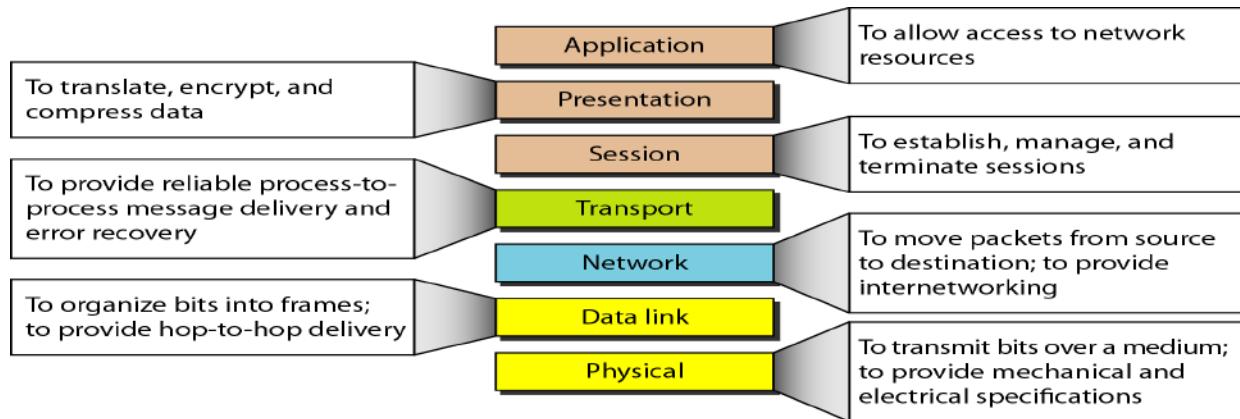
- Concerned with how data is presented to the network
- Handles three primary tasks: -Translation , -Compression , -Encryption



## ***The interaction between layers in the OSI model***

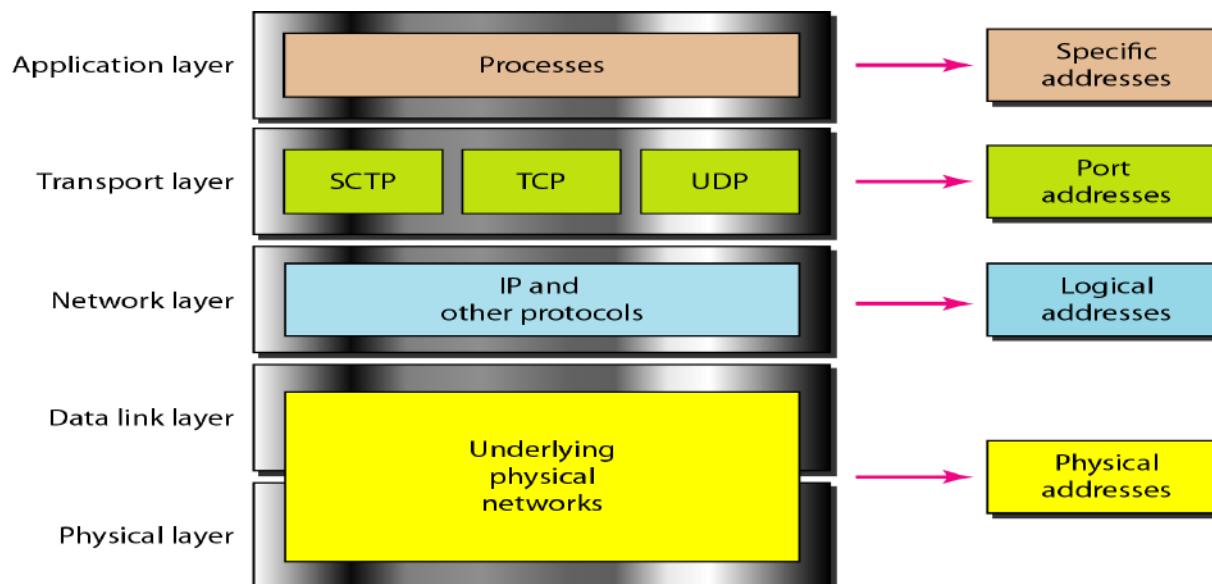


***An exchange using the OSI model***



## **TCP/IP Model (Transmission Control Protocol/Internet Protocol)**

-A *protocol suite* is a large number of related protocols that work together to allow networked computers to communicate



***Relationship of layers and addresses in TCP/IP***

### **Application Layer**

- Application layer protocols define the rules when implementing specific network applications

- TCP is a connection-oriented protocol
- Does not mean it has a physical connection between sender and receiver
- TCP provides the function to allow a connection virtually exists – also called virtual circuit
- UDP provides the functions:
- Dividing a chunk of data into segments
- Reassembly segments into the original chunk
- Provide further the functions such as reordering and data resend
- Offering a reliable byte-stream delivery service
- Functions the same as the Transport layer in OSI
- Synchronize source and destination computers to set up the session between the respective computers

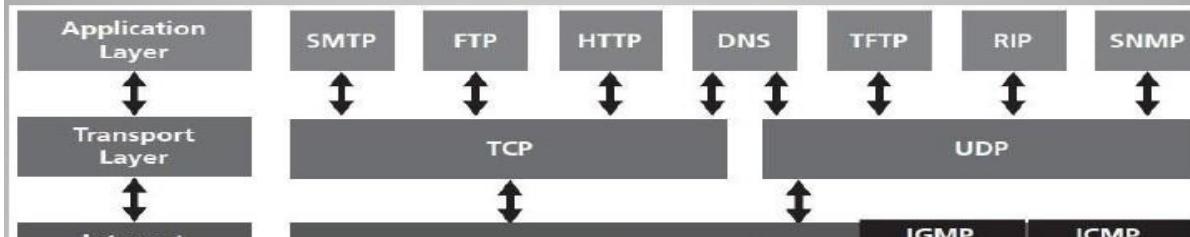
### **Internet Layer**

- The network layer, also called the internet layer, deals with packets and connects independent networks to transport the packets across network boundaries. The network layer protocols are the IP and the Internet Control Message Protocol ([ICMP](#)), which is used for error reporting.

### **Host-to-network layer**

The **Host-to-network layer** is the lowest **layer** of the **TCP/IP** reference model. It combines the link **layer** and the physical **layer** of the ISO/OSI model. At this **layer**, data is transferred between adjacent **network** nodes in a WAN or between nodes on the same LAN.

### **TCP/IP Model and its Relation to Protocols of the TCP/IP Suite**



<b>OSI MODEL</b>	<b>TCP/IP MODEL</b>
Contains 7 Layers	Contains 4 Layers
Uses Strict Layering resulting in vertical layers.	Uses Loose Layering resulting in horizontal layers.
Supports both connectionless & connection-oriented communication in the Network layer, but only connection-oriented communication in Transport Layer	Supports only connectionless communication in the Network layer, but both connectionless & connection-oriented communication in Transport Layer
It distinguishes between Service, Interface and Protocol.	Does not clearly distinguish between Service, Interface and Protocol.
Protocols are better hidden and can be replaced relatively easily as technology changes (No transparency)	Protocols are not hidden and thus cannot be replaced easily. (Transparency) Replacing IP by a substantially different protocol would be virtually impossible
OSI reference model was devised before the corresponding protocols were designed.	The protocols came first and the model was a description of the existing protocols

## THE INTERNET

The Internet has revolutionized many aspects of our daily lives. It has affected the way we do business as well as the way we spend our leisure time. Count the ways you've used the Internet recently. Perhaps you've sent electronic mail (e-mail) to a business associate, paid a utility bill, read a newspaper from a distant city, or looked up a local movie schedule-all by using the Internet. Or maybe you researched a medical topic, booked a hotel reservation, chatted with a fellow Trekkie, or comparison-shopped for a car. The Internet is a communication system that has brought a wealth of information to our fingertips and organized it for our use.

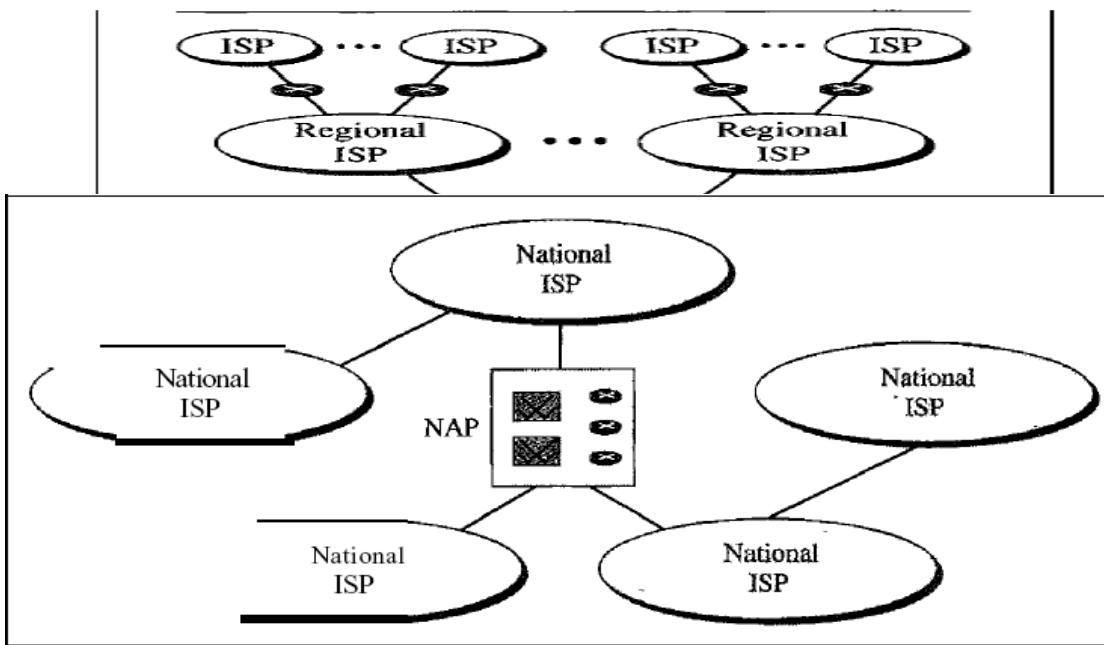
(ARPA) in the Department of Defense (DoD) was interested in finding a way to connect computers so that the researchers they funded could share their findings, thereby reducing costs and eliminating duplication of effort.

In 1967, at an Association for Computing Machinery (ACM) meeting, ARPA presented its ideas for ARPANET, a small network of connected computers. The idea was that each host computer (not necessarily from the same manufacturer) would be attached to a specialized computer, called an *interface message processor* (IMP). The IMPs, in turn, would be connected to one another. Each IMP had to be able to communicate with other IMPs as well as with its own attached host. By 1969, ARPANET was a reality. Four nodes, at the University of California at Los Angeles (UCLA), the University of California at Santa Barbara (UCSB), Stanford Research Institute (SRI), and the University of Utah, were connected via the IMPs to form a network. Software called the *Network Control Protocol* (NCP) provided communication between the hosts.

In 1972, Vint Cerf and Bob Kahn, both of whom were part of the core ARPANET group, collaborated on what they called the *Internetting Project*. Cerf and Kahn's landmark 1973 paper outlined the protocols to achieve end-to-end delivery of packets. This paper on Transmission Control Protocol (TCP) included concepts such as encapsulation, the datagram, and the functions of a gateway. Shortly thereafter, authorities made a decision to split TCP into two protocols: Transmission Control Protocol (TCP) and Internetworking Protocol (IP). IP would handle datagram routing while TCP would be responsible for higher-level functions such as segmentation, reassembly, and error detection. The internetworking protocol became known as TCPIIP.

## The Internet Today

The Internet has come a long way since the 1960s. The Internet today is not a simple hierarchical structure. It is made up of many wide- and local-area networks joined by connecting devices and switching stations. It is difficult to give an accurate representation of the Internet because it is continually changing-new networks are being added, existing networks are adding



b. Interconnection of national ISPs

### **International Internet Service Providers:**

At the top of the hierarchy are the international service providers that connect nations together.

### **National Internet Service Providers:**

The national Internet service providers are backbone networks created and maintained by specialized companies. There are many national ISPs operating in North America; some of the most well known are SprintLink, PSINet, UUNet Technology, AGIS, and internet Mel. To provide connectivity between the end users, these backbone networks are connected by complex switching stations (normally run by a third party) called network access points (NAPs). Some national ISP networks are also connected to one another by private switching stations called *peering points*. These normally operate at a high data rate (up to 600 Mbps).

### **Regional Internet Service Providers:**

Regional internet service providers or regional ISPs are smaller ISPs that are connected to one or more national ISPs. They are at the third level of the

# **UNIT- II**

## **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. To overcome this problem, the data link layer uses the flow control to prevent the sending node on one side of the link from overwhelming the receiving node on another side of the link. This prevents traffic jam at the receiver side.

**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.

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.

## **FRAMING:**

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. This bit stream is not guaranteed to be error free. The number of bits received may be less than, equal to, or more than the number of bits transmitted, and they may have different values. It is up to the data link layer to **detect and, if necessary, correct errors**. The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the checksum for each frame (**framing**). When a frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it (e.g., discarding the bad frame and possibly also sending back an error report). We will look at four framing methods:

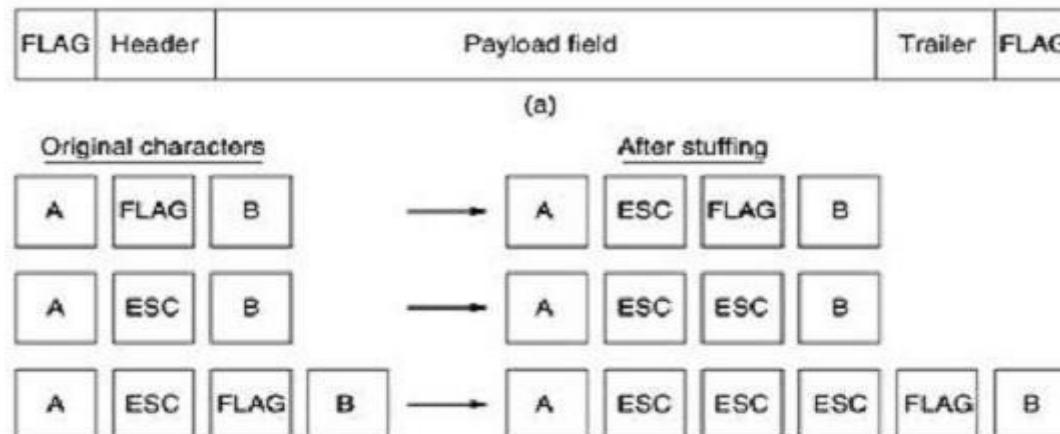
1. Character count.
2. Flag bytes with byte stuffing.
3. Starting and ending flags, with bit stuffing.
4. Physical layer coding violations.

**Character count** method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is. This method is shown in Figure 6.1. The first few bytes of the frame contain the character count.

A character stream. (a) Without errors. (b) With one error

The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the character count of 5 in the second frame of Fig. (b) becomes a 7, the destination will get out of synchronization and will be unable to locate the start of the next frame. Even if the checksum is incorrect so the destination knows that the frame is bad, it still has no way of telling where the next frame starts. Sending a frame back to the source asking for a retransmission does not help either, since the destination does not know how many characters to skip over to get to the start of the retransmission. For this reason, the character count method is rarely used anymore.

**Flag bytes with byte stuffing** method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. In the past, the starting and ending bytes were different, but in recent years most protocols have used the same byte, called a flag byte, as both the starting and ending delimiter, as shown in Fig. (a) as FLAG. In this way, if the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame. Two consecutive flag bytes indicate the end of one frame and start of the next one.



Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it.

What happens if an escape byte occurs in the middle of the data? The answer is that, it too is stuffed with an escape byte. Thus, any single escape byte is part of an escape sequence, whereas a doubled one indicates that a single escape occurred naturally in the data. Some examples are shown in Fig. (b). In all cases, the byte sequence delivered after de stuffing is exactly the same as the original byte sequence.

A major disadvantage of using this framing method is that it is closely tied to the use of 8-bit characters. Not all character codes use 8-bit characters. For example UNICODE uses 16-bit characters, so a new technique had to be developed to allow arbitrary sized characters

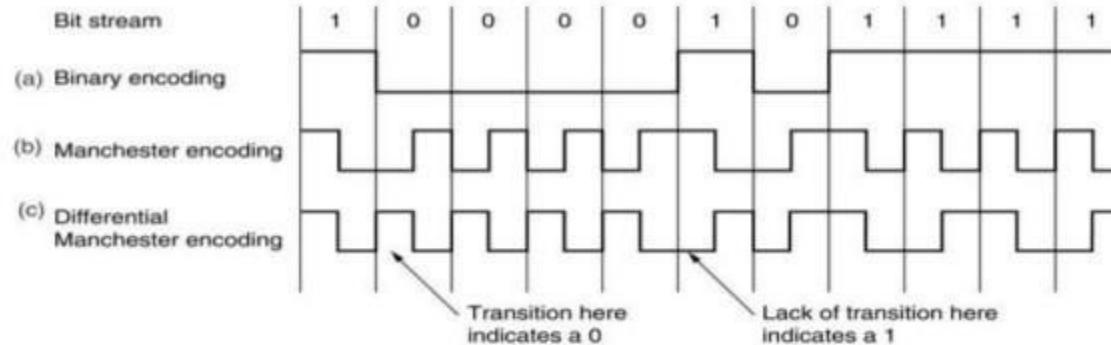
**Starting and ending flags, with bit stuffing** allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character. It works like this. Each frame begins and ends with a special bit pattern, 01111110 (in fact, a flag byte). Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data.

When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically de-stuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110.

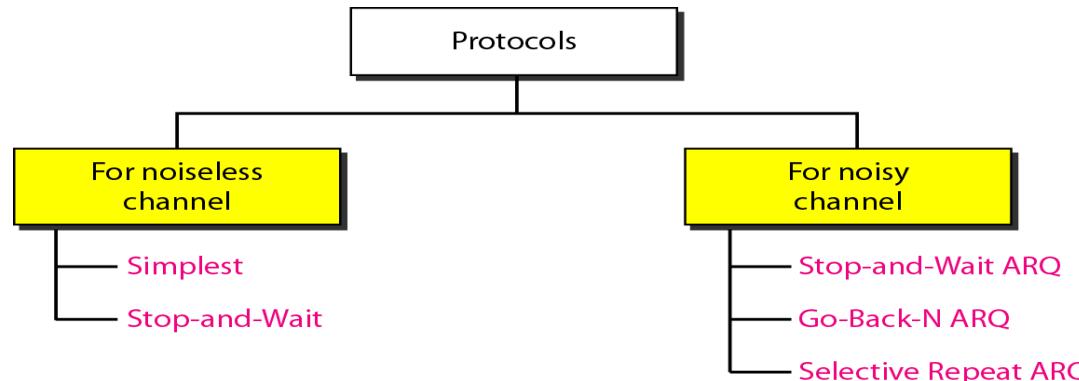
(a) 011011111111111111110010

(b) 0110111110111110111111010010  
            ↑  
            Stuffed bits

high and low-low are not used for data but are used for delimiting frames in some protocols.

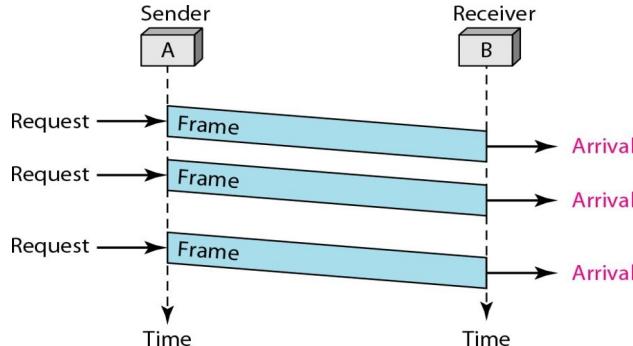


As a final note on framing, many data link protocols use combination of a character count with one of the other methods for extra safety. When a frame arrives, the count field is used to locate the end of the frame. Only if the appropriate delimiter is present at that position and the checksum is correct is the frame accepted as valid. Otherwise, the input stream is scanned for the next delimiter



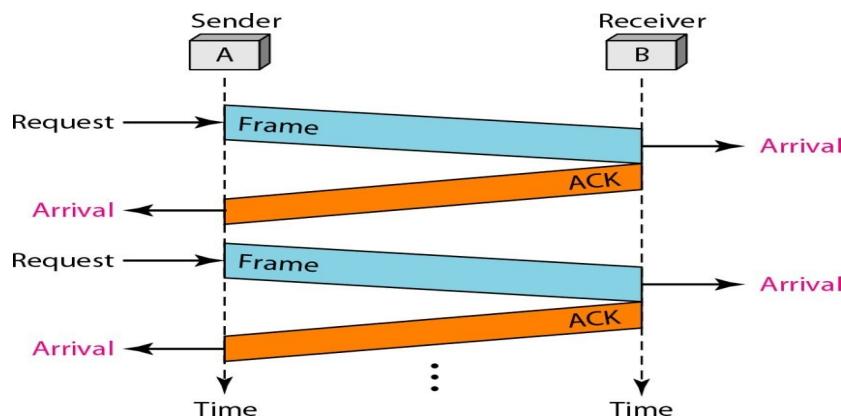
## ELEMENTARY DATA LINK PROTOCOLS

### Simplest Protocol



It is very simple. The sender sends a sequence of frames without even thinking about the receiver. Data are transmitted in one direction only. Both sender & receiver always ready. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link layers never damages or loses frames. This thoroughly unrealistic protocol, which we will nickname “Utopia,” .The utopia protocol is unrealistic because **it does not handle either flow control or error correction**

### Stop-and-wait Protocol



It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame

It is Stop-and-Wait Protocol because the sender sends one frame, stops until it

1 Stop-and-Wait Automatic Repeat

Request

2 Go-Back-N Automatic Repeat

Request

### **3 Selective Repeat Automatic Repeat Request**

#### **1 Stop-and-Wait Automatic Repeat Request**

To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.

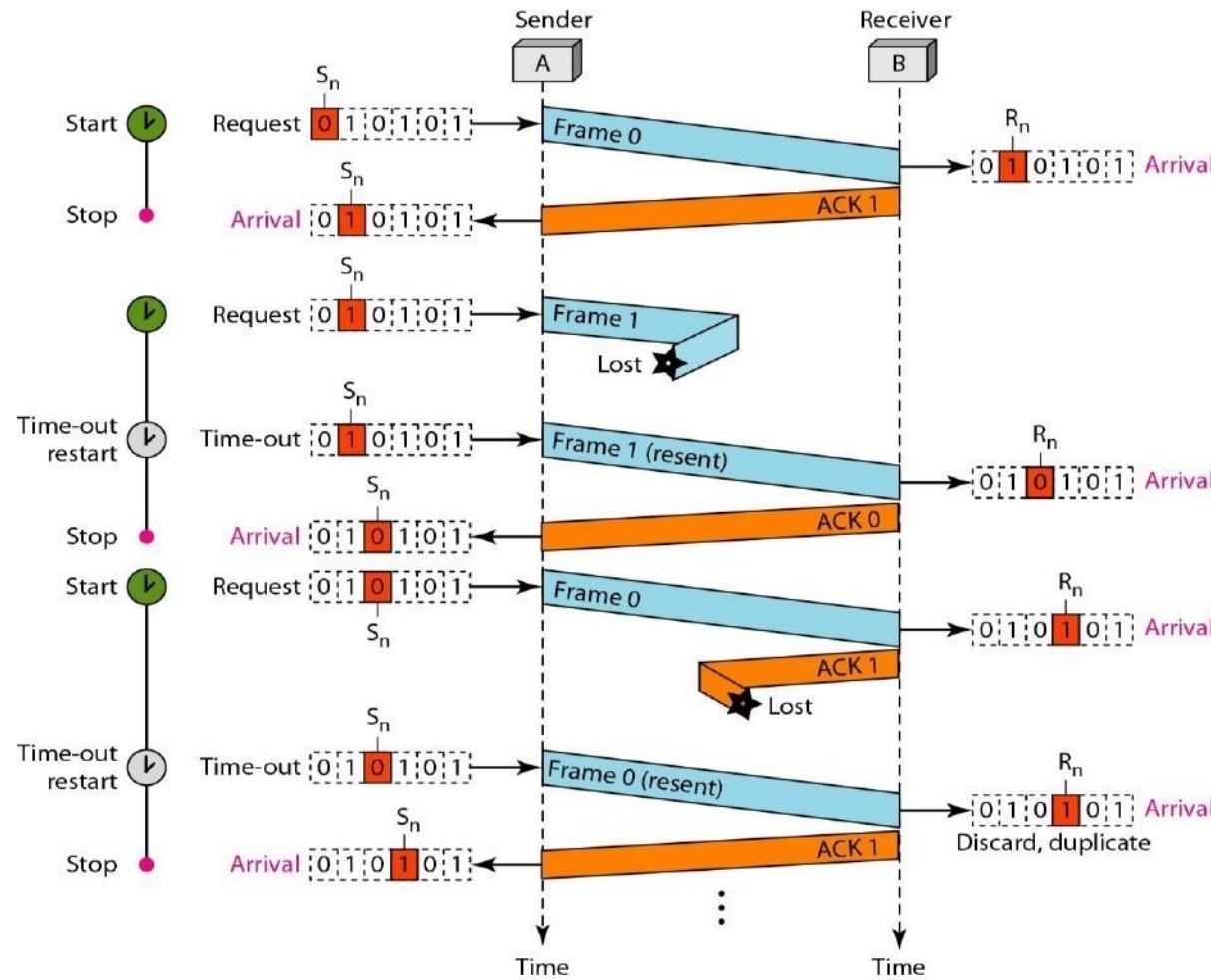
Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated

The lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, how can the sender know which frame to resend? To remedy this problem, the sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires

**In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.**

**In Stop-and-Wait ARQ, the acknowledgment number always announces**



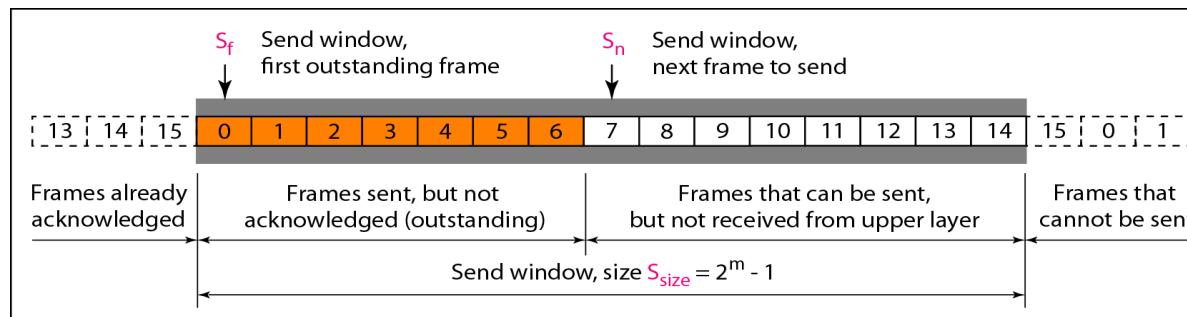
### Bandwidth Delay Product:

Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

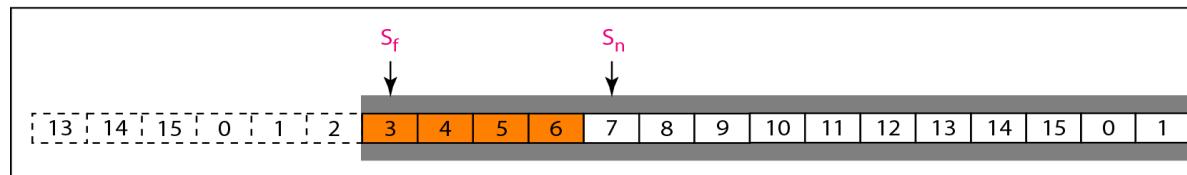
$$(1 \times 10^6) \times (20 \times 10^{-3}) = 20,000 \text{ bits}$$

The first is called Go-Back-N Automatic Repeat. In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

**In the Go-Back-N Protocol, the sequence numbers are modulo  $2^m$ , where  $m$  is the size of the sequence number field in bits.** The sequence numbers range from 0 to  $2^m - 1$ . For example, if  $m$  is 4, the only sequence numbers are 0 through 15 inclusive.



a. Send window before sliding



b. Send window after sliding

The **sender window** at any time divides the possible sequence numbers into four regions.

The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them.

The second region, colored in Figure (a), defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames.

The third region, white in the figure, defines the range of sequence

acknowledged, so the window has slide to the right three slots. Note that the value of  $S_f$  is 3 because frame 3 is now the first outstanding frame. **The send window can slide one or more slots when a valid acknowledgment arrives.**

**Receiver window:** variable  $R_n$  (receive window, next frame expected) .

The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of  $R_n$  is accepted and acknowledged. The receive window also slides, but only one slot at a time. When a correct frame is received (and a frame is received only one at a time), the window slides.( see below figure for receiving window)

The receive window is an abstract concept defining an imaginary box of size 1 with one single variable  $R_n$ . The window slides when a correct frame has arrived; sliding occurs one slot at a time

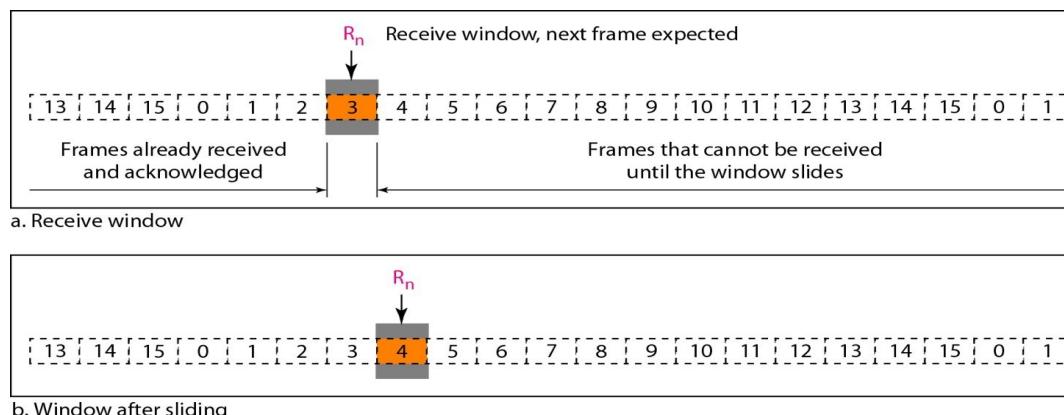
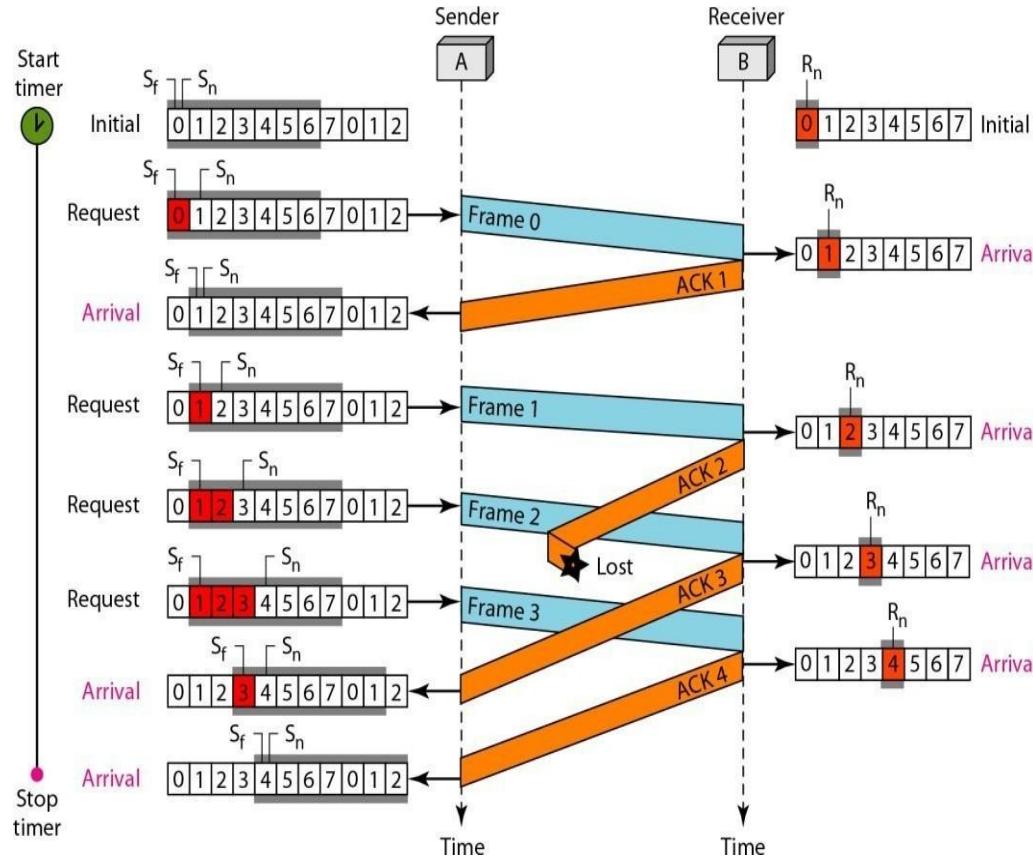


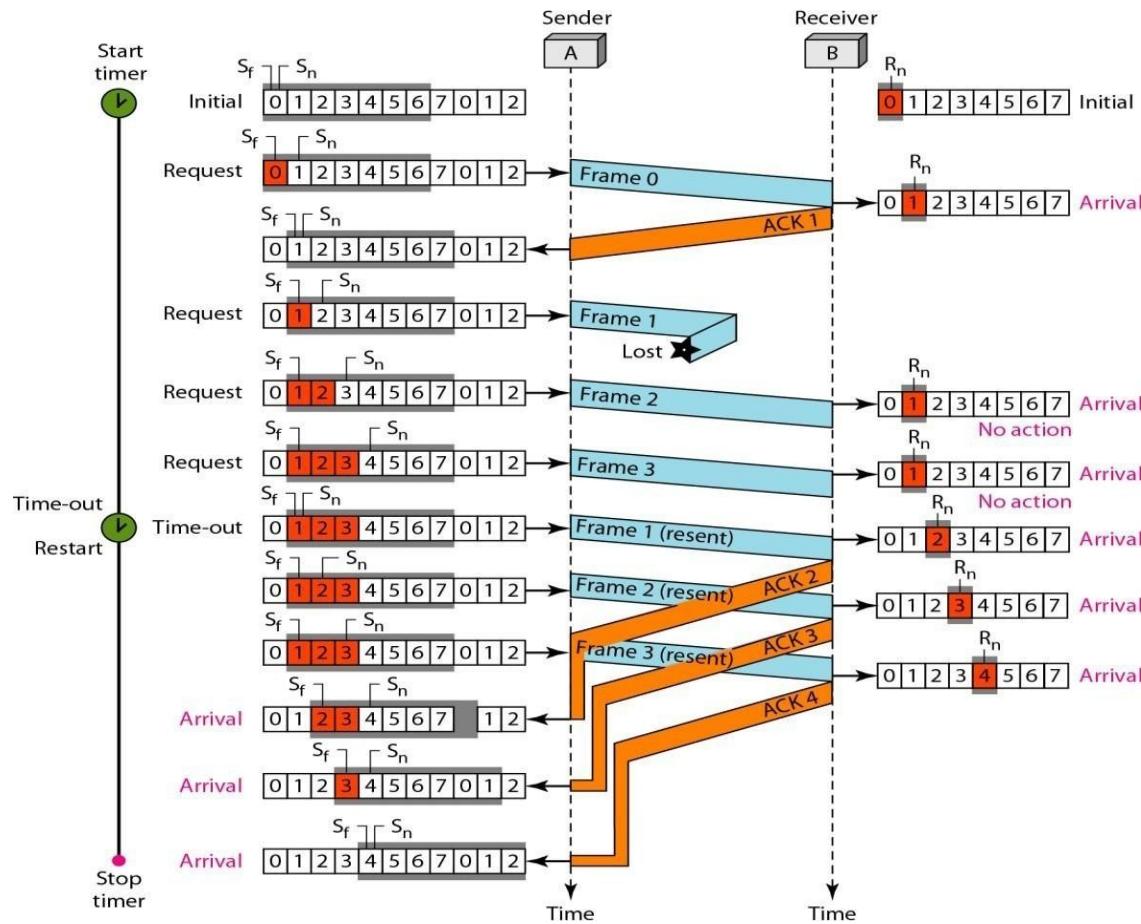
Fig: Receiver window (before sliding (a), After sliding (b))

## Resending a Frame

When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3,4,5, and 6 again. That is why the protocol is called *Go-Back-N* ARQ.

Below figure is an example(if ack lost) of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost





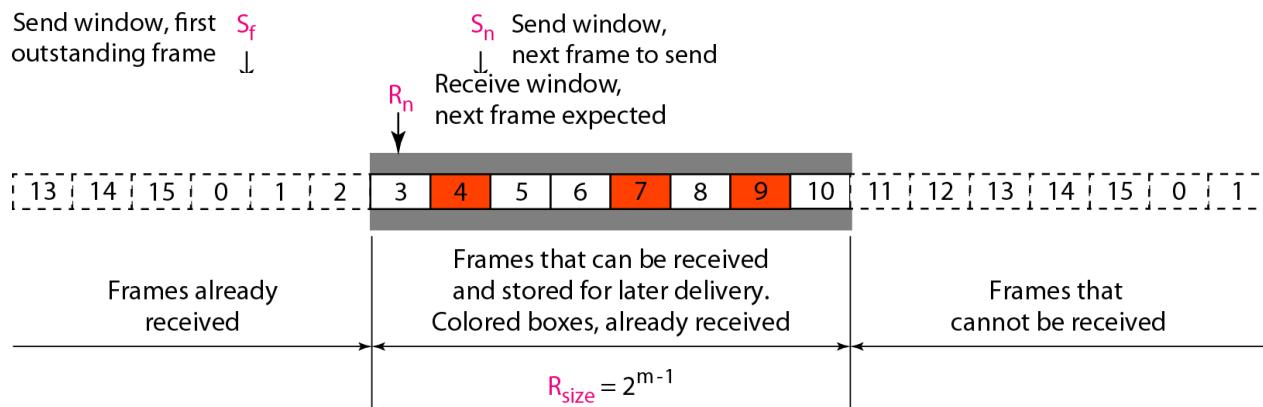
Stop-and-Wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1.

### **3 Selective Repeat Automatic Repeat Request**

In Go-Back-N ARQ, The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link.

In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission.

For noisy links, there is another mechanism that does not resend  $N$  frames



### Receiver window

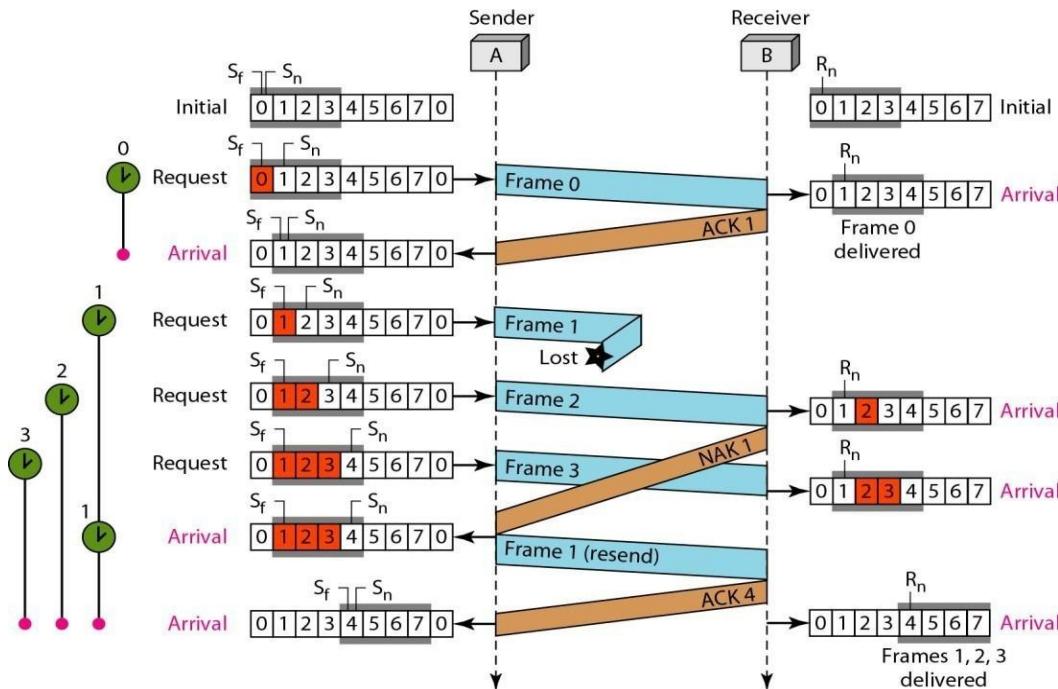
The receiver window in Selective Repeat is totally different from the one in Go Back-N. First, the size of the receive window is the same as the size of the send window ( $2^{m-1}$ ).

The Selective Repeat Protocol allows as many frames as the size of the receiver window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered. However the receiver never delivers packets out of order to the network layer. Above Figure shows the receive window. Those slots inside the window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of  $2^m$

### **Delivery of Data in Selective Repeat ARQ:**





## Differences between Go-Back N & Selective Repeat

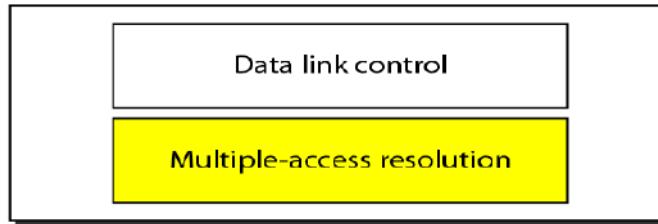
One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives.

There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window. After the first arrival, there was only one frame and it started from the beginning of the window. After the last arrival, there are three frames and the first one starts from the beginning of the window.

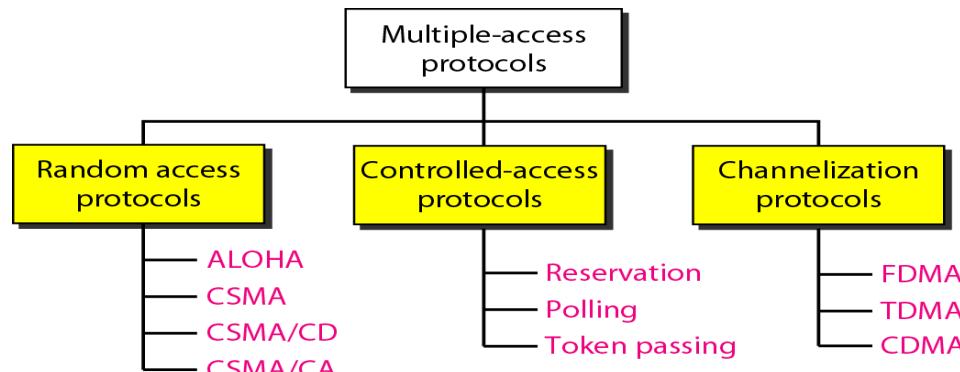
Another important point is that a NAK is sent.

The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to

## Data link layer



The upper sub layer that is responsible for flow and error control is called the logical link control (LLC) layer; the lower sub layer that is mostly responsible for multiple access resolution is called the media access control (MAC) layer. When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link.



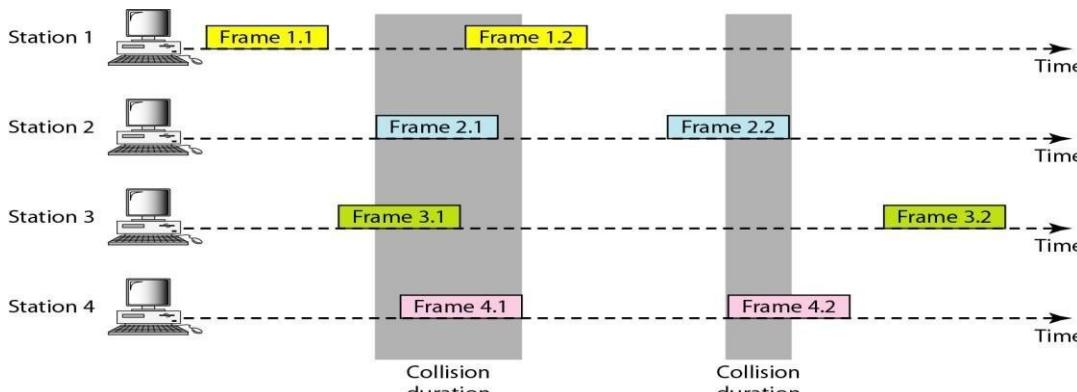
Taxonomy of multiple-access protocols

## 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. Stations compete with one another to access the medium. That is why these methods are also called *contention* methods.

## ALOHA

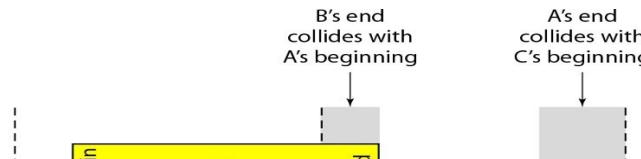


### Frames in a pure ALOHA network

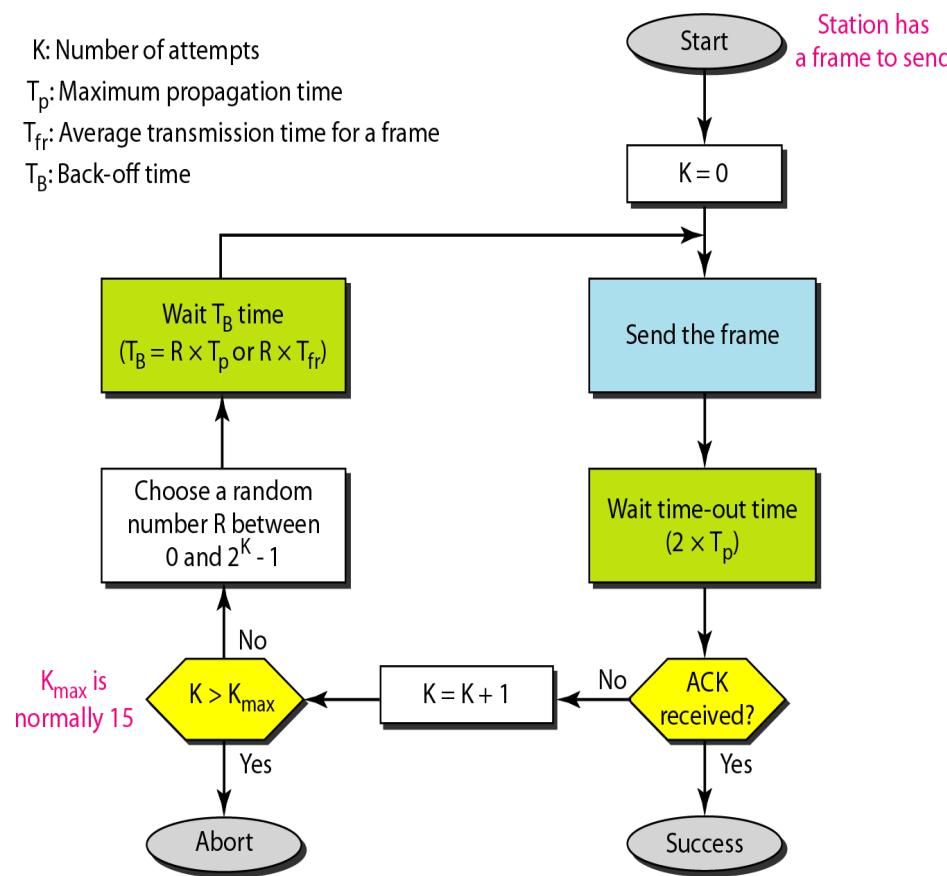
In pure ALOHA, the stations transmit frames whenever they have data to send.

- 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.
- Therefore pure ALOHA dictates that when time-out period passes, each station must wait for a random amount of time before resending its frame. This randomness will help avoid more collisions.

**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.



Looking at Figure, we see that the vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time. Pure ALOHA vulnerable time =  $2 \times T_{fr}$



### *Procedure for pure ALOHA protocol*

#### **Example**

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?

#### **Solution**

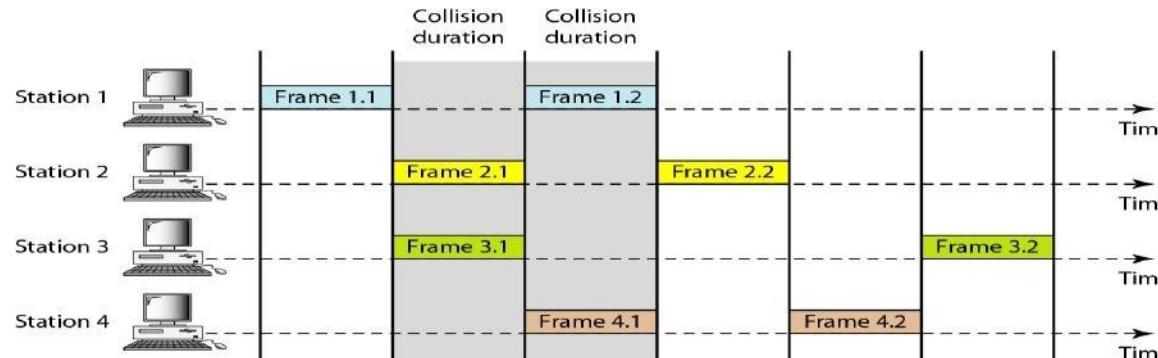
millisecond. The load is 1. In this case  $S = G \times e^{-2^G}$  or  $S = 0.135$  (13.5 percent). This means that the throughput is  $1000 \times 0.135 = 135$  frames. Only 135 frames out of 1000 will probably survive.

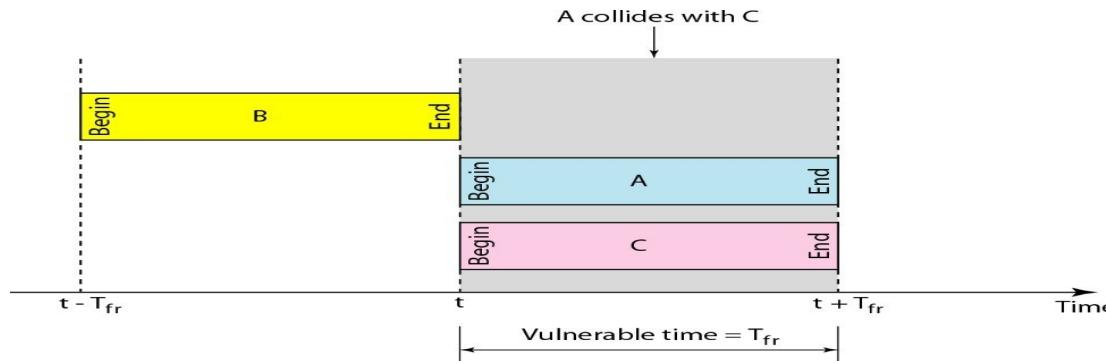
- b. If the system creates 500 frames per second, this is  $(1/2)$  frame per millisecond. The load is  $(1/2)$ . In this case  $S = G \times e^{-2^G}$  or  $S = 0.184$  (18.4 percent). This means that the throughput is  $500 \times 0.184 = 92$  and that only 92 frames out of 500 will probably survive. Note that this is the maximum throughput case, percentage wise.
- c. If the system creates 250 frames per second, this is  $(1/4)$  frame per millisecond. The load is  $(1/4)$ . In this case  $S = G \times e^{-2^G}$  or  $S = 0.152$  (15.2 percent). This means that the throughput is  $250 \times 0.152 = 38$ . Only 38 frames out of 250 will probably survive.

## **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 efficiency of pure ALOHA.

In slotted ALOHA we divide the time into slots of  $T_{fr}$  s and force the station to send only at the beginning of the time slot. Figure 3 shows an example of frame collisions in slotted ALOHA





The throughput for slotted ALOHA is  $S = G \times e^{-G}$ . The maximum throughput  $S_{max} = 0.368$  when  $G = 1$ .

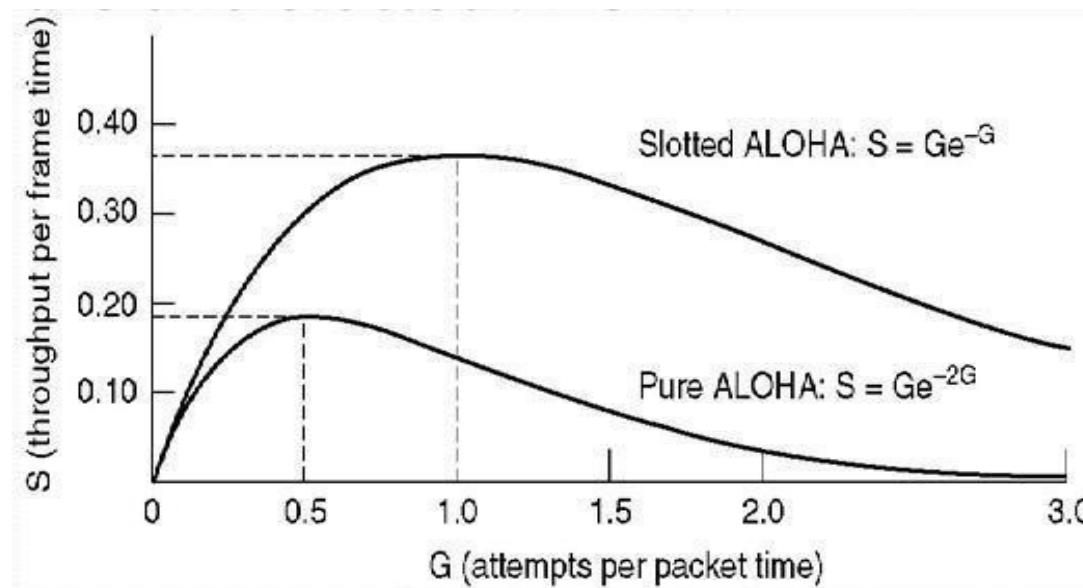
A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200- Kbps bandwidth. Find the throughput if the system (all stations together) produces

- a. 1000 frames per second b. 500 frames per second c. 250 frames per second

### Solution

This situation is similar to the previous exercise except that the network is using slotted ALOHA instead of pure ALOHA. The frame transmission time is  $200/200$  kbps or 1 ms.

- a. In this case  $G$  is 1. So  $S = G \times e^{-G}$  or  $S = 0.368$  (36.8 percent). This means that the throughput is  $1000 \times 0.0368 = 368$  frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.
- b. Here  $G$  is  $1/2$  In this case  $S = G \times e^{-G}$  or  $S = 0.303$  (30.3 percent). This means that the throughput is  $500 \times 0.0303 = 151$ . Only 151 frames out of 500 will probably survive.
- c. Now  $G$  is  $1/4$ . In this case  $S = G \times e^{-G}$  or  $S = 0.195$  (19.5 percent). This means that the throughput is  $250 \times 0.195 = 49$ . Only 49 frames out of 250 will probably survive.



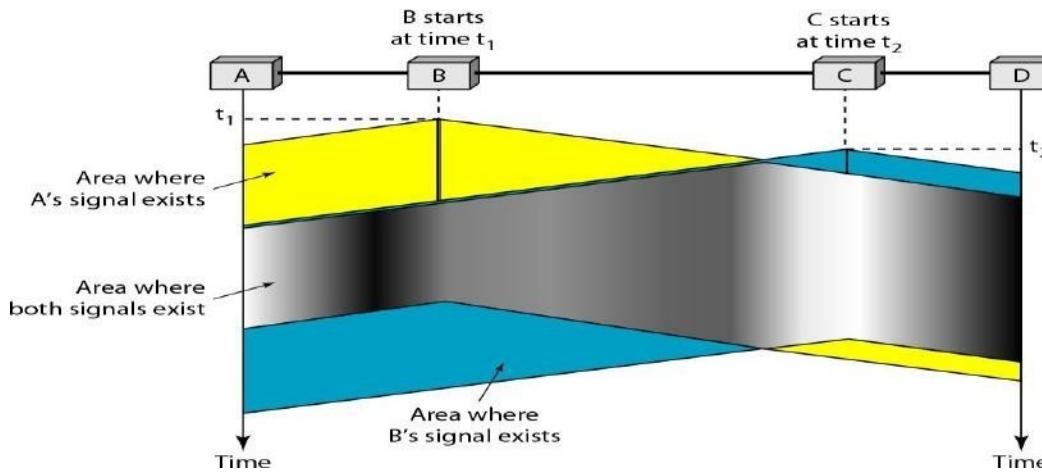
## **Carrier Sense Multiple Access (CSMA)**

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. In other words, CSMA is based on the principle "sense before transmit" or "listen before talk."

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in below Figure. Stations are connected to a shared channel (usually a dedicated medium).

The possibility of collision still exists because of propagation delay; station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

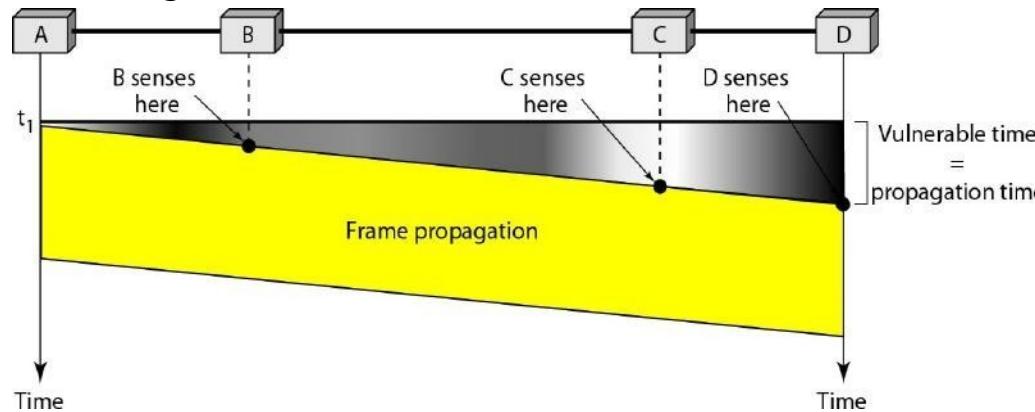
At time  $t'$  station B senses the medium and finds it idle, so it sends a



### Space/time model of the collision in CSMA

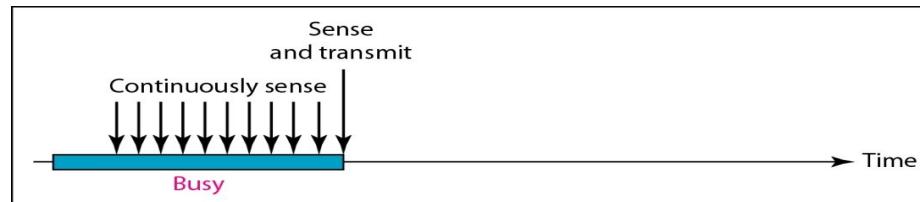
#### Vulnerable Time

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. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending.

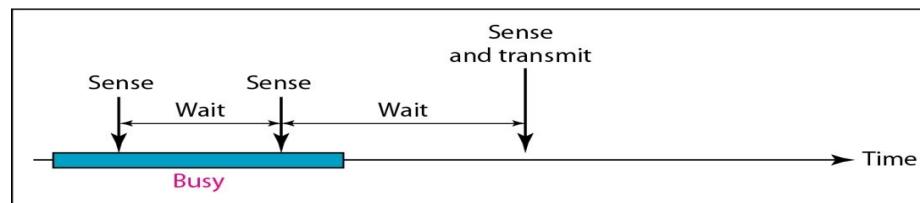


#### Vulnerable time in CSMA

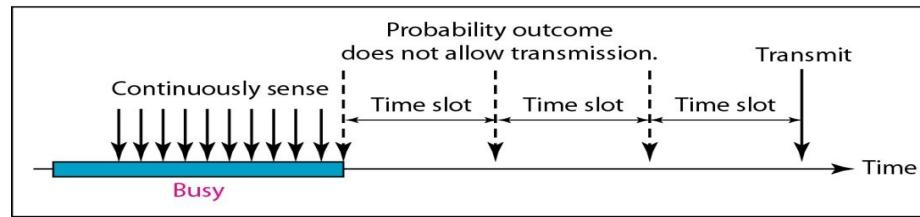
#### Persistence Methods



a. 1-persistent



b. Nonpersistent

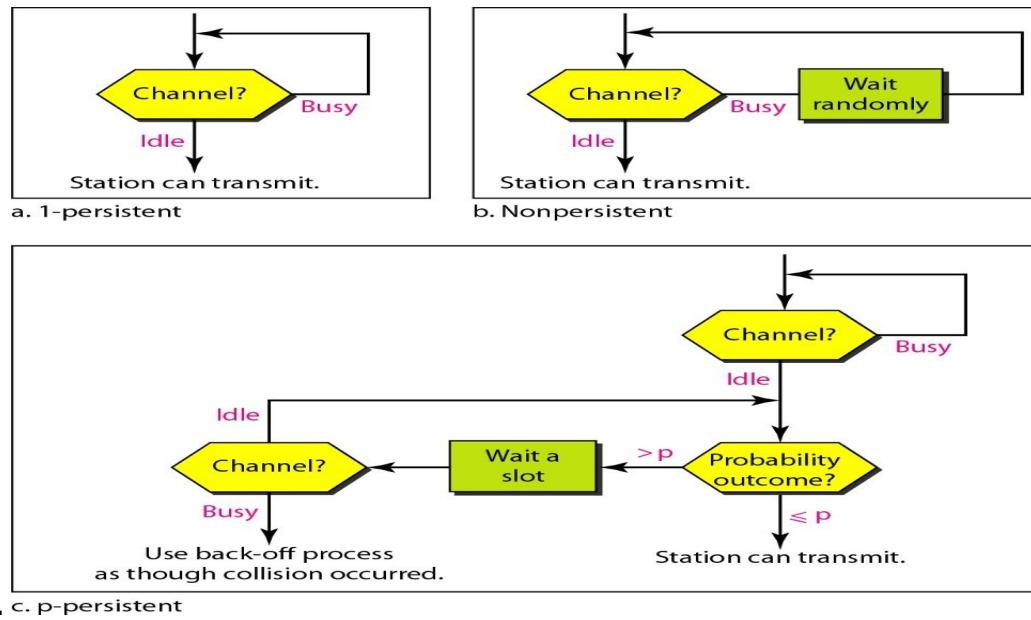


c. p-persistent

**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. However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to send.

**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

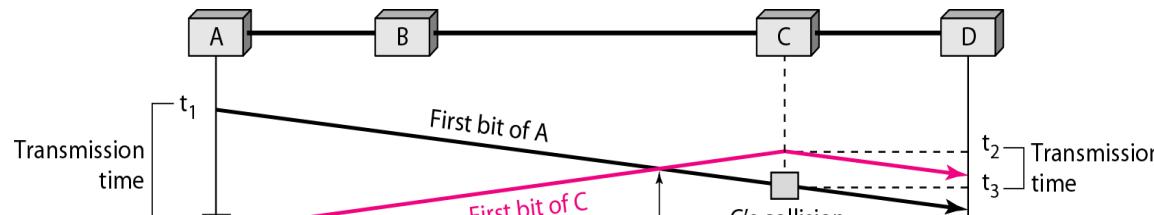


## Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

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.

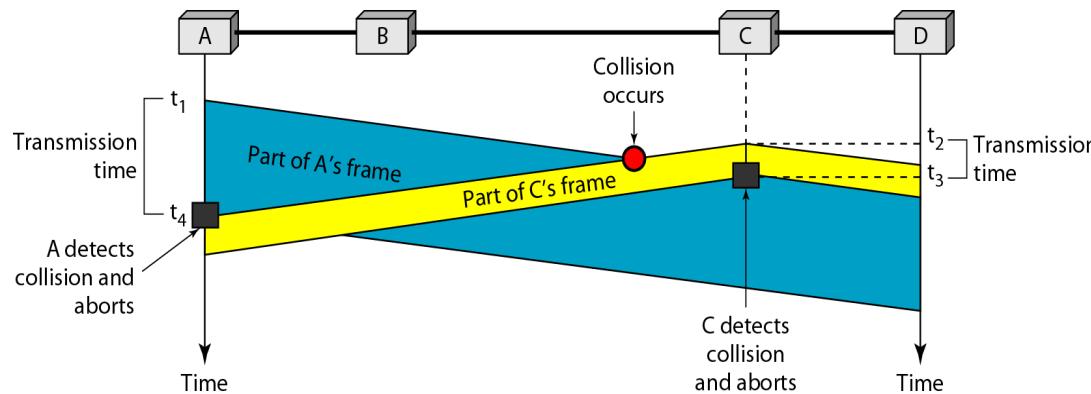
To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In below Figure, stations A and C are involved in the collision.



transmits for the duration  $t_4 - t_1$ ; C transmits for the duration  $t_3 - t_2$ .

### Minimum Frame Size

For CSMA/CD to work, we need a restriction on the frame size. Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission. This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the frame transmission time  $T_{fr}$  must be at least two times the maximum propagation time  $T_p$ . To understand the reason, let us think about the worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time  $T_p$  to reach the second, and the effect of the collision takes another time  $T_p$  to reach the first. So the requirement is that the first station must still be transmitting after  $2T_p$ .



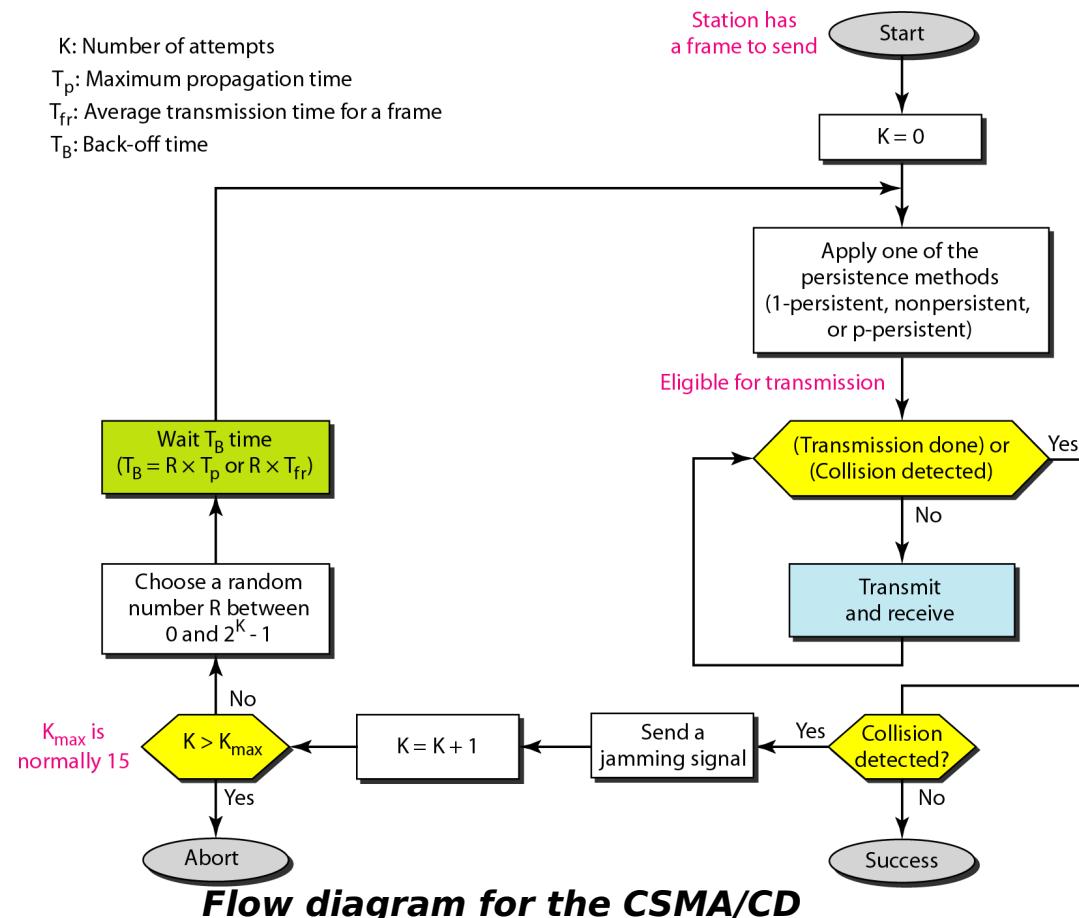
## Collision and abortion in CSMA/CD

$K$ : Number of attempts

$T_p$ : Maximum propagation time

$T_{fr}$ : Average transmission time for a frame

$T_B$ : Back-off time



**Flow diagram for the CSMA/CD**

### PROBLEM

A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6  $\mu$ s, what is the minimum size of the frame?

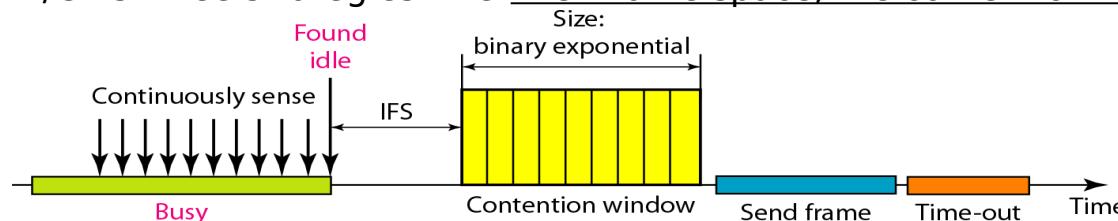
### SOL

The frame transmission time is  $T_{fr} = 2 \times T_p = 51.2 \mu$ s. This means, in the worst case, a station needs to transmit for a period of 51.2  $\mu$ s to detect the

The third difference is the sending of a short jamming signal that enforces the collision in case other stations have not yet sensed the collision.

## 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

### Timing in CSMA/CA

#### Inter frame Space (IFS)

First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the inter frame space or IFS.

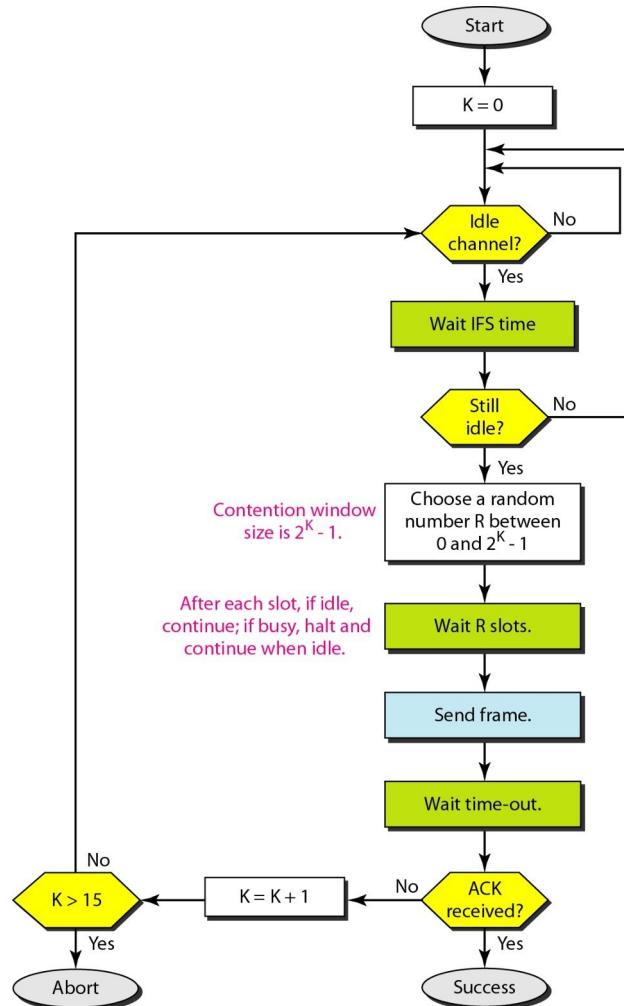
Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station's signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the contention time. The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned shorter IFS has a higher priority.

In CSMA/CA, the IFS can also be used to define the priority of a station or a frame.

In CSMA/CA, if the station finds the channel busy, it does not restart the timer of the contention window; it stops the timer and restarts it when the channel becomes idle.

### Acknowledgment

With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.



parameter, waits for the back off time and re senses the line

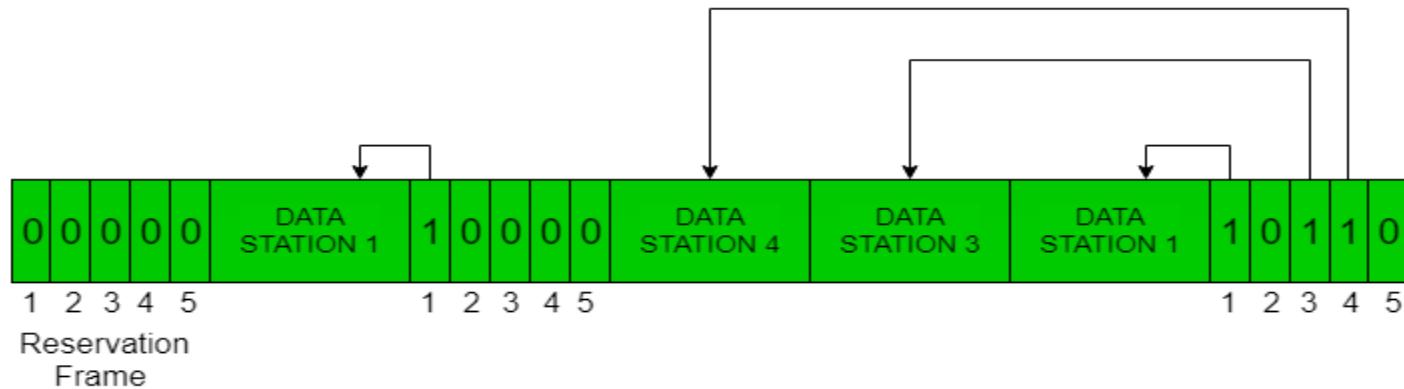
## Controlled Access Protocols

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:

- 1 Reservation
- 2 Polling
- 3 Token Passing

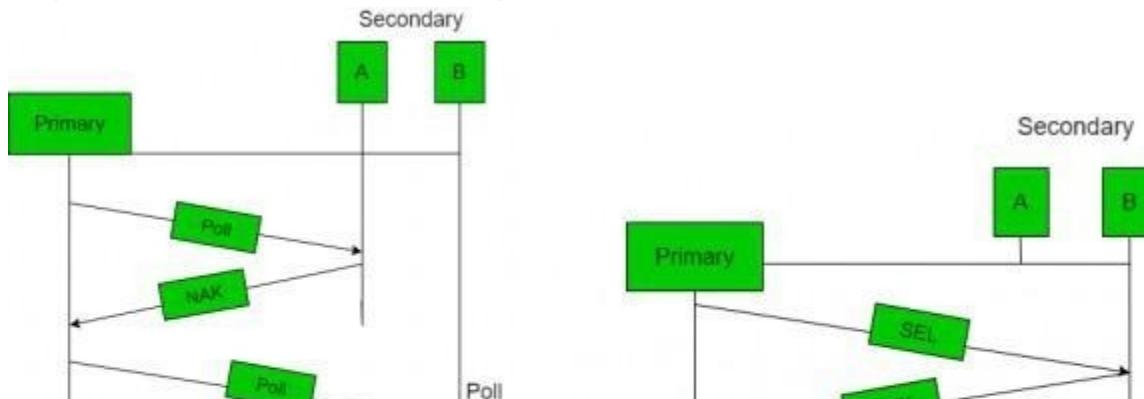
### Reservation

- 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.
- If there are M stations, the reservation interval is divided into M slots, and each station has one slot.
- Suppose if station 1 has a frame to send, it transmits 1 bit during the slot
  1. No other station is allowed to transmit during this slot.
- In general,  $i^{\text{th}}$  station may announce that it has a frame to send by inserting a 1 bit into  $i^{\text{th}}$  slot. After all N slots have been checked, each station knows which stations wish to transmit.
- The stations which have reserved their slots transfer their frames in that order.
- After data transmission period, next reservation interval begins.

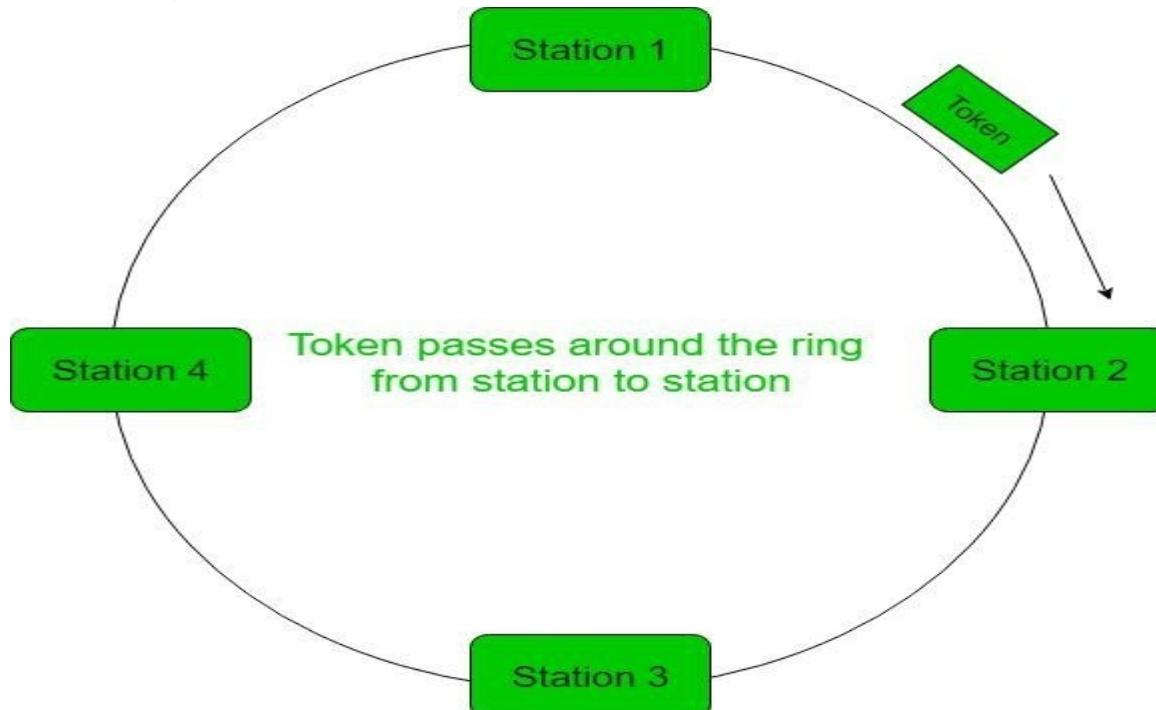


## Polling

- 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) message is sent back.
- Problems include high overhead of the polling messages and high dependence on the reliability of the controller.



- In Token ring, token is passed from one station to another adjacent station in the ring whereas incase 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.
- There exists problems like duplication of token or token is lost or insertion of new station, removal of a station, which need be tackled for correct and reliable operation of this scheme.



## Error Detection

additional bits are added to facilitate detection of errors. Some popular techniques for error detection are:

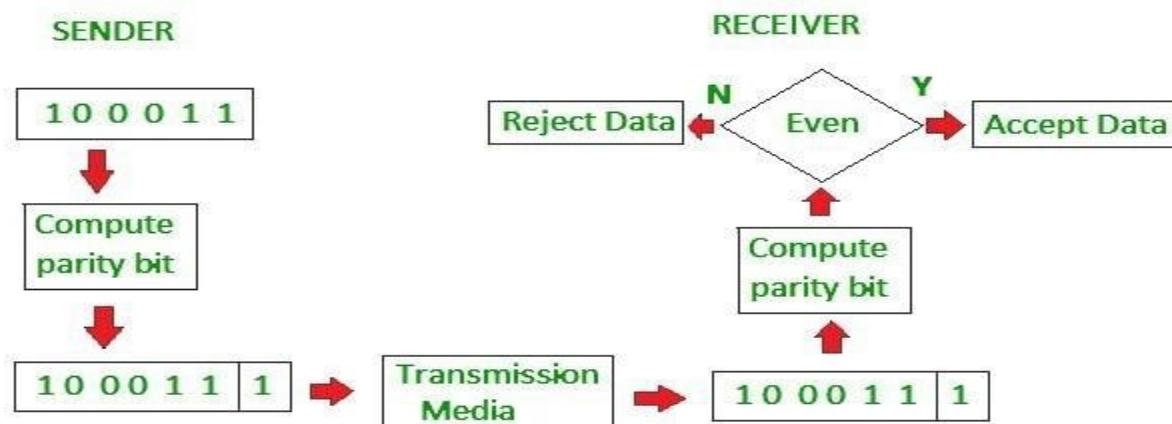
1. Simple Parity check
2. Two-dimensional Parity check
3. Checksum
4. Cyclic redundancy check

### **Simple Parity check**

Blocks of data from the source are subjected to a check bit or parity bit generator form, where a parity of : 1 is added to the block if it contains odd number of 1's, and

0 is added if it contains even number of 1's

This scheme makes the total number of 1's even, that is why it is called even parity checking.



### **Two-dimensional Parity check**

### Original Data

10011001	11100010	00100100	10000100
----------	----------	----------	----------

### Row parities

10011001	0
11100010	0
00100100	0
10000100	0
11011011	0

Column parities  
→

100110010	111000100	001001000	100001000	110110110
-----------	-----------	-----------	-----------	-----------

### Data to be sent

## Checksum

- In checksum error detection scheme, the data is divided into  $k$  segments each of  $m$  bits.
- In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.

### Original Data

10011001	11100010	00100100	10000100
----------	----------	----------	----------

1  
 $k=4, m=8$

2

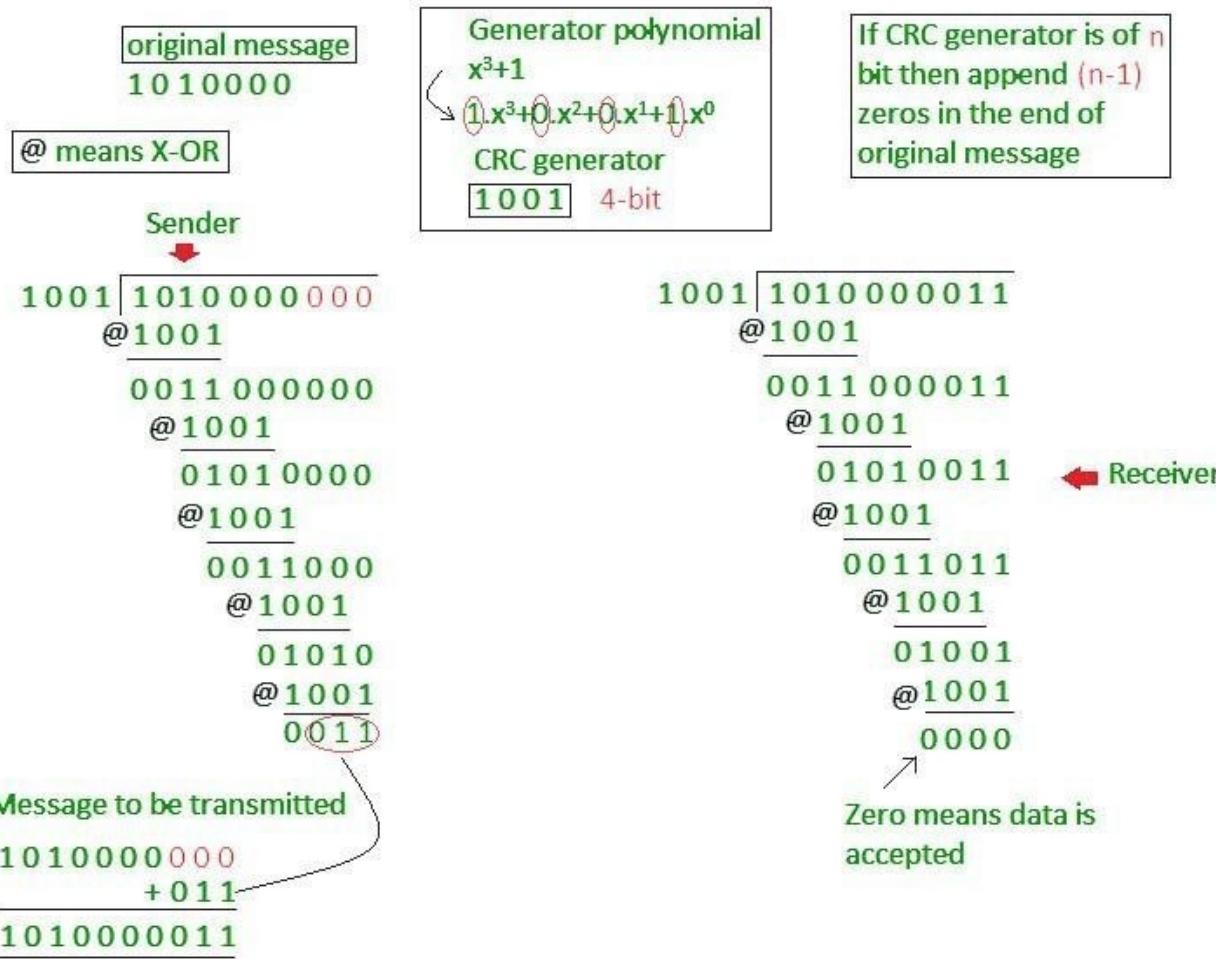
3

4

Receiver

Sender

1 10011001  
2 11100010



- Unlike checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.

For correcting the errors, one has to know the exact position of the error. For example, If we want to calculate a single-bit error, the error correction code will determine which one of seven bits is in error. To achieve this, we have to add some additional redundant bits.

Suppose  $r$  is the number of redundant bits and  $d$  is the total number of the data bits. The number of redundant bits  $r$  can be calculated by using the formula:

$$2^r \geq d+r+1$$

The value of  $r$  is calculated by using the above formula. For example, if the value of  $d$  is 4, then the possible smallest value that satisfies the above relation would be 3.

To determine the position of the bit which is in error, a technique developed by R.W Hamming is Hamming code which can be applied to any length of the data unit and uses the relationship between data units and redundant units.

### **Hamming Code**

**Parity bits:** The bit which is appended to the original data of binary bits so that the total number of 1s is even or odd.

**Even parity:** To check for even parity, if the total number of 1s is even, then the value of the parity bit is 0. If the total number of 1s occurrences is odd, then the value of the parity bit is 1.

**Odd Parity:** To check for odd parity, if the total number of 1s is even, then the value of parity bit is 1. If the total number of 1s is odd, then the value of parity bit is 0.

### **Algorithm of Hamming code:**

An information of ' $d$ ' bits are added to the redundant bits ' $r$ ' to form  $d+r$ .

The location of each of the  $(d+r)$  digits is assigned a decimal value.

The ' $r$ ' bits are placed in the positions  $1, 2, \dots, 2^{k-1}$

At the receiving end, the parity bits are recalculated. The decimal value of the parity bits determines the position of an error.

### **Relationship b/w Error position & binary number.**

$$2^r \geq 4+r+1$$

Therefore, the value of r is 3 that satisfies the above relation.  
Total number of bits = d+r = 4+3 = 7;

### Determining the position of the redundant bits

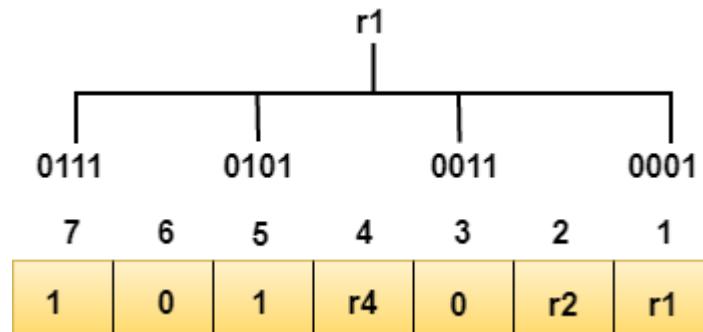
The number of redundant bits is 3. The three bits are represented by r1, r2, r4. The position of the redundant bits is calculated with corresponds to the raised power of 2. Therefore, their corresponding positions are 1,  $2^1$ ,  $2^2$ .  
The position of r1 = 1, The position of r2 = 2 , The position of r4 = 4

Representation of Data on the addition of parity bits:

7	6	5	4	3	2	1
1	0	1	r4	0	r2	r1

### Determining the Parity bits

**Determining the r1 bit:** The r1 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the first position.

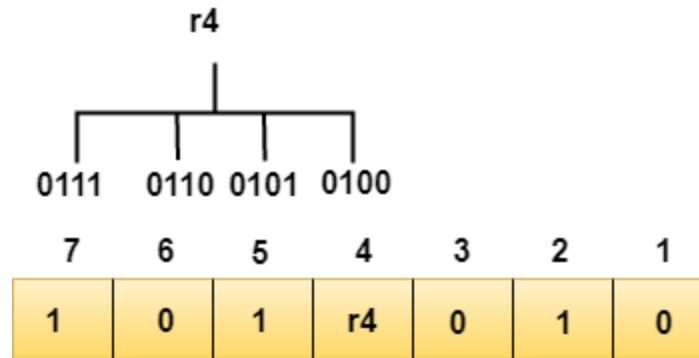


We observe from the above figure that the bit position that includes 1 in the first position are 1, 3, 5, 7. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r1 is even, therefore, the value of the r1 bit is 0.

**Determining r2 bit:** The r2 bit is calculated by performing a parity check on the

bit positions. The total number of 1 at these bit positions corresponding to r2 is odd, therefore, the value of the r2 bit is 1.

**Determining r4 bit:** The r4 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the third position.



We observe from the above figure that the bit positions that includes 1 in the third position are 4, 5, 6, 7. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r4 is even, therefore, the value of the r4 bit is 0.

**Data transferred is given below:**

7	6	5	4	3	2	1
1	0	1	0	0	1	0

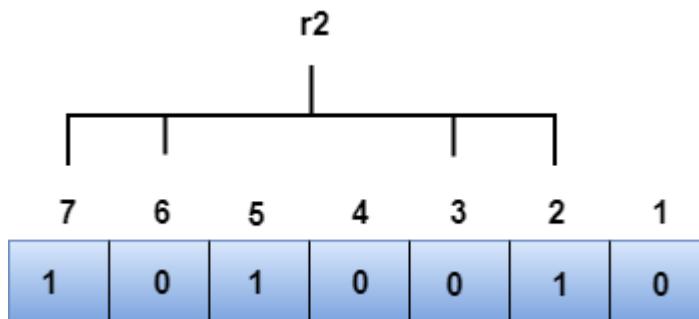
Suppose the 4th bit is changed from 0 to 1 at the receiving end, then parity bits are recalculated.

**R1 bit**

The bit positions of the r1 bit are 1,3,5,7

r1

|

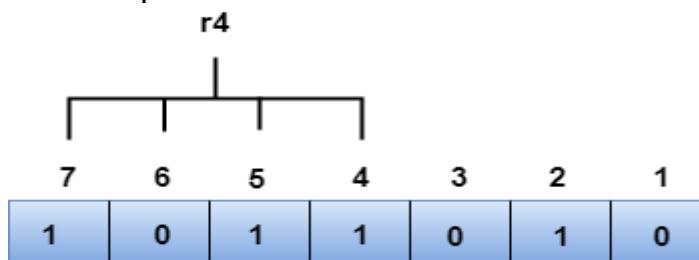


We observe from the above figure that the binary representation of r2 is 1001.

Now, we perform the even-parity check, the total number of 1s appearing in the r2 bit is an even number. Therefore, the value of r2 is 0.

#### R4 bit

The bit positions of r4 bit are 4,5,6,7.



We observe from the above figure that the binary representation of r4 is 1011.

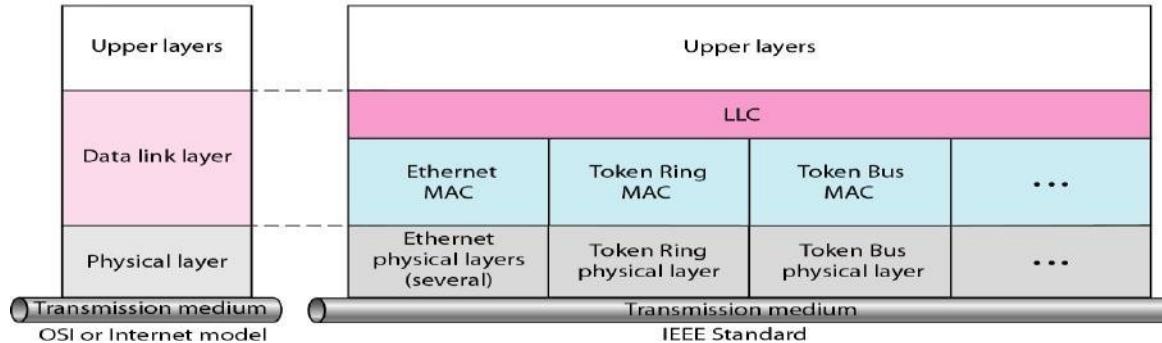
Now, we perform the even-parity check, the total number of 1s appearing in the r4 bit is an odd number. Therefore, the value of r4 is 1.

The binary representation of redundant bits, i.e., r4r2r1 is 100, and its corresponding decimal value is 4. Therefore, the error occurs in a 4th bit position. The bit value must be changed from 1 to 0 to correct the error.

## Wired LANs: Ethernet

In 1985, the Computer Society of the IEEE started a project, called Project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers. Project 802 is a way of specifying functions of the

LLC: Logical link control  
MAC: Media access control



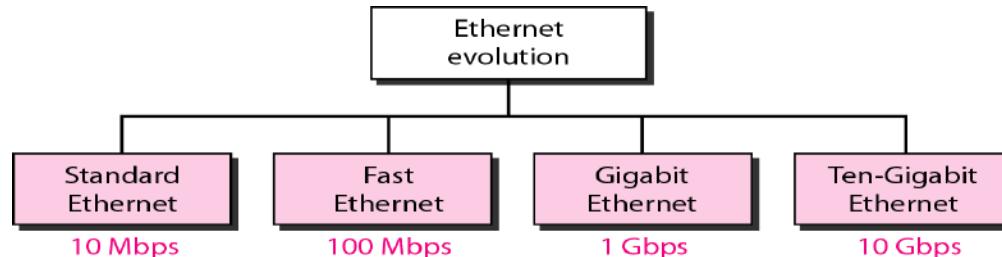
### *IEEE standard for LANs*

## **STANDARD ETHERNET**

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations.

Standard Ethernet (10 Mbps), Fast Ethernet (100 Mbps), Gigabit Ethernet (1 Gbps), and Ten-Gigabit Ethernet (10 Gbps),

We briefly discuss the Standard (or traditional) Ethernet in this section



*Ethernet evolution through four generations*

## **MAC Sublayer**

In Standard Ethernet, the MAC sublayer governs the operation of the access method. It also frames data received from the upper layer and passes them to the physical layer.

## **Frame Format**

## 802.3 MAC frame

Preamble. The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0s and 1s that alerts the receiving system to the coming frame and enables it to synchronize its input timing. The pattern provides only an alert and a timing pulse. The 56-bit pattern allows the stations to miss some bits at the beginning of the frame. The preamble is actually added at the physical layer and is not (formally) part of the frame.

Start frame delimiter (SFD). The second field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.

Destination address (DA). The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet.

Source address (SA). The SA field is also 6 bytes and contains the physical address of the sender of the packet.

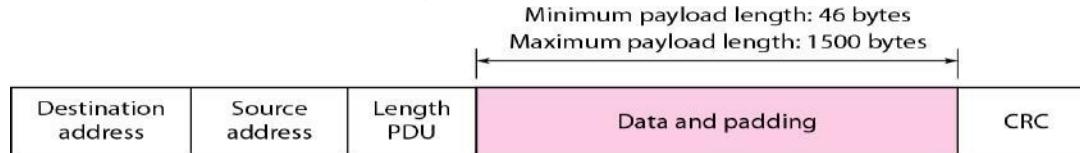
Length or type. This field is defined as a type field or length field. The original Ethernet used this field as the type field to define the upper-layer protocol using the MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field. Both uses are common today.

Data. This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes.

CRC. The last field contains error detection information, in this case a CRC-32

### **Frame Length**

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame, as shown in below Figure



the maximum length of the payload is 1500 bytes.

The maximum length restriction has two historical reasons.

First, memory was very expensive when Ethernet was designed: a maximum length restriction helped to reduce the size of the buffer.

Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

### **Addressing**

The Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between the bytes.

***Example of an Ethernet address in hexadecimal notation***

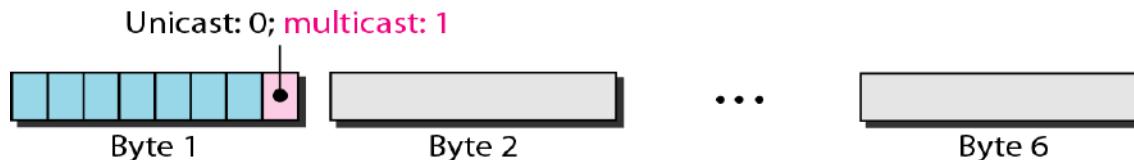
**06 : 01 : 02 : 01 : 2C : 4B**

|

**6 bytes = 12 hex digits = 48 bits**

Unicast, Multicast, and Broadcast Addresses A source address is always a unicast address-the frame comes from only one station. The destination address, however, can be **unicast, multicast, or broadcast**. Below Figure shows how to distinguish a unicast address from a multicast address.

If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.



to send 512 bits. This means that the actual slot time depends on the data rate; for traditional 10-Mbps Ethernet it is 51.2 micro sec.

Slot Time and Maximum Network Length There is a relationship between the slot time and the maximum length of the network (collision domain). It is dependent on the propagation speed of the signal in the particular medium.

In most transmission media, the signal propagates at  $2 \times 10^{-8}$  m/s (two-thirds of the rate for propagation in air).

For traditional Ethernet, we calculate

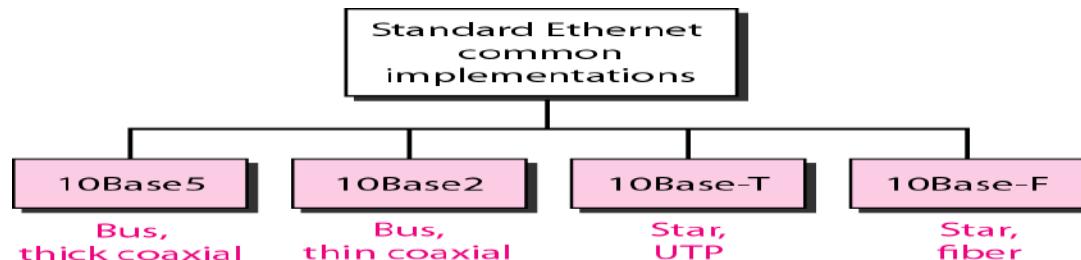
$$\text{MaxLength} = \text{PropagationSpeed} \times (\text{SlotTime}/2)$$

$$\text{MaxLength} = (2 \times 10^8) \times (51.2 \times 10^{-6})/2 = 5120\text{m}$$

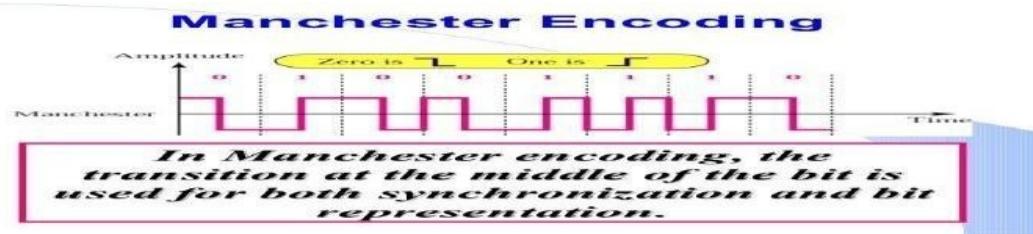
Of course, we need to consider the delay times in repeaters and interfaces, and the time required to send the jam sequence. These reduce the maximum-length of a traditional Ethernet network to 2500 m, just 48 percent of the theoretical calculation.  $\text{MaxLength}=2500\text{ m}$

## **Physical Layer**

The Standard Ethernet defines several physical layer implementations; four of the most common, are shown in Figure



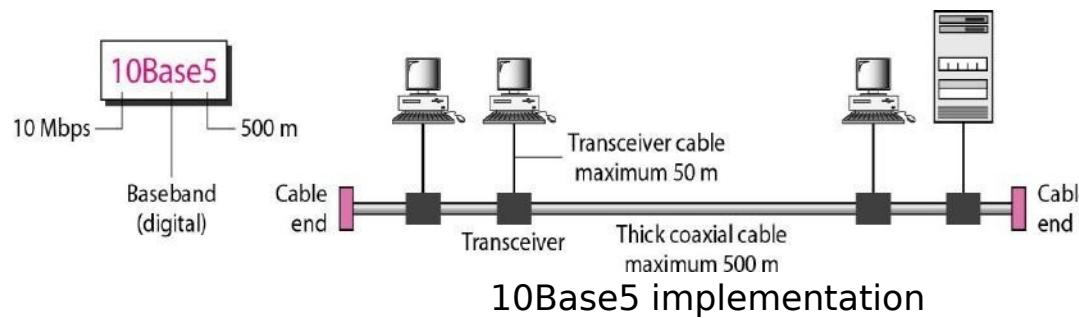
In Manchester encoding, the transition at the middle of the bit is used for synchronization



### IOBase5: Thick Ethernet

The first implementation is called **10Base5, thick Ethernet, or Thicknet**.

IOBase5 was the first Ethernet specification to use a bus topology with an external **transceiver** (transmitter/receiver) connected via a tap to a thick coaxial cable. Figure shows a schematic diagram of a IOBase5 implementation



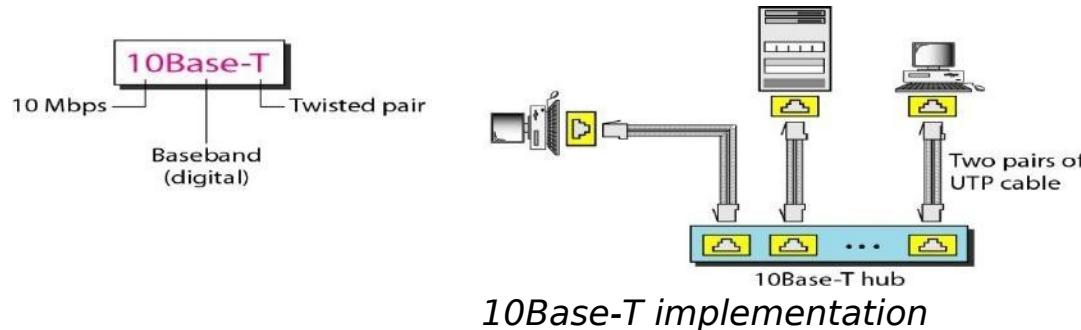
### 10Base2: Thin Ethernet

The second implementation is called 10 Base2, **thin** Ethernet, or Cheapernet. 10Base2 also uses a bus topology, but the cable is much thinner and more flexible. Figure shows the schematic diagram of a 10Base2 implementation.

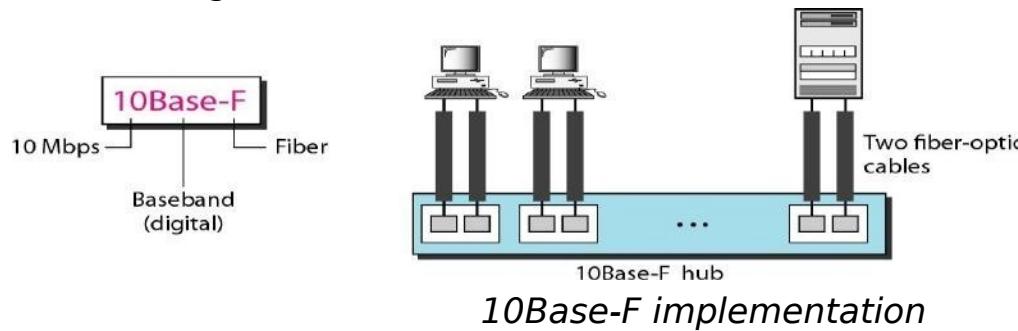


The third implementation is called 10Base-T or twisted-pair Ethernet. It uses a physical star topology. The stations are connected to a hub via two pairs of twisted cable, as shown in Figure

The maximum length of the twisted cable here is defined as 100 m, to minimize the effect of attenuation in the twisted cable



Although there are several types of optical fiber 10-Mbps Ethernet, the most common is called 10Base-F. 10Base-F uses a star topology to connect stations to a hub. The stations are connected to the hub using two fiber-optic cables, as shown in Figure

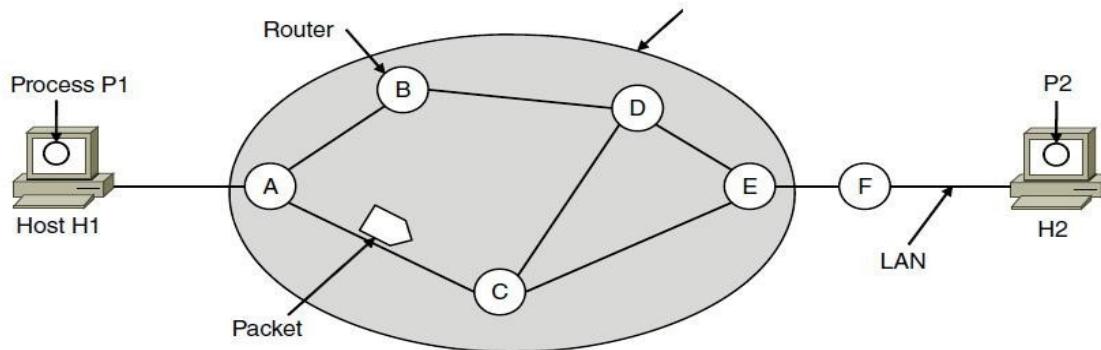


## UNIT-III

### 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

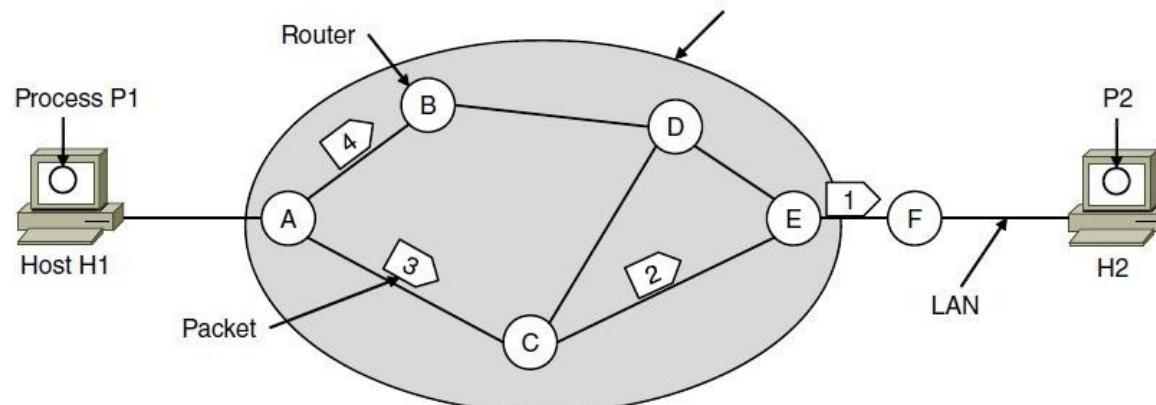


A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the 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 with the help of the transport layer protocols.

are frequently called **datagrams** (in analogy with telegrams) and the network is called a **datagram network**.



A's table (initially)

A	⊗
B	B
C	C
D	B
E	C
F	C

Dest. Line

A's table (later)

A	⊗
B	B
C	C
D	B
E	D
F	D

C's Table

A	A
B	A
C	⊗
D	E
E	E
F	E

E's Table

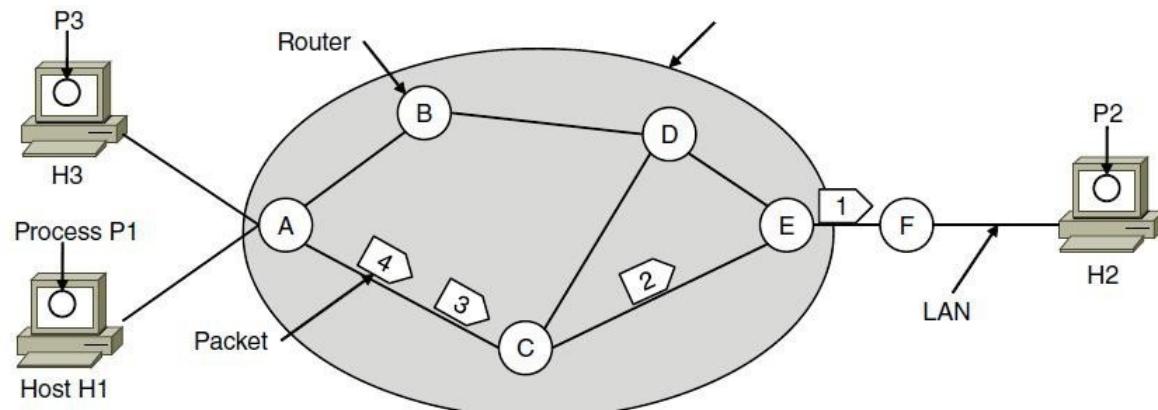
A	C
B	D
C	C
D	D
E	⊗
F	F

Let us assume for this example that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4, and send each of them in turn to router A.

Every router has an internal table telling it where to send packets for each of the possible destinations. Each table entry is a pair(destination and the outgoing line). Only directly connected lines can be used.

A's initial routing table is shown in the figure under the label "initially."

## 4 Implementation of connection-oriented service



A's table	
H1	1
H3	1
In	
C	1
C	2
Out	

C's Table	
A	1
A	2
E	1
E	2

E's Table	
C	1
C	2
F	1
F	2

If connection-oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a **VC (virtual circuit)**, and the network is called a **virtual-circuit network**

When a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works. When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.

This leads to the second row in the tables. Note that we have a conflict here because although A can easily distinguish connection 1 packets from  $H_1$  from connection 1 packets from  $H_3$ , C cannot do this. For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection. Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets.

In some contexts, this process is called **label switching**. An example of a connection-oriented network service is **MPLS (Multi Protocol Label Switching)**.

## 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 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.

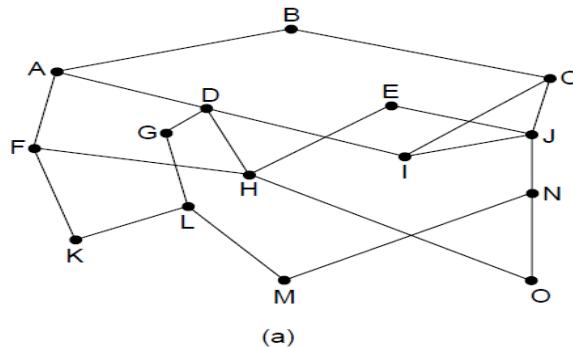
Adaptive algorithms differ in

- 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  $\Delta T$  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).

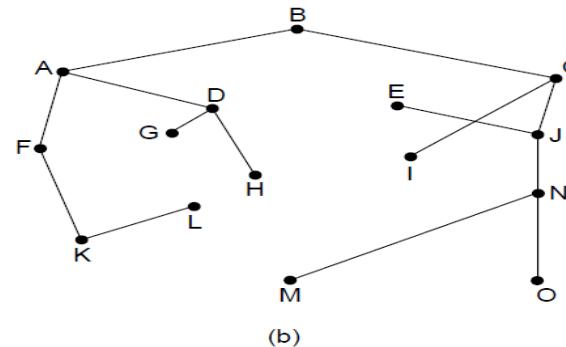
This procedure is called dynamic routing

### Different Routing Algorithms

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



(a) A network.



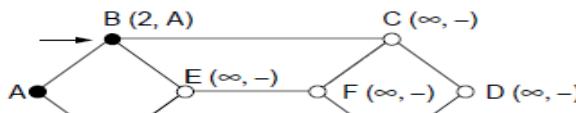
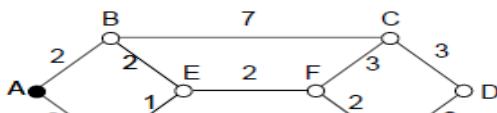
(b) A sink tree for router  $B$ .

### Shortest Path Routing (Dijkstra's)

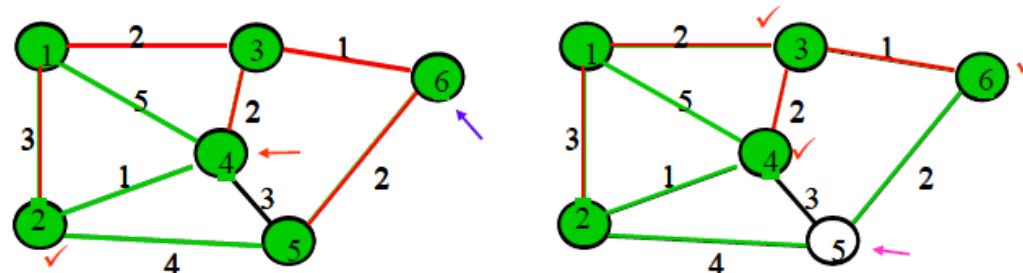
The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line or link.

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

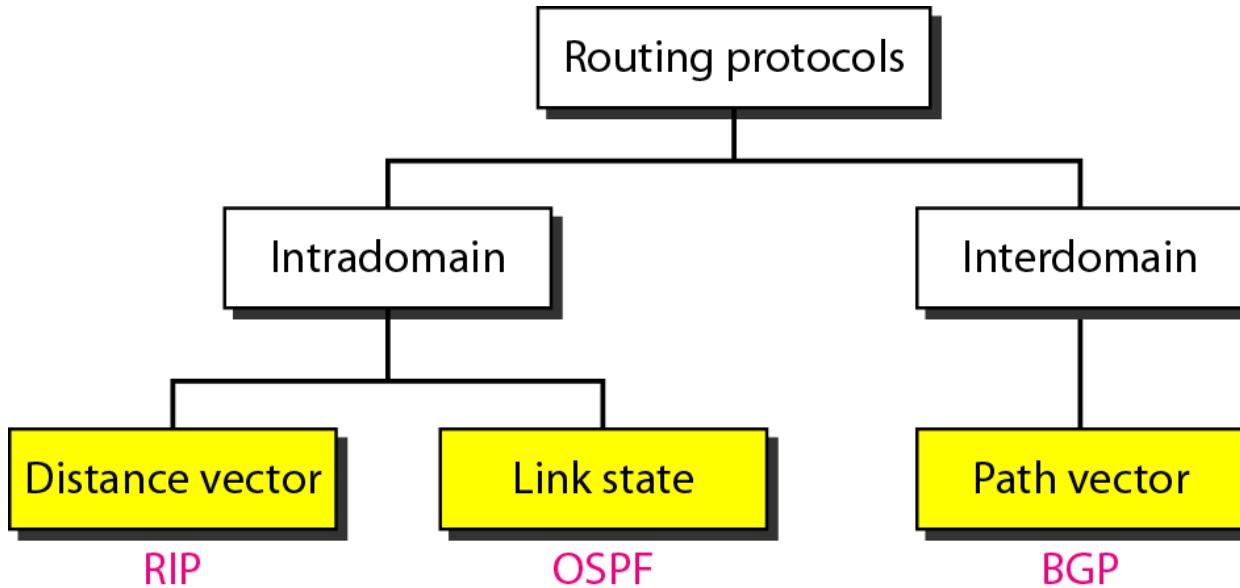


Iteration	Permanent	tentative	$D_2$	$D_3$	$D_4$	$D_5$	$D_6$
Initial	{1}	{2,3,4}	3	2 ✓	5	$\infty$	$\infty$
1	{1,3}	{2,4,6}	3 ✓	2	4	$\infty$	3
2	{1,2,3}	{4,6,5}	3	2	4	7	3 ✓
3	{1,2,3,6}	{4,5}	3	2	4 ✓	5	3
4	{1,2,3,4,6}	{5}	3	2	4	5 ✓	3
5	{1,2,3,4,5,6}	{}	3	2	4	5	3

## 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.

Routing between autonomous systems is referred to as inter domain routing. (PATH VECTOR)  
Each autonomous system can choose one or more intra domain routing protocols to handle routing inside the autonomous system. However, only one inter domain routing protocol handles routing between autonomous systems.



### Distance Vector Routing

In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node.

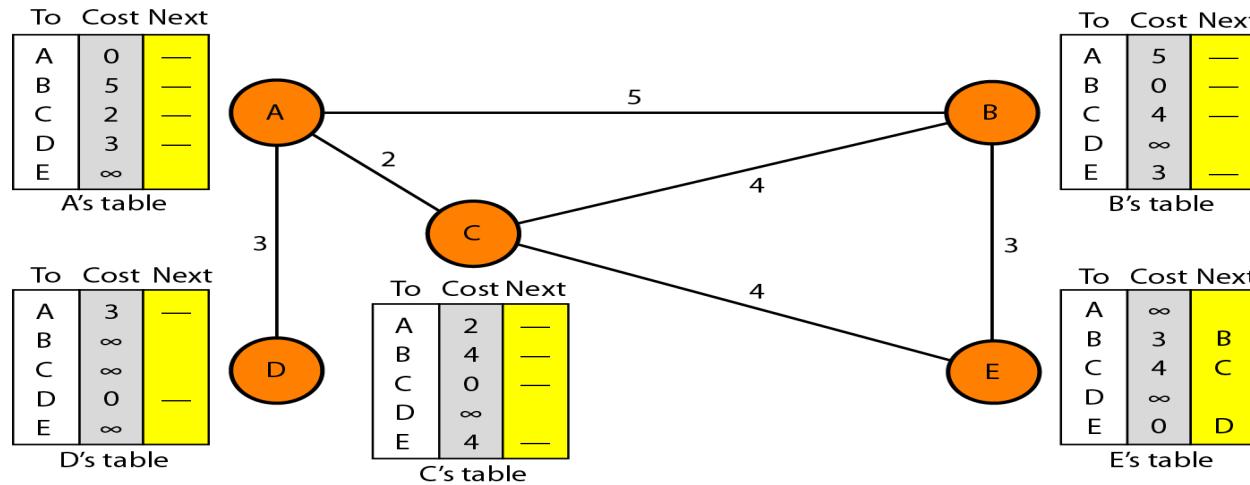
Mainly 3 things in this

*Initialization*

*Sharing*

*Updating*

## *Initialization of tables in distance vector routing*



## Sharing

The whole idea of distance vector routing is the sharing of information between neighbors. Although node A does not know about node E, node C does. So if node C shares its routing table with A, node A can also know how to reach node E. On the other hand, node C does not know how to reach node D, but node A does. If node A shares its routing table with node C, node C also knows how to reach node D. In other words, nodes A and C, as immediate neighbors, can improve their routing tables if they help each other.

NOTE: In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change

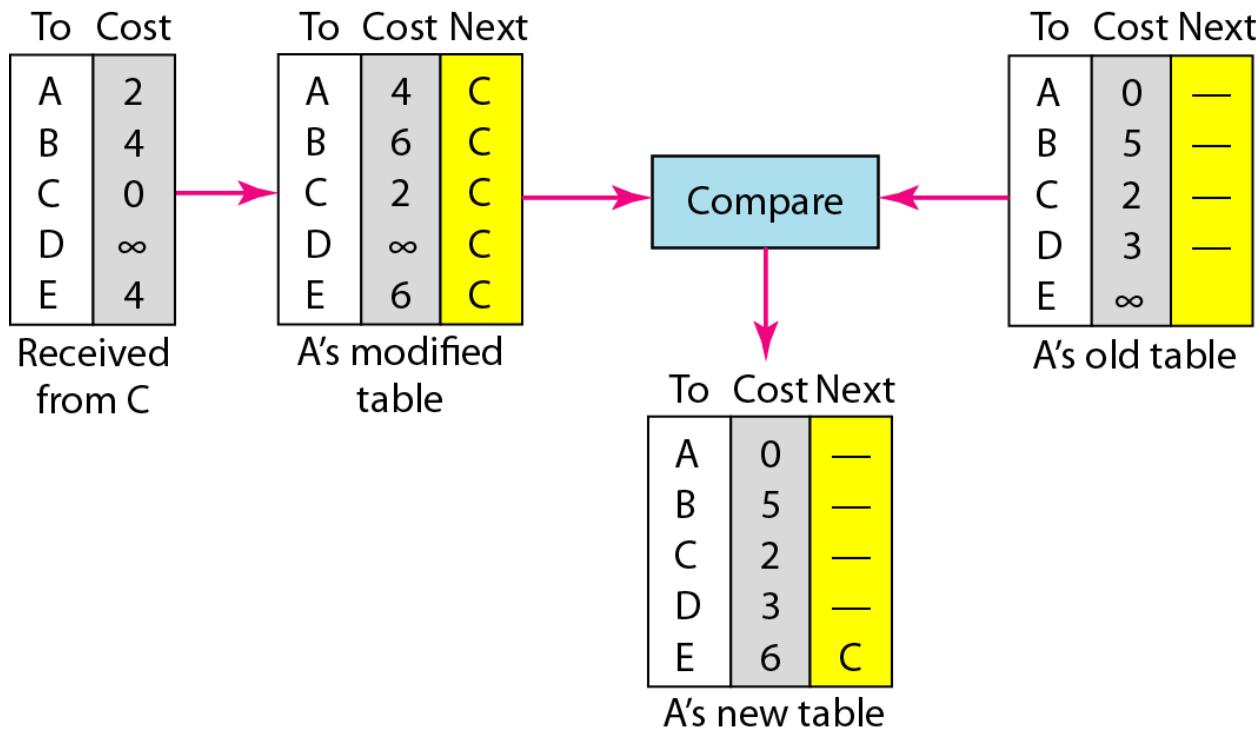
## Updating

When a node receives a two-column table from a neighbor, it needs to update its routing table. Updating takes three steps:

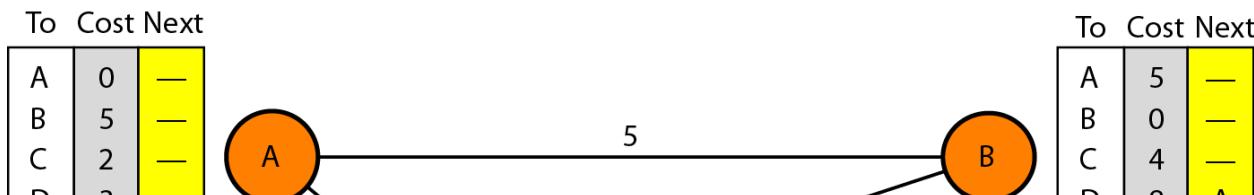
1. The receiving node needs to add the cost between itself and the sending node to each value

For example, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist anymore. The new route has a distance of infinity.

### ***Updating in distance vector routing***



### Final Diagram



## When to Share

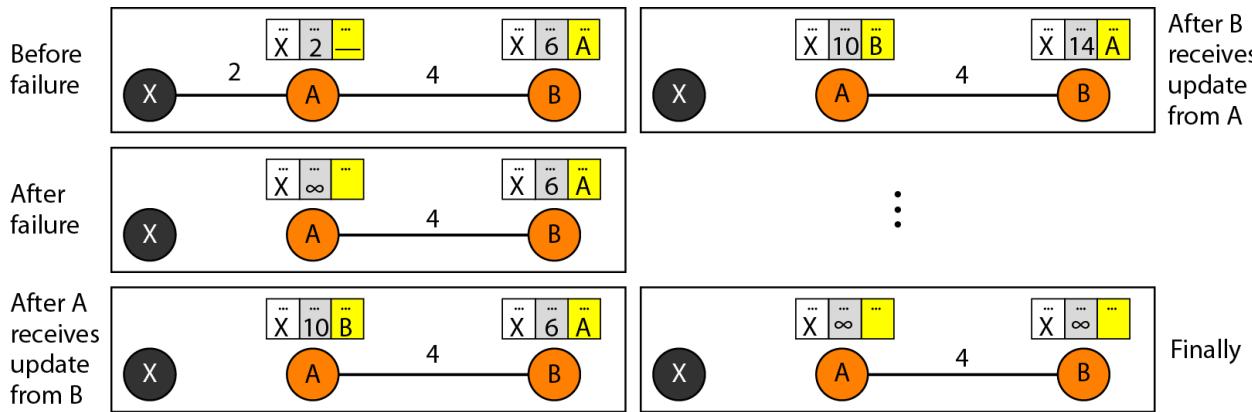
The question now is, When does a node send its partial routing table (only two columns) to all its immediate neighbors? The table is sent both periodically and when there is a change in the table.

Periodic Update A node sends its routing table, normally every 30 s, in a periodic update. The period depends on the protocol that is using distance vector routing.

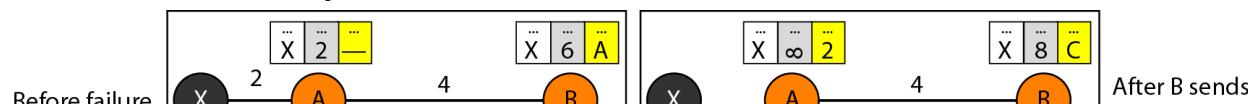
Triggered Update A node sends its two-column routing table to its neighbors anytime there is a change in its routing table. This is called a triggered update. The change can result from the following.

1. A node receives a table from a neighbor, resulting in changes in its own table after updating.
2. A node detects some failure in the neighboring links which results in a distance change to infinity.

## **Two-node instability**



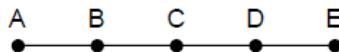
## **Three-node instability**



be 1 and define 16 as infinity. However, this means that the distance vector routing cannot be used in large systems. The size of the network, in each direction, cannot exceed 15 hops.

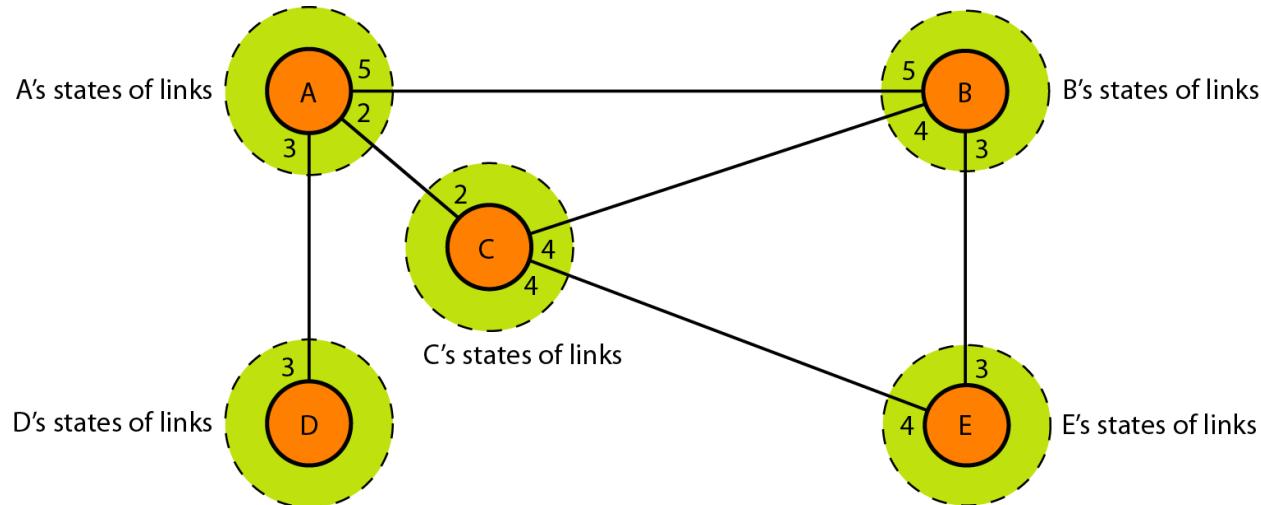
2. **Split Horizon:** In this strategy, instead of flooding the table through each interface, each node sends **only part of its table** through each interface. If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows). Taking information from node A, modifying it, and sending it back to node A creates the confusion. In our scenario, node B eliminates the last line of its routing table before it sends it to A. In this case, node A keeps the value of infinity as the distance to X. Later when node A sends its routing table to B, node B also corrects its routing table. The system becomes stable after the first update: both node A and B know that X is not reachable.
3. **Split Horizon and Poison Reverse** Using the split horizon strategy has one drawback. Normally, the distance vector protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess that this is due to the split horizon strategy (the source of information was A) or because B has not received any news about X recently. The split horizon strategy can be combined with the poison reverse strategy. Node B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."

### The Count-to-Infinity Problem



## Link State Routing

Link state routing is based on the assumption that, although the global knowledge about the topology is not clear, each node has partial knowledge: it knows the state (type, condition, and cost) of its links. **In other words, the whole topology can be compiled from the partial knowledge of each node**



### Building Routing Tables

1. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called **flooding, in an efficient and reliable way**.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree

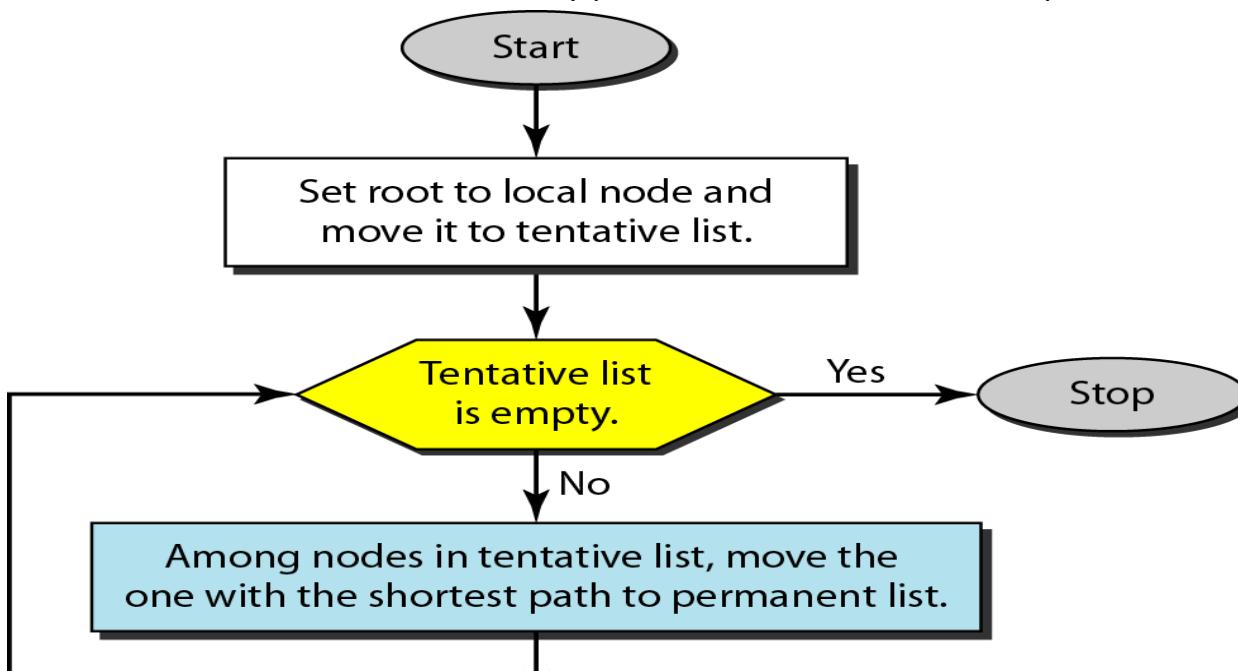
- I. **Creation of Link State Packet (LSP)** A link state packet can carry a large amount of information. For the moment, we assume that it carries a minimum amount of data: the node identity, the list of links, a sequence number, and age. The first two, node identity and the list of links, are needed to make the topology. The third, sequence number, facilitates flooding and distinguishes new LSPs from old ones. The fourth, age, prevents old LSPs from

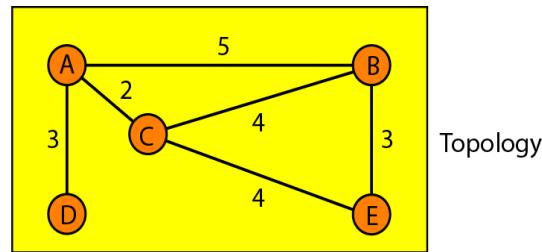
2. A node that receives an LSP compares it with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer, the node does the following:
- It discards the old LSP and keeps the new one.
  - It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain (where a node has only one interface).

### III. Formation of Shortest Path Tree: Dijkstra Algorithm

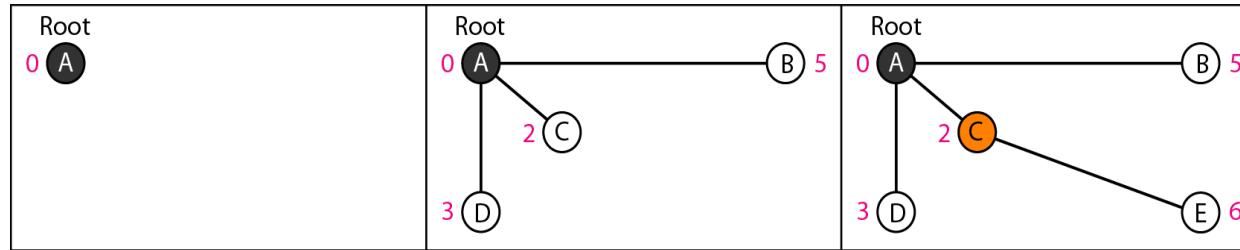
A shortest path tree is a tree in which the path between the root and every other node is the shortest.

The Dijkstra algorithm creates a shortest path tree from a graph. The algorithm divides the nodes into two sets: **tentative and permanent**. It finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent.

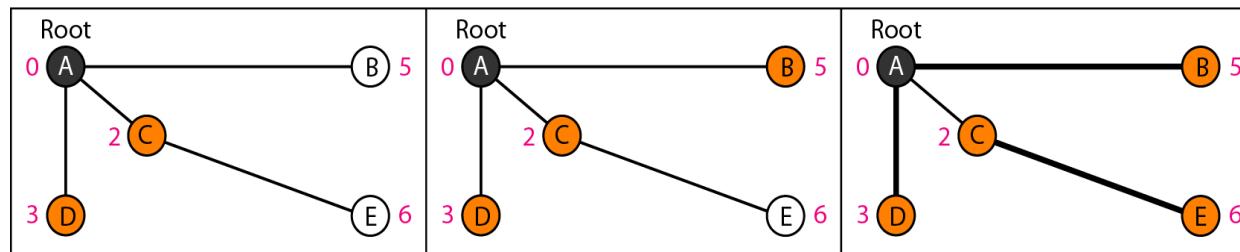




Topology



1. Set root to A and move A to tentative list.  
2. Move A to permanent list and add B, C, and D to tentative list.  
3. Move C to permanent and add E to tentative list.



4. Move D to permanent list.  
5. Move B to permanent list.  
6. Move E to permanent list  
(tentative list is empty).

#### IV. Calculation of a routing table

routing table for node A

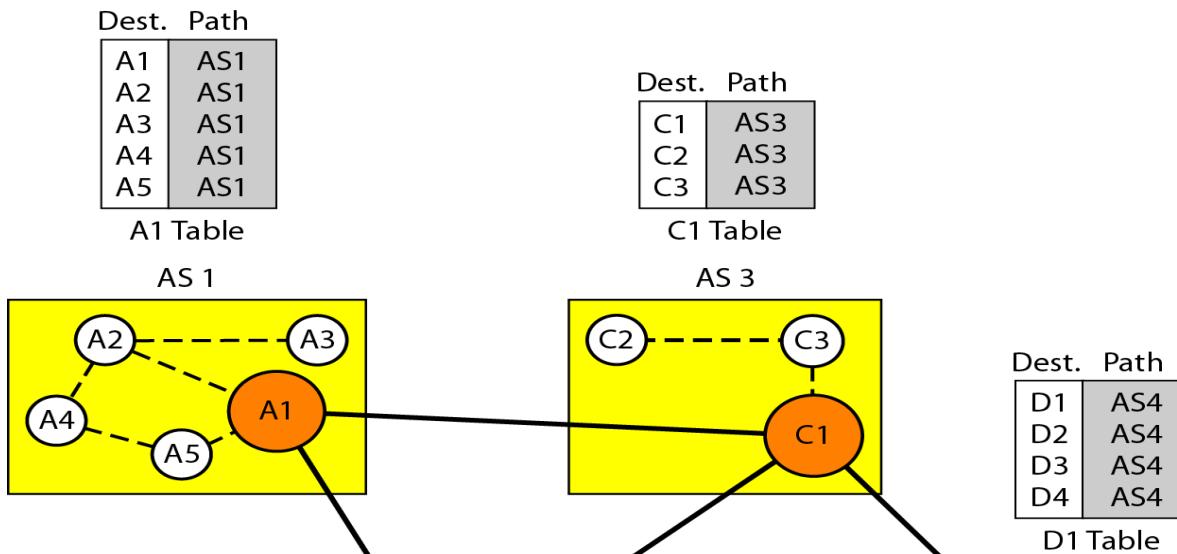
<i>Node</i>	<i>Cost</i>	<i>Next Router</i>
A	0	—
B	5	—
C	2	—

**huge amount of resources** to calculate routing tables. It also creates heavy traffic because of flooding. There is a need for a third routing protocol which we call path vector routing.

Path vector routing proved to be useful for inter domain routing. The principle of path vector routing is similar to that of distance vector routing. **In path vector routing, we assume that there is one node** (there can be more, but one is enough for our conceptual discussion) **in each AS** that acts on behalf of the entire AS. Let us call it the **speaker node**. The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighboring ASs. The idea is the same as for distance vector routing except that only speaker nodes in each AS can communicate with each other. However, what is advertised is different. A speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems

## Initialization

*Initial routing tables in path vector routing*



and C1. Node C1 shares its table with nodes D1, B1, and A1. Node B1 shares its table with C1 and A1. Node D1 shares its table with C1.

Dest.	Path	Dest.	Path	Dest.	Path	Dest.	Path
A1	AS1	A1	AS2-AS1	A1	AS3-AS1	A1	AS4-AS3-AS1
...		...		...		...	
A5	AS1	A5	AS2-AS1	A5	AS3-AS1	A5	AS4-AS3-AS1
B1	AS1-AS2	B1	AS2	B1	AS3-AS2	B1	AS4-AS3-AS2
...	...	...		...		...	
B4	AS1-AS2	B4	AS2	B4	AS3-AS2	B4	AS4-AS3-AS2
C1	AS1-AS3	C1	AS2-AS3	C1	AS3	C1	AS4-AS3
...	...	...		...		...	
C3	AS1-AS3	C3	AS2-AS3	C3	AS3	C3	AS4-AS3
D1	AS1-AS2-AS4	D1	AS2-AS3-AS4	D1	AS3-AS4	D1	AS4
...	...	...		...		...	
D4	AS1-AS2-AS4	D4	AS2-AS3-AS4	D4	AS3-AS4	D4	AS4

A1 Table                  B1 Table                  C1 Table                  D1 Table

**Updating** When a speaker node receives a two-column table from a neighbor, it updates its own table by adding the nodes that are not in its routing table and adding its own autonomous system and the autonomous system that sent the table. After a while each speaker has a table and knows how to reach each node in other Ass

- a) **Loop prevention.** The instability of distance vector routing and the creation of loops can be avoided in path vector routing. When a router receives a message, it checks to see if its AS is in the path list to the destination. If it is, looping is involved and the message is ignored.
- b) **Policy routing.** Policy routing can be easily implemented through path vector routing. When a router receives a message, it can check the path. If one of the AS listed in the path

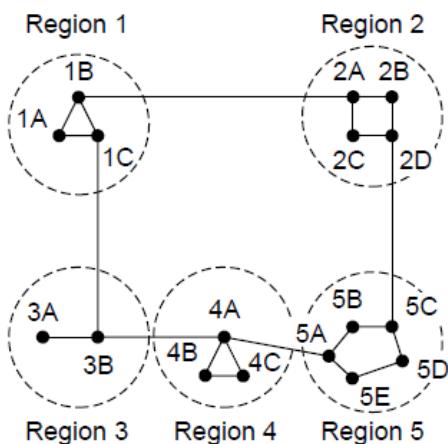
## Hierarchical Routing

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.

At a certain point, the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically, as it is in the telephone network.

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.

For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the regions into clusters, the clusters into zones, the zones into groups, and so on, until we run out of names for aggregations



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	1C	5
5D	1C	5
5E	1C	5

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

If a three-level hierarchy is chosen, with 8 clusters each containing 9 regions of 10 routers, each router needs 10 entries for local routers, 8 entries for routing to other regions within its own cluster, and 7 entries for distant clusters, for a total of 25 entries

Kamoun and Kleinrock (1979) discovered that the optimal number of levels for an  $N$  router network is  $\ln N$ , requiring a total of  $e \ln N$  entries per router

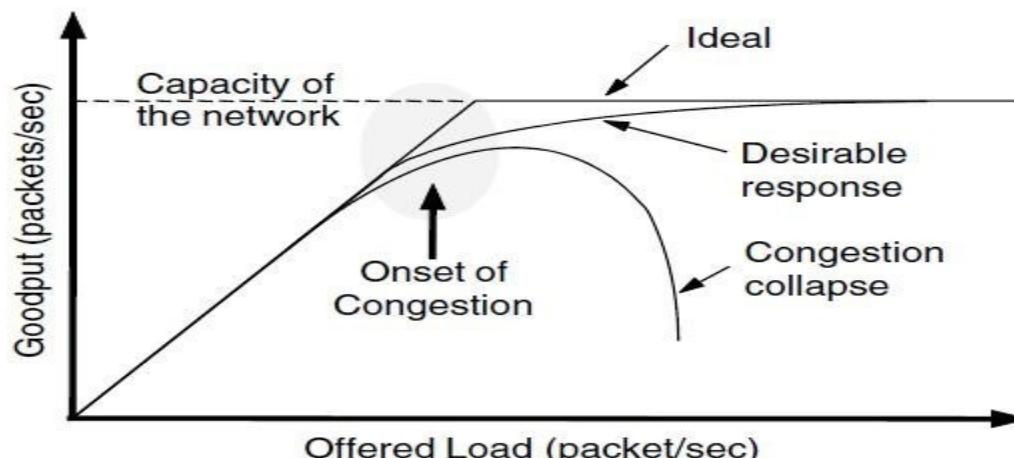
---

## 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.

However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together. In this chapter we will look at the network aspects of congestion.



When too much traffic is offered, congestion sets in and performance degrades sharply

## **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. Although there is no congestion (the network itself is not in trouble), flow control is needed to force the supercomputer to stop frequently to give the personal computer a chance to breathe.

At the other extreme, consider a network with 1-Mbps lines and 1000 large computers, half of which are trying to transfer files at 100 kbps to the other half. Here, the problem is not that of fast senders overpowering slow receivers, but that the total offered traffic exceeds what the network can handle.

The reason congestion control and flow control are often confused is that the best way to handle both problems is to get the host to slow down. Thus, a host can get a “slow down” message either because the receiver cannot handle the load or because the network cannot handle it.

Several techniques can be employed. These include:

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

## **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.
4. An example of a choke packet is the 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.

## Traffic Shaping

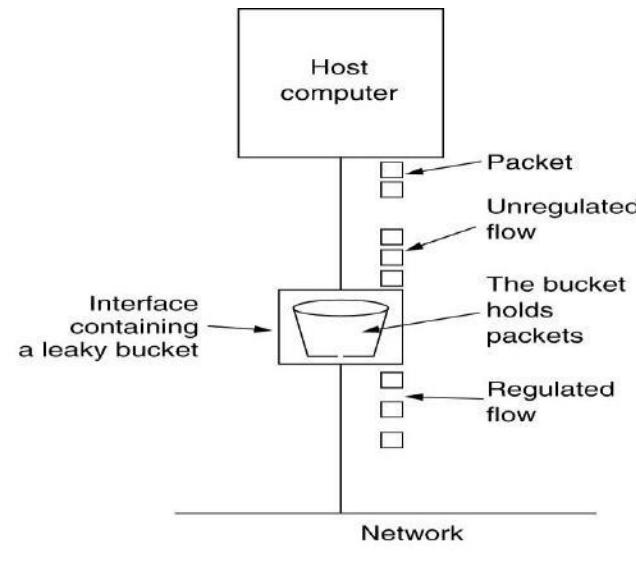
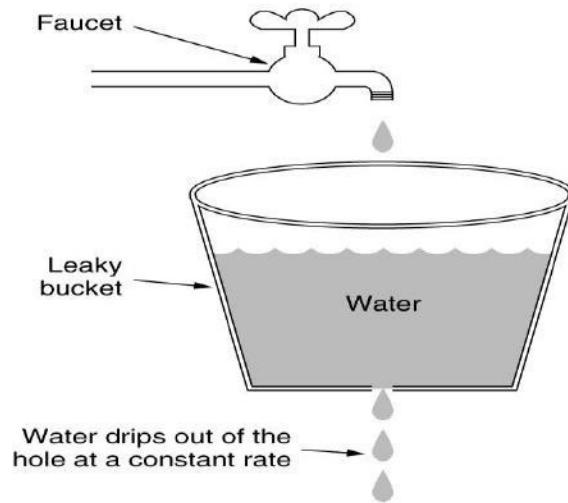
1. Another method of congestion control is to “shape” the traffic before it enters the network.
2. Traffic shaping controls the *rate* at which packets are sent (not just how many). Used in ATM and Integrated Services networks.
3. At connection set-up time, the sender and carrier negotiate a traffic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

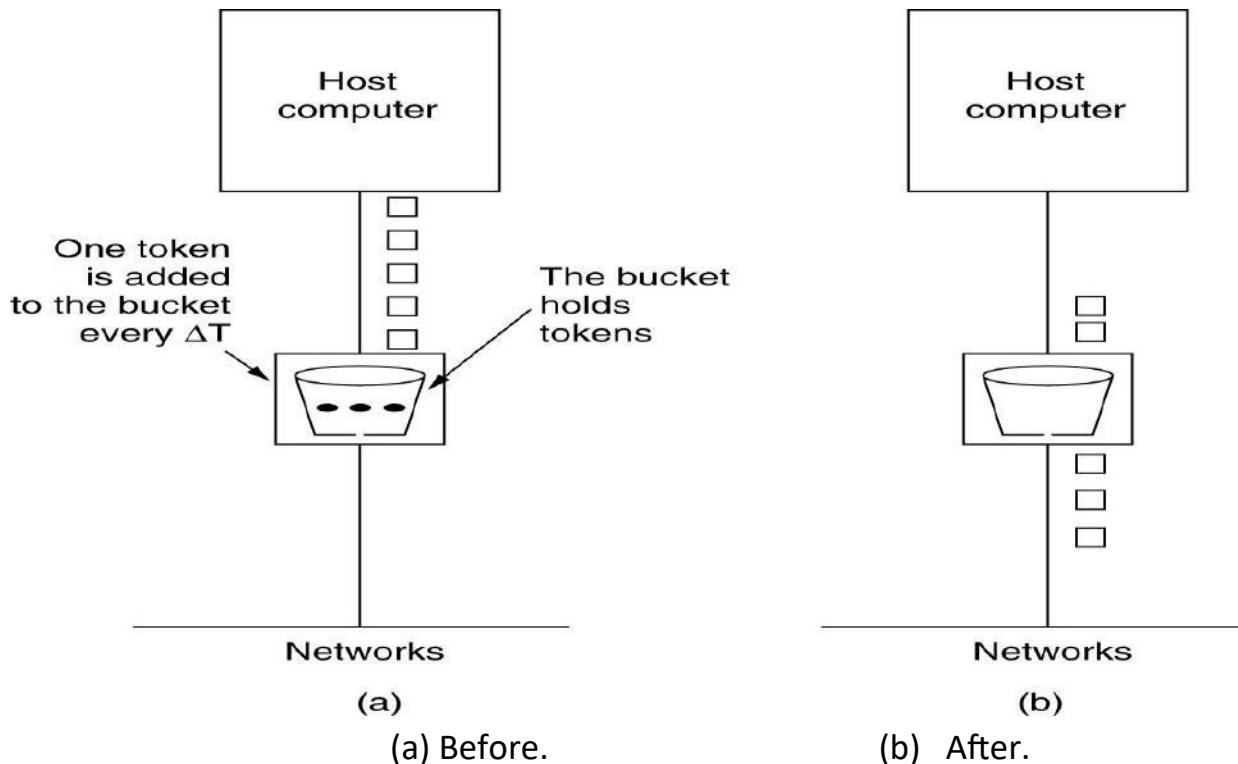


(a)

(b)

## 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  $\Delta t$  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.



## TRANSPORT LAYER

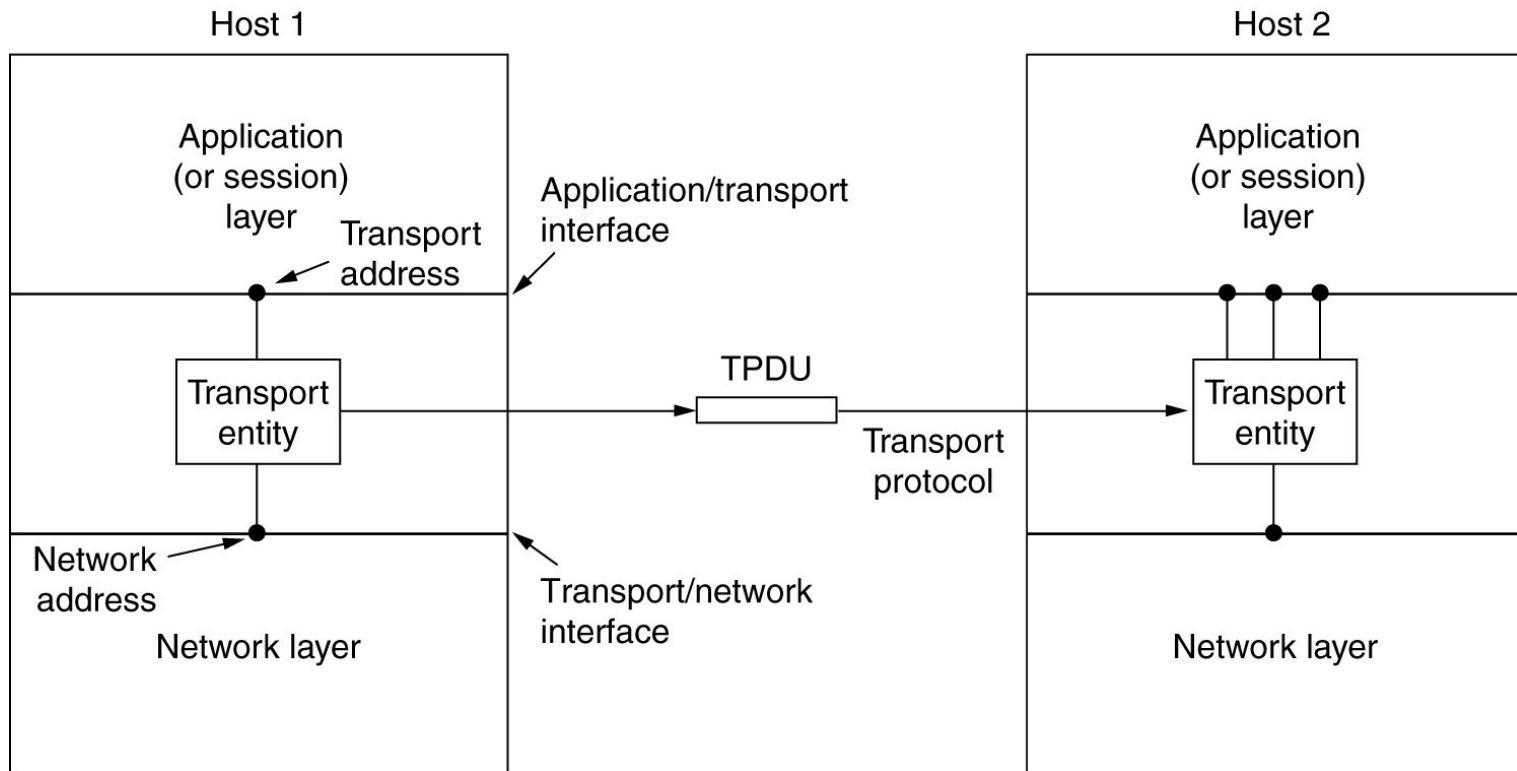
The network layer provides end-to-end packet delivery using datagrams or virtual circuits.

The transport layer builds on the network layer to provide **data transport from a process on a source machine to a process on a destination machine** with a desired level of reliability that is independent of the physical networks currently in use.

## 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**.

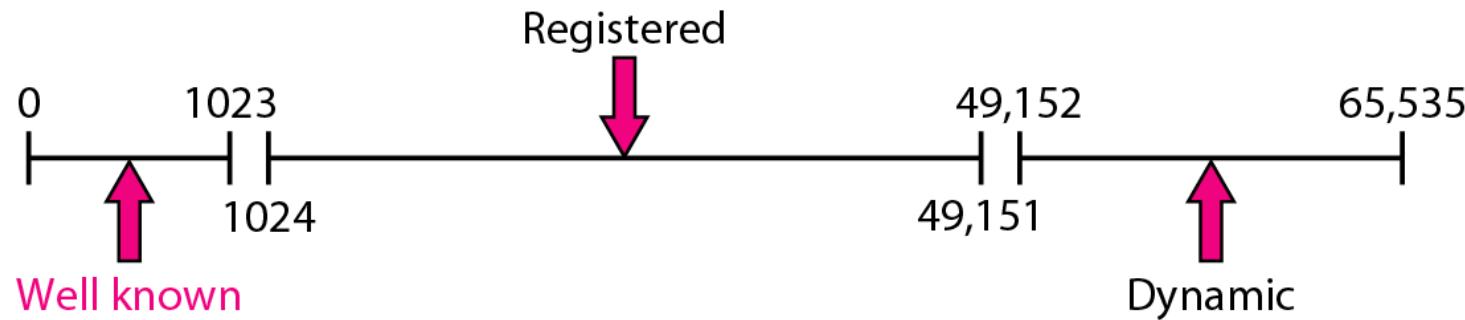


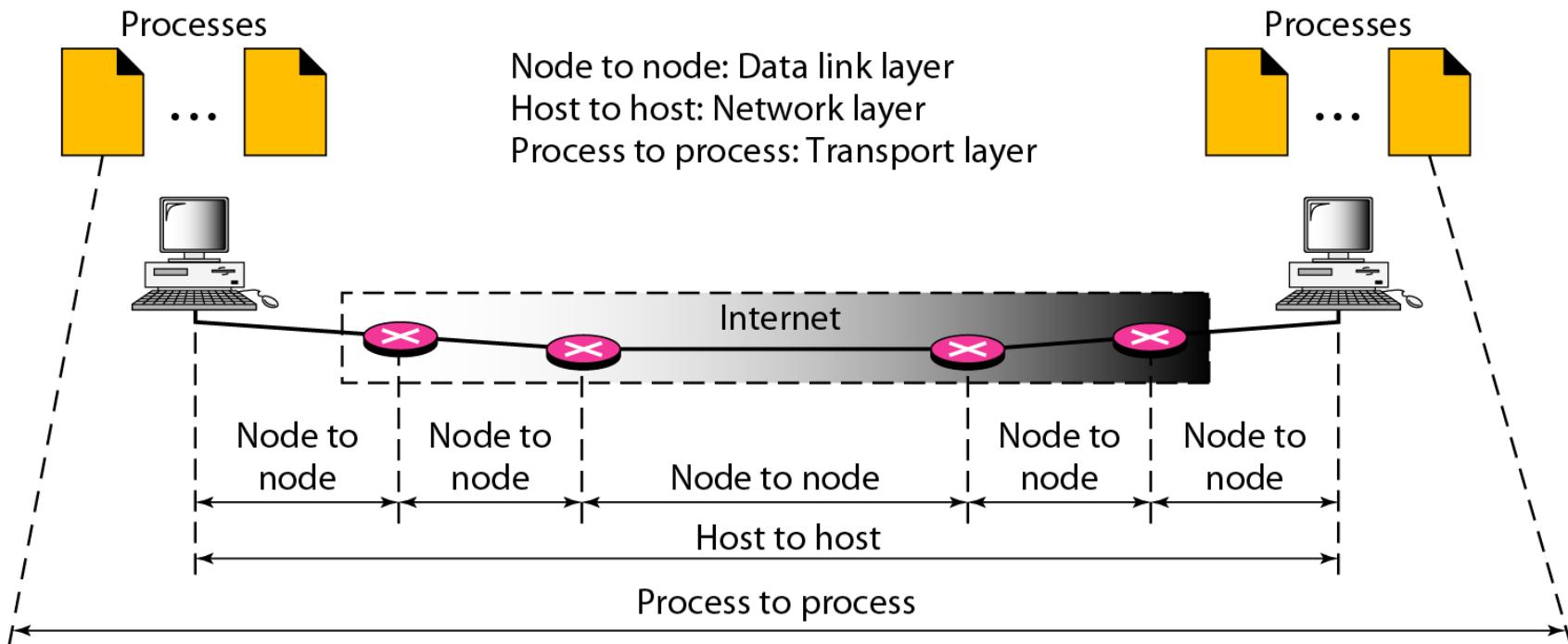
The network, transport, and application layers.

The connection-oriented transport service. connections have three phases: establishment, data transfer, and release.

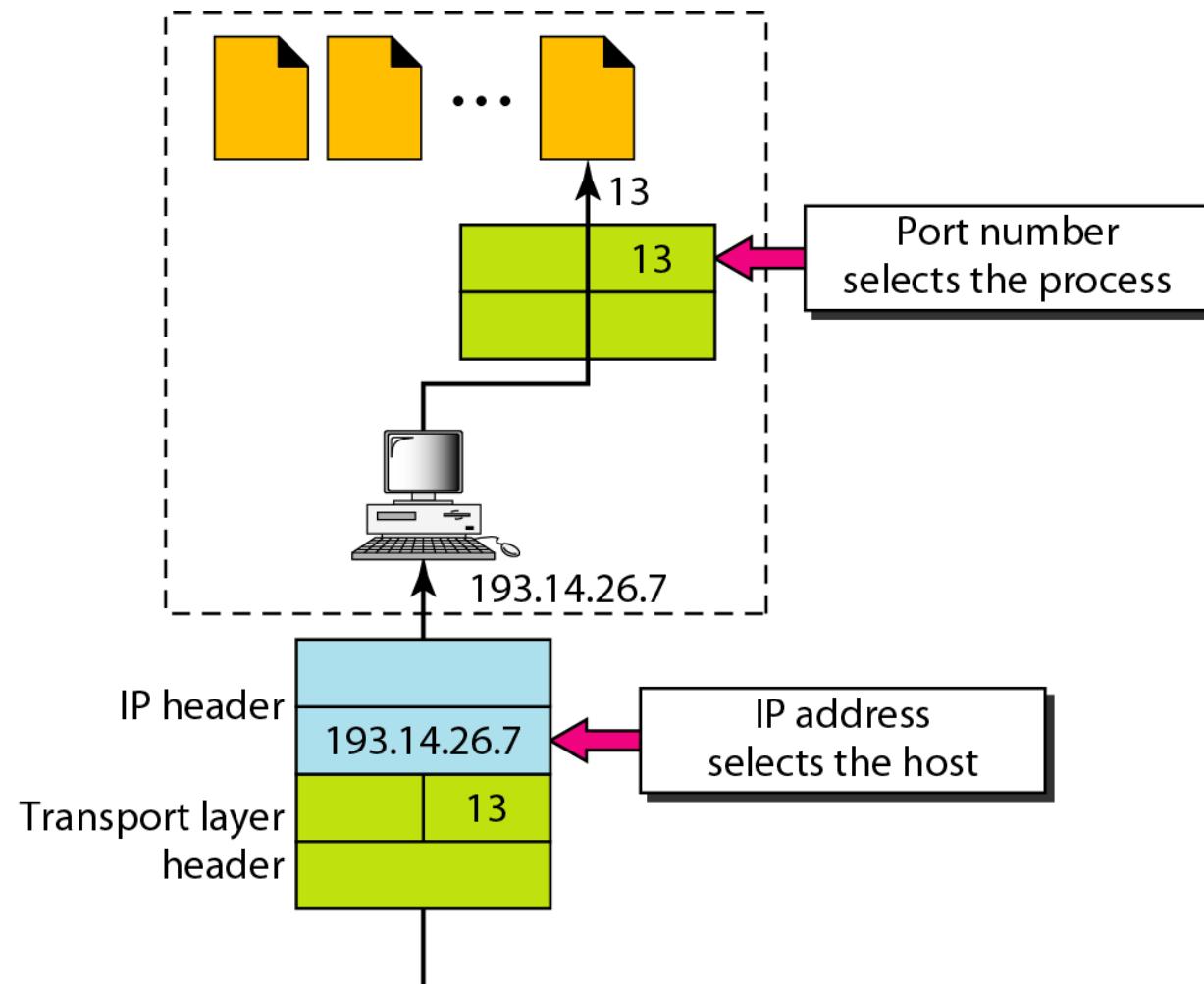
Addressing and flow control

The connectionless transport service .





## *IP addresses versus port numbers*



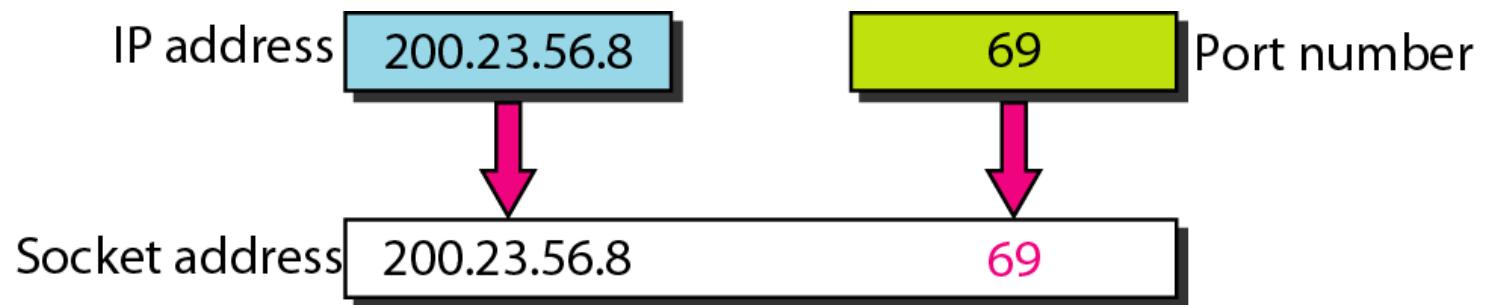
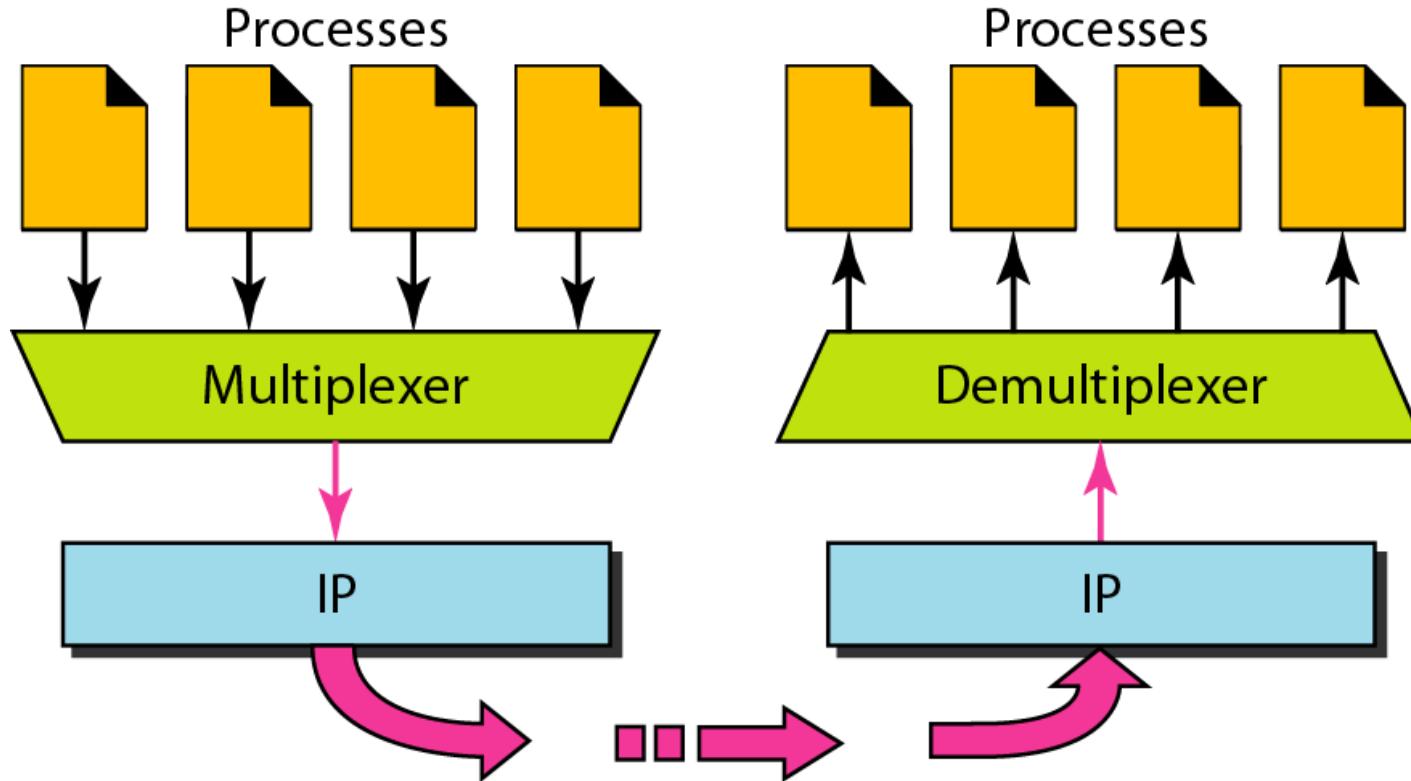
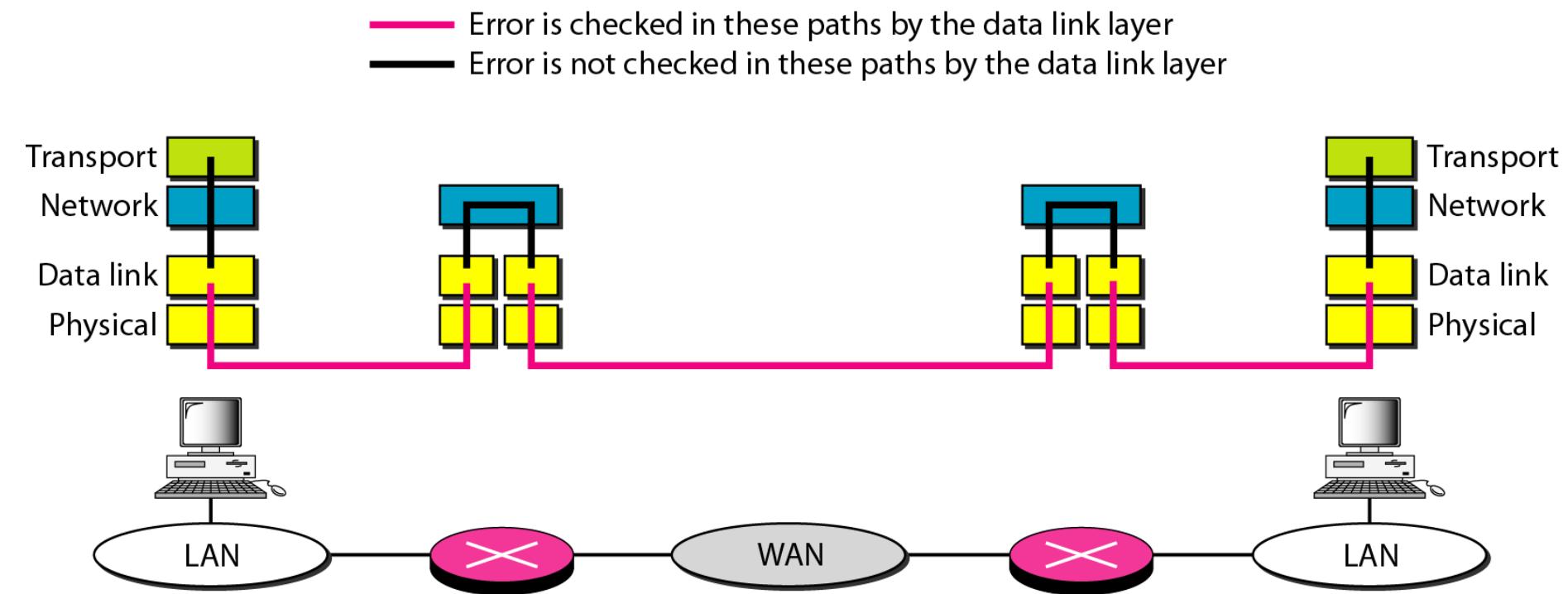


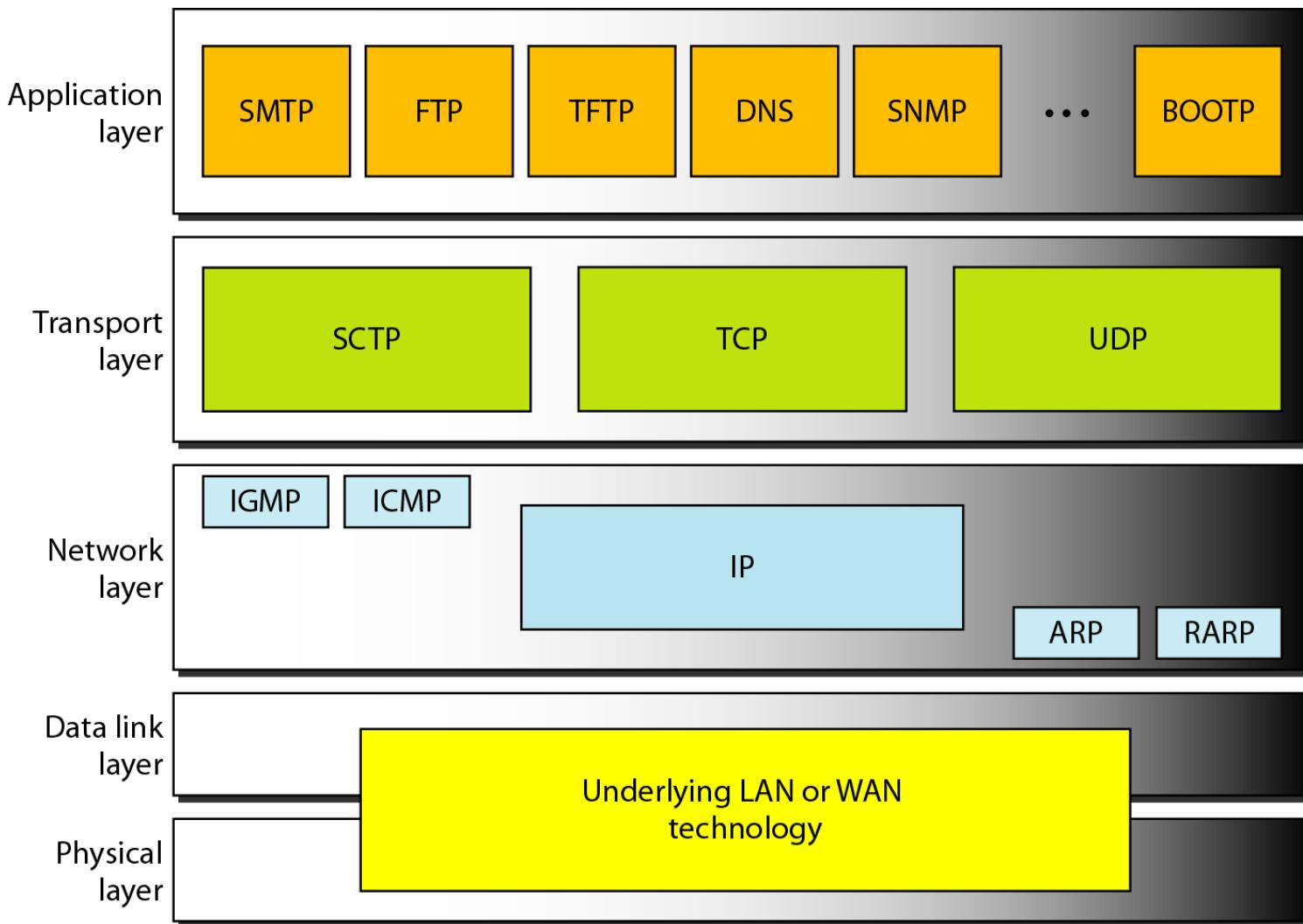
Figure 23.6 Multiplexing and demultiplexing



**Figure 23.7** Error control



**Figure 23.8 Position of UDP, TCP, and SCTP in TCP/IP suite**



# Transport Service Primitives

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.	This side wants to release the connection

The primitives for a simple transport service.

To start with, the server executes a LISTEN primitive, typically by calling a library procedure that makes a system call that blocks the server until a client turns up.

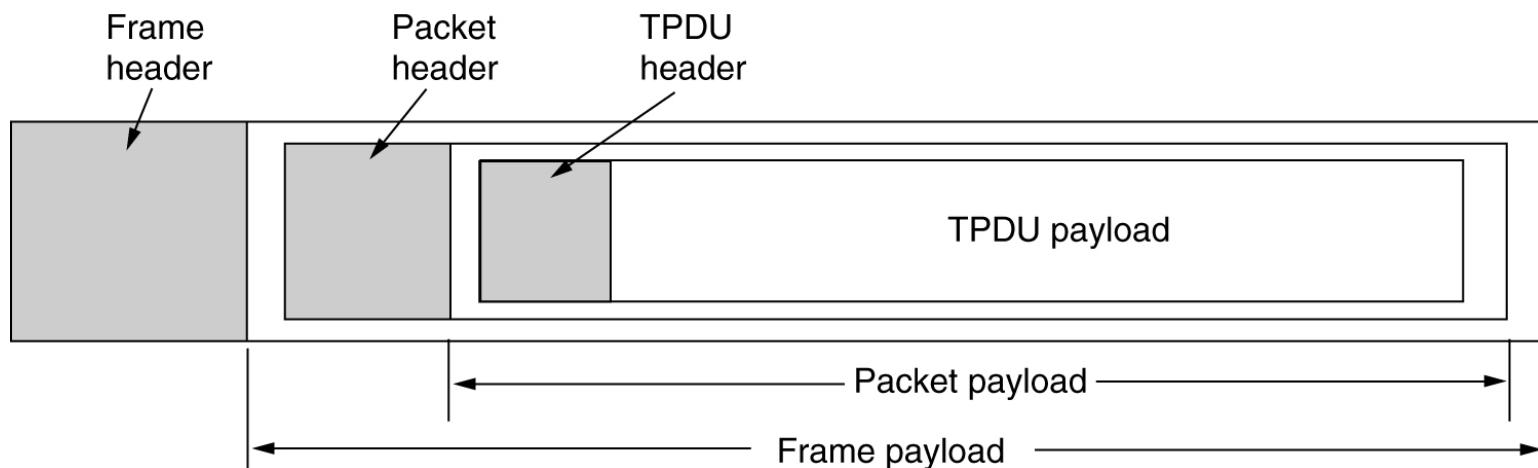
When a client wants to talk to the server, it executes a CONNECT primitive. The transport entity carries out this primitive by blocking the caller and sending a packet to the server. The client's CONNECT call causes a CONNECTION REQUEST segment to be sent to the server. When it arrives, the transport entity checks to see that the server is blocked on a LISTEN (i.e., is interested in handling requests). If so, it then unblocks the server and sends a CONNECTION ACCEPTED segment back to the client. When this segment arrives, the client is unblocked and the connection is established.

Data can now be exchanged using the SEND and RECEIVE primitives. In the simplest form, either party can do a (blocking) RECEIVE to wait for the other party to do a SEND. When the segment arrives, the receiver is unblocked. It can then process the segment and send a reply. As long as both sides can keep track of whose turn it is to send, this scheme works fine.

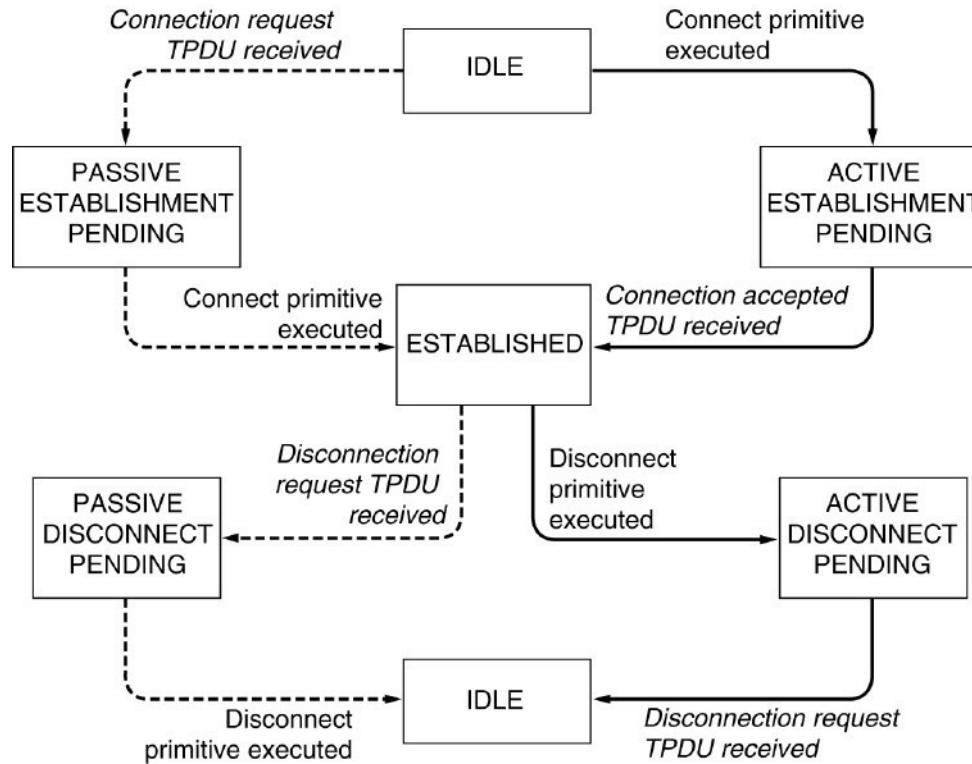
When a connection is no longer needed, it must be released to free up table space within the two transport entities. Disconnection has two variants: asymmetric and symmetric.

In the asymmetric variant, either transport user can issue a DISCONNECT primitive, which results in a DISCONNECT segment being sent to the remote transport entity. Upon its arrival, the connection is released.

In the symmetric variant, each direction is closed separately, independently of the other one. When one side does a DISCONNECT, that means it has no more data to send but it is still willing to accept data from its partner. In this model, a connection is released when both sides have done a DISCONNECT



# Transport Service Primitives

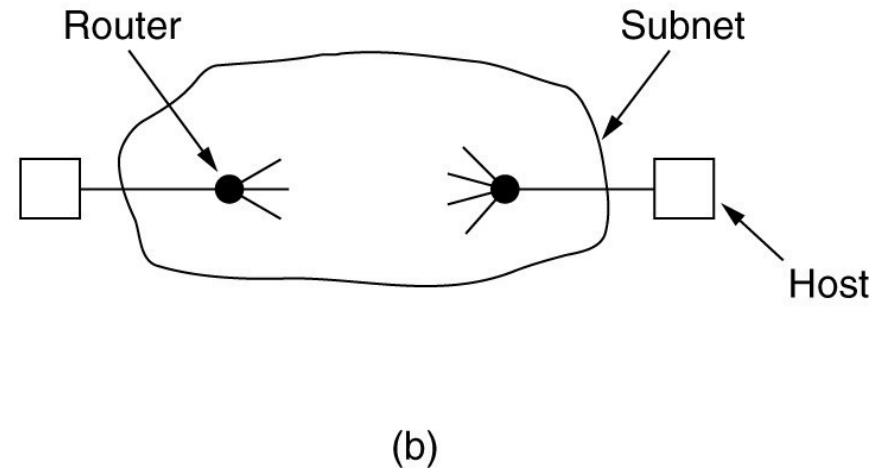
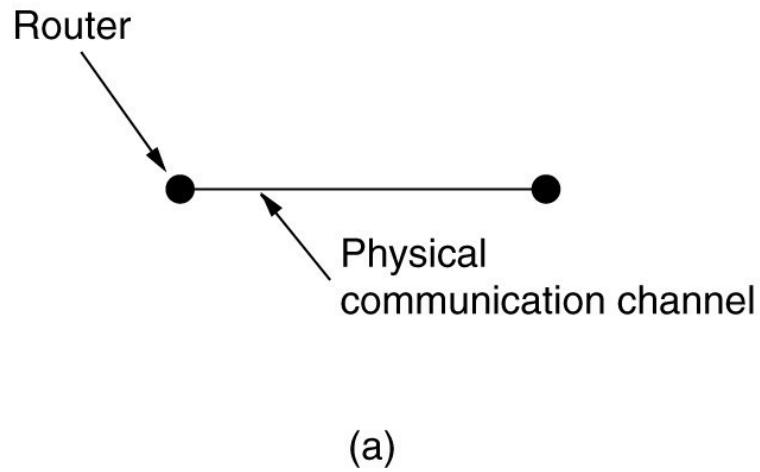


A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

# Elements of Transport Protocols

- Addressing
- Connection Establishment
- Connection Release
- Flow Control and Buffering
- Multiplexing
- Crash Recovery

# Transport Protocol



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

1 Over point-to-point links such as wires or optical fiber, it is usually not necessary for a router to specify which router it wants to talk to—each outgoing line leads directly to a particular router. In the transport layer, explicit addressing of destinations is required.

2 The process of establishing a connection over the wire of Fig(a) is simple: the other end is always there (unless it has crashed, in which case it is not there). Either way, there is not much to do. Even on wireless links the process is not much different. Just sending a message is sufficient to have it reach all other destinations. If the message is not acknowledged due to an error, it can be resent. In the transport layer, initial connection establishment is complicated, as we will see.

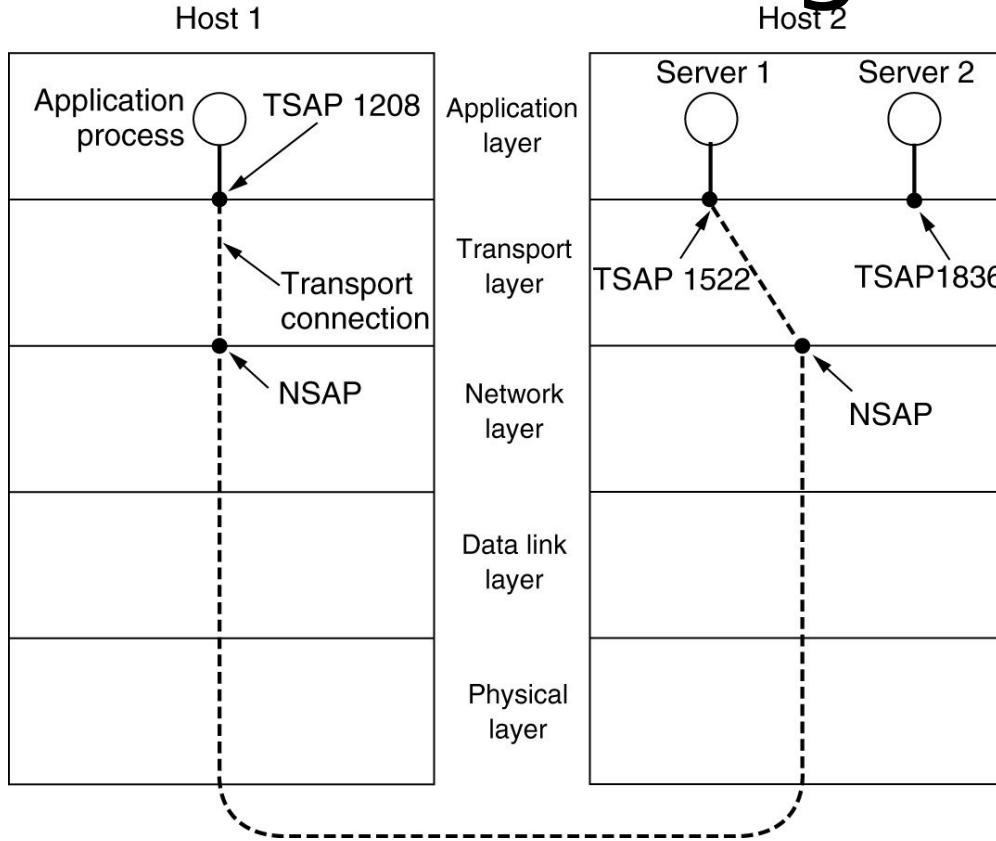
3 Another (exceedingly annoying) difference between the data link layer and the transport layer is the potential existence of storage capacity in the network. The consequences of the network's ability to delay and duplicate packets can sometimes be disastrous and can require the use of special protocols to correctly transport information.

4. 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

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. (Connectionless transport has the same problem: to whom should each message be sent?) The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called **ports**. We will use the generic term **TSAP (Transport Service Access Point)** 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 (Network Service Access Points)**. IP addresses are examples of NSAPs.

# Addressing



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. 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.

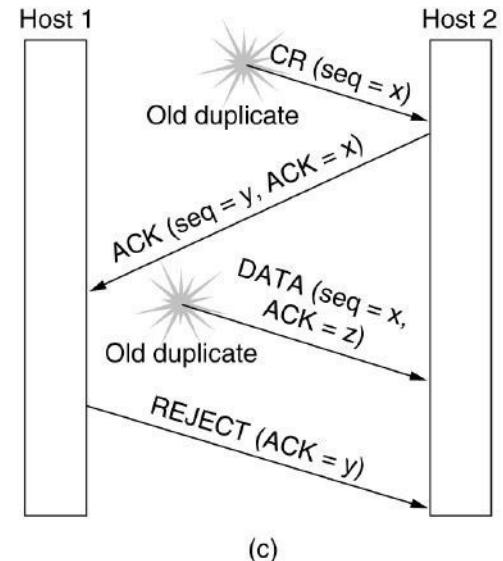
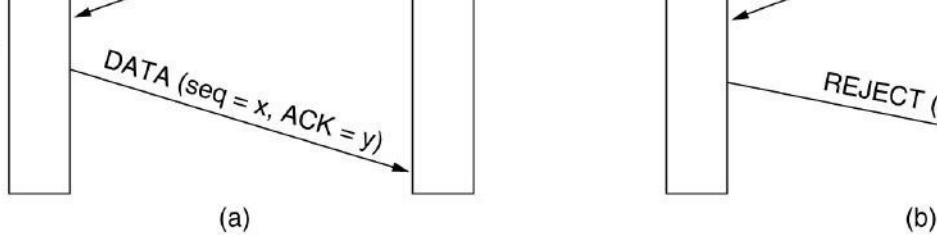
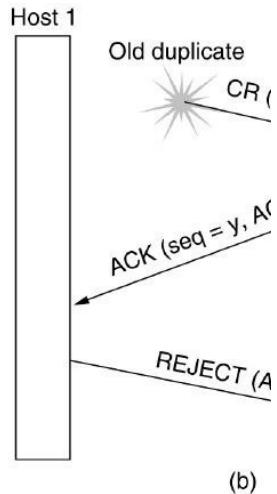
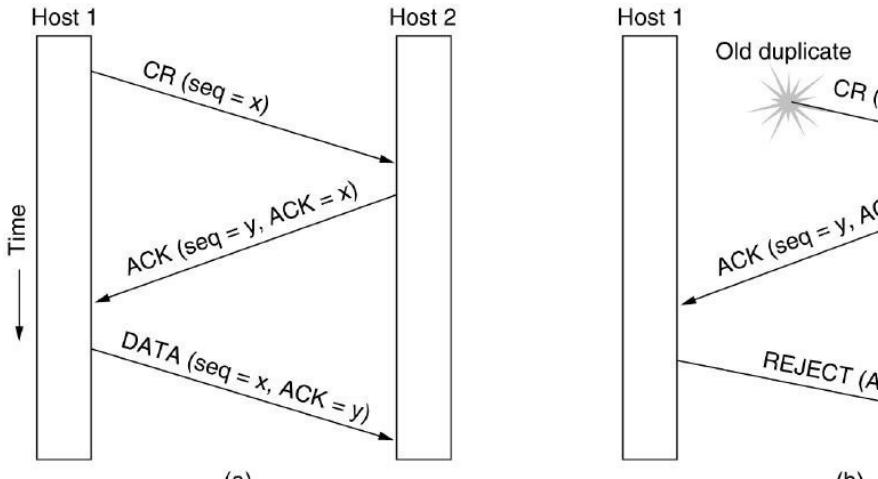
special process called a **portmapper**

## CONNECTION ESTABLISHMENT

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, 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

To solve this specific problem,(DELAYED DUPLICATES) Tomlinson (1975) introduced the **three-way handshake**. This establishment protocol involves one peer checking with the other that the connection request is indeed current. The normal setup procedure when host 1 initiates is shown in Fig. (a). 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.

# Connection E



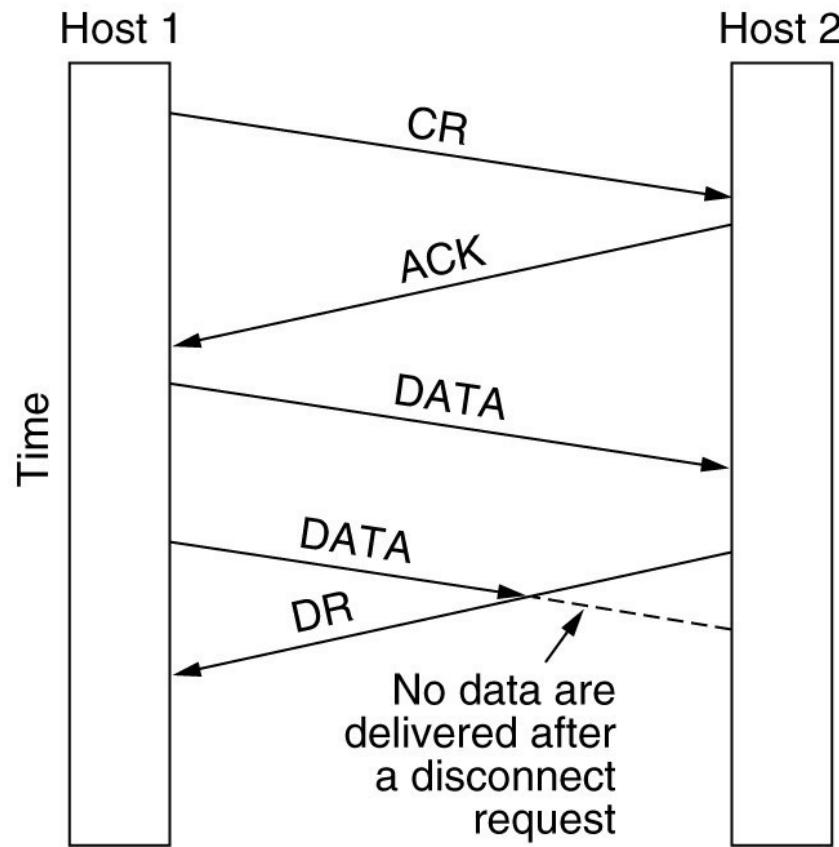
;) a connection using a CONNECTION REQUEST.

- (a) Normal operation,
  - (b) Old CONNECTION REQUEST appearing out of nowhere.
  - (c) Duplicate CONNECTION REQUEST and duplicate ACK.
- (c)

In Fig.(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. (c). As in the previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it. At this point, it is crucial to realize that host 2 has proposed using  $y$  as *the initial sequence number* for host 2 to host 1 traffic, knowing full well that no segments containing sequence number  $y$  or acknowledgements to  $y$  are *still in existence*. When the second delayed segment arrives at host 2, the fact that  $z$  *has been acknowledged rather than  $y$*  tells host 2 that this, too, is an old duplicate. *The important thing to realize here is that there is no combination of old segments that can cause the protocol to fail and have a connection set up by accident when no one wants it.*

# Connection Release



Abrupt disconnection with loss of data.

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.

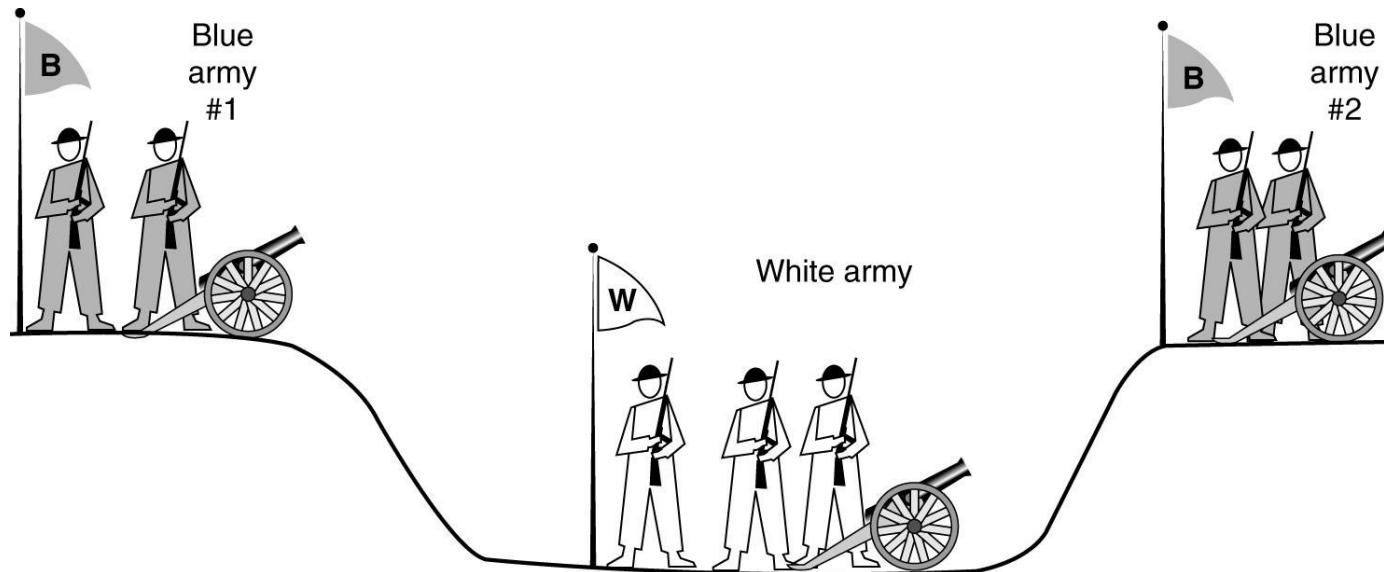
Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. After the connection is established, host 1 sends a segment that arrives properly at host 2. Then host 1 sends another segment. Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.

Clearly, a more sophisticated release protocol is needed to avoid data loss. One way is to use symmetric release, in which each direction is released independently of the other one. Here, a host can continue to receive data even after it has sent a DISCONNECT segment.

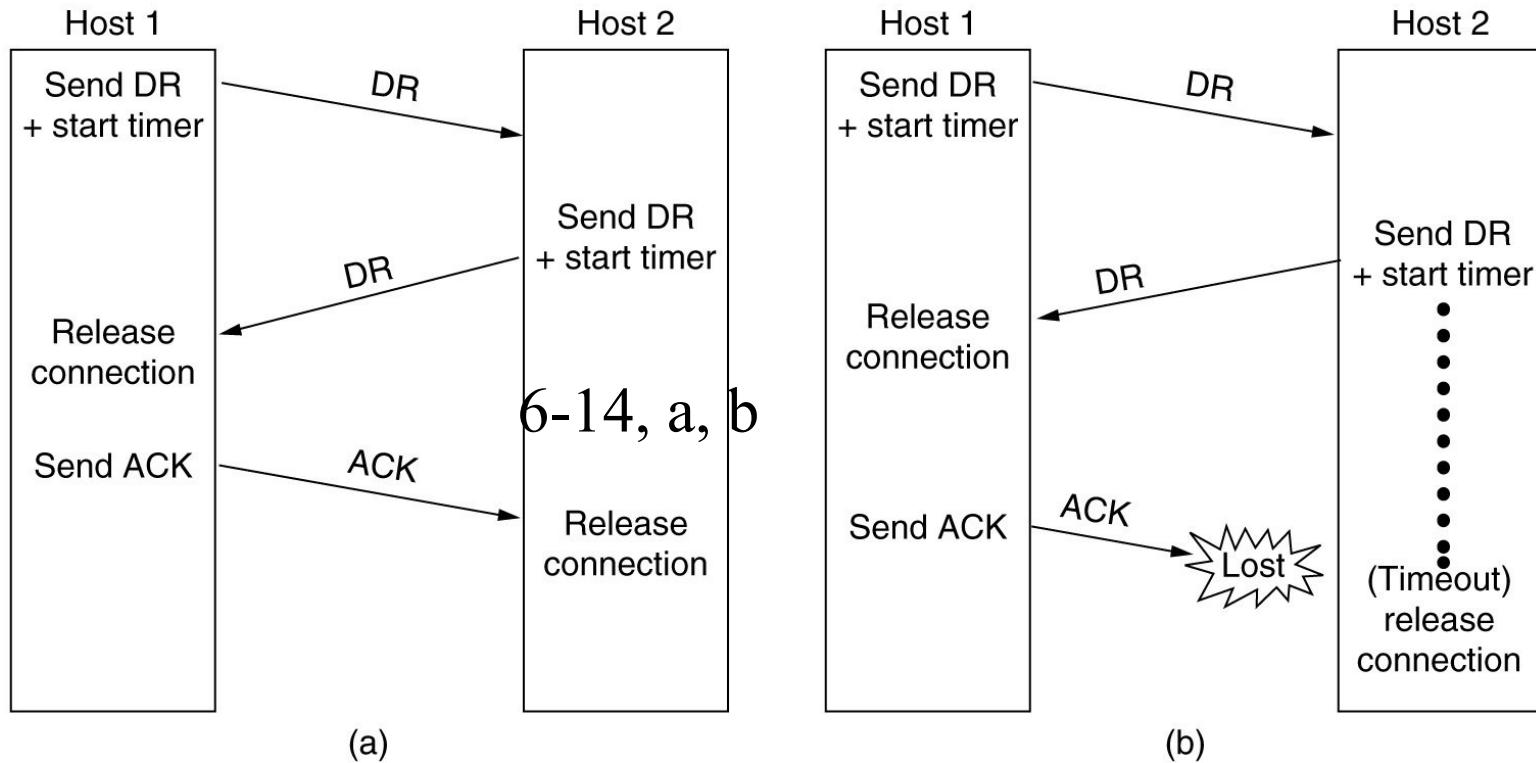
Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it. One can envision a protocol in which host 1 says “I am done. Are you done too?” If host 2 responds: “I am done too. Goodbye, the connection can be safely released.”

# Connection Release

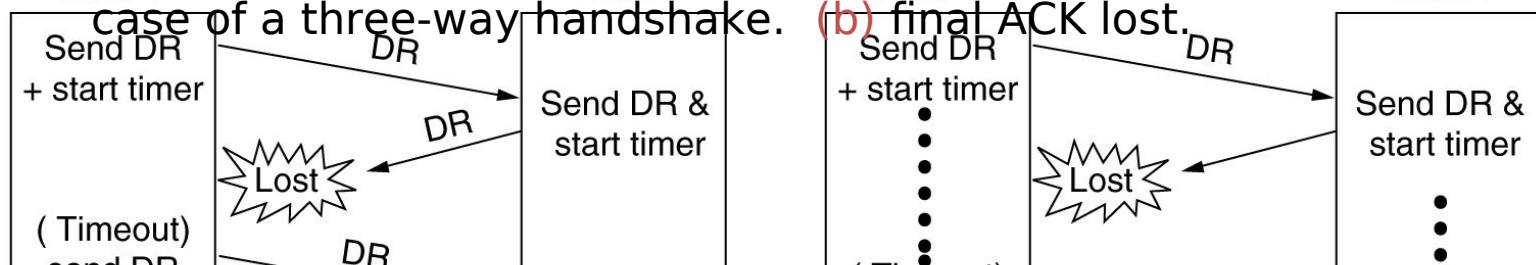
The two-army problem.



# Connection Release



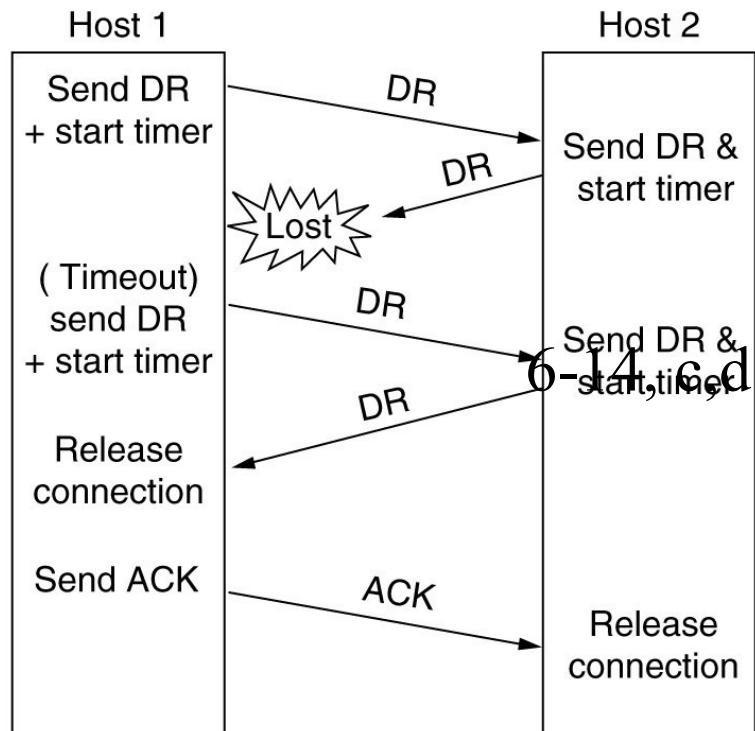
Four protocol scenarios for releasing a connection. (a) Normal case of a three-way handshake. (b) final ACK lost.



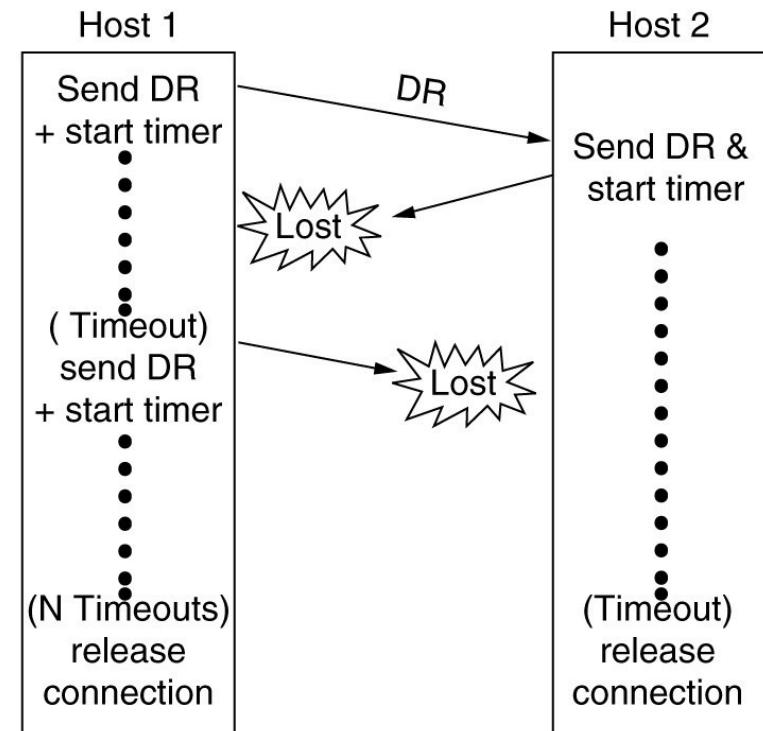
connection

(a)

(b)



(c)



(d)

**(c) Response lost. (d) Response lost and subsequent DRs lost.**

In Fig. (a), we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) segment to initiate the connection release. When it arrives, the recipient sends back a DR segment and starts a timer, just in case its DR is lost. When this DR arrives, the original sender sends back an ACK segment and releases the connection. Finally, when the ACK segment arrives, the receiver also releases the connection.

If the final ACK segment is lost, as shown in Fig.(b), the situation is saved by the timer. When the timer expires, the connection is released anyway. Now consider the case of the second DR being lost. The user initiating the disconnection will not receive the expected response, will time out, and will start all over again.

In Fig.(c), we see how this works, assuming that the second time no segments are lost and all segments are delivered correctly and on time.

Last scenario, Fig.(d), is the same as Fig. (c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost segments. After  $N$  retries, the sender just gives up and releases the connection. Meanwhile, the receiver times out and

# TCP

TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level. In brief, TCP is called a *connection-oriented, reliable transport protocol. It adds* connection-oriented and reliability features to the services of IP.

## *Topics discussed in this section:*

TCP Services

TCP Features

Segment

A TCP Connection

Flow Control

Error Control

# TCP Services

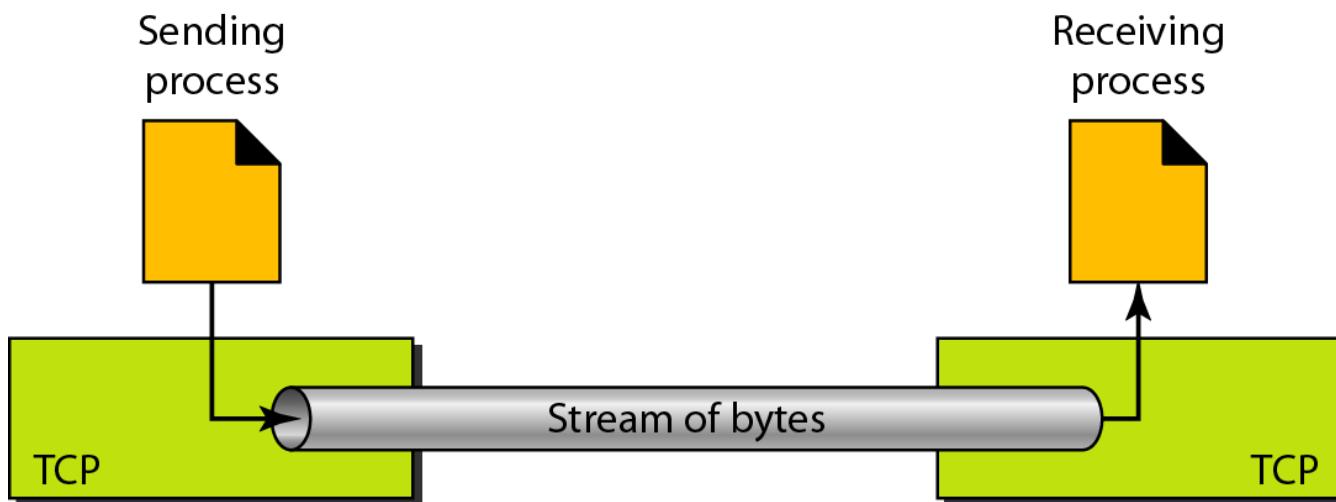
## ***1 Process-to-Process Communication***

TCP provides process-to-process communication using port numbers. Below Table lists some well-known port numbers used by TCP.

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

## **2 Stream Delivery Service**

TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their data across the Internet. This imaginary environment is showed in below Figure. The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them



**3 Sending and Receiving Buffers** Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction. One way to implement a buffer is to use a circular array of 1-byte locations as shown in Figure. For simplicity, we have shown two buffers of 20 bytes each. Normally the buffers are hundreds or thousands of bytes, depending on the implementation. We also show the buffers as the same size, which is not always the case.

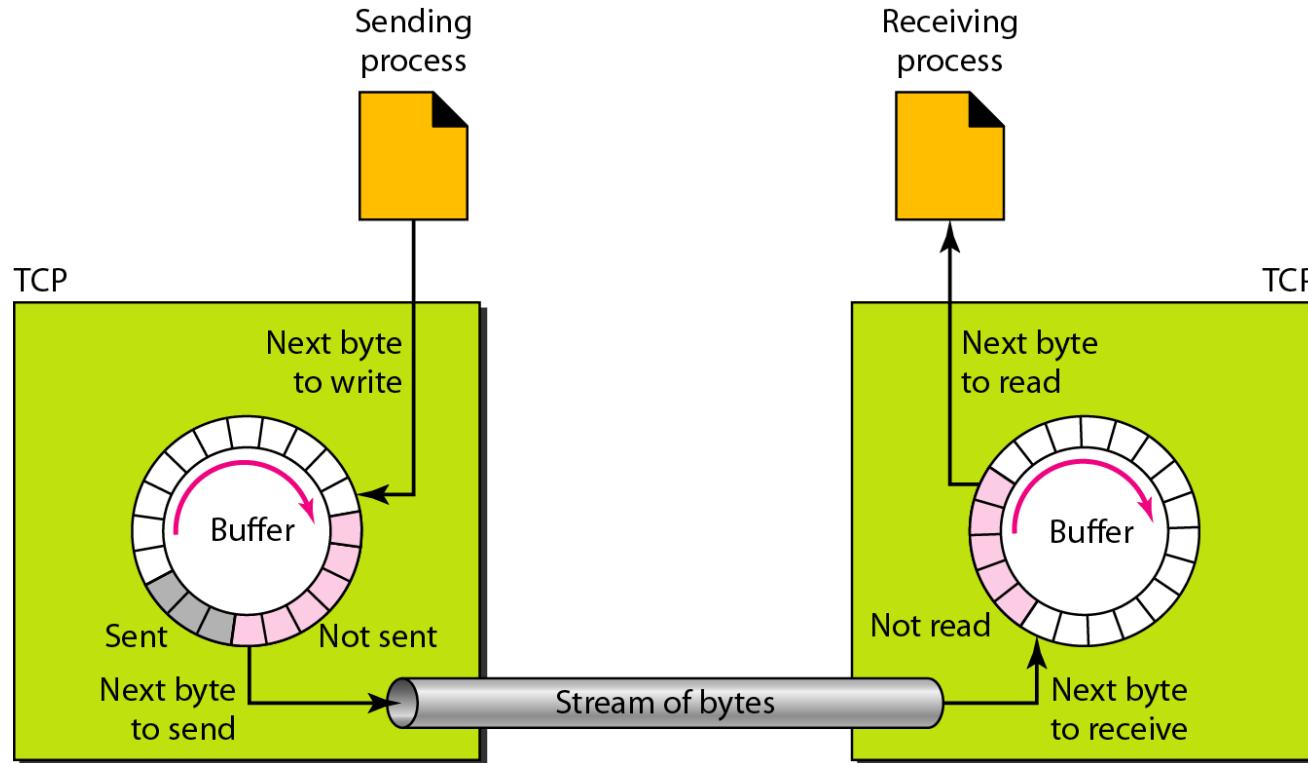


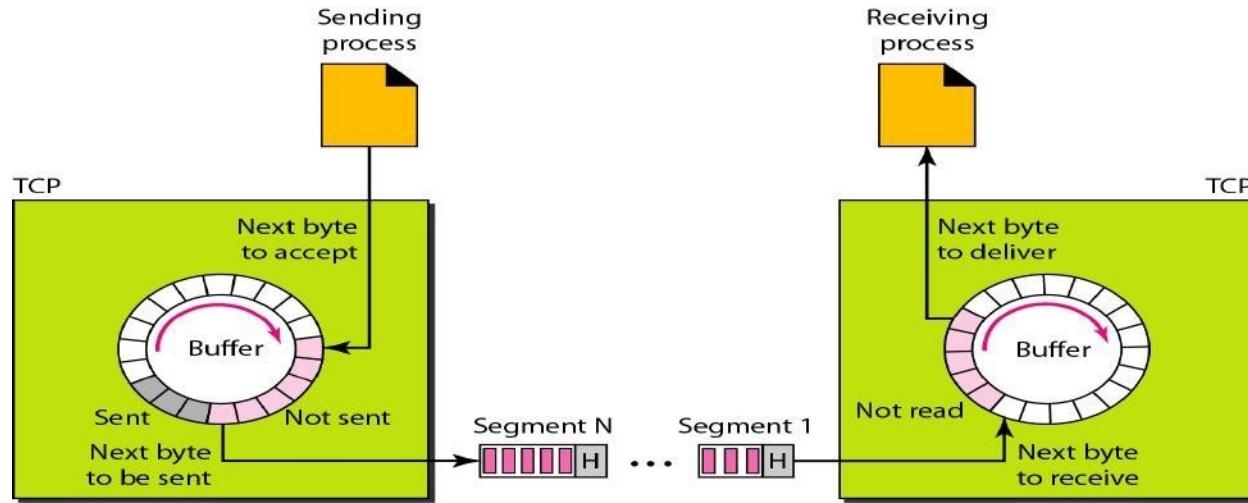
Figure shows the movement of the data in one direction. At the sending site, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer). The gray area holds bytes that have been sent but not yet acknowledged. TCP keeps these bytes in the buffer until it receives an acknowledgment. The colored area contains bytes to be sent by the sending TCP.

However, as we will see later in this chapter, TCP may be able to send only part of this colored section. This could be due to the slowness of the receiving process or perhaps to congestion in the network. Also note that after the bytes in the gray chambers are acknowledged, the chambers are recycled and available for use by the sending process.

This is why we show a circular buffer.

The operation of the buffer at the receiver site is simpler. The circular buffer is divided into two areas (shown as white and colored). The white area contains empty chambers to be filled by bytes received from the network. The colored sections contain received bytes that can be read by the receiving process. When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

## 4 TCP segments



At the transport layer, TCP groups a number of bytes together into a packet called a **segment**. TCP adds a **header** to each segment (for control purposes) and delivers the segment to the IP layer for transmission. The segments are encapsulated in IP datagrams and transmitted.

This entire operation is transparent to the receiving process. Later we will see that segments may be received out of order, lost, or corrupted and resent. All these are handled by TCP with the receiving process unaware of any activities. Above fig shows how segments are created from the bytes in the buffers

## **5 Full-Duplex Communication**

TCP offers full-duplex service, in which data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer, and segments move in both directions

## **6 Connection-Oriented Service**

TCP is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two TCPs establish a connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

## **7 Reliable Service**

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data. We will discuss this feature further in the section on error control.

# TCP Features

## 1 Numbering System

There are two fields called the sequence number and the acknowledgment number. These two fields refer to the byte number and not the segment number.

**Byte Number** The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number. For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056. We will see that byte numbering is used for flow and error control.

**Sequence Number** After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.

**Acknowledgment Number** The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

## **2 Flow Control**

TCP, provides *flow control*. *The receiver of the data controls the amount of data that are to be sent by the sender.* This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

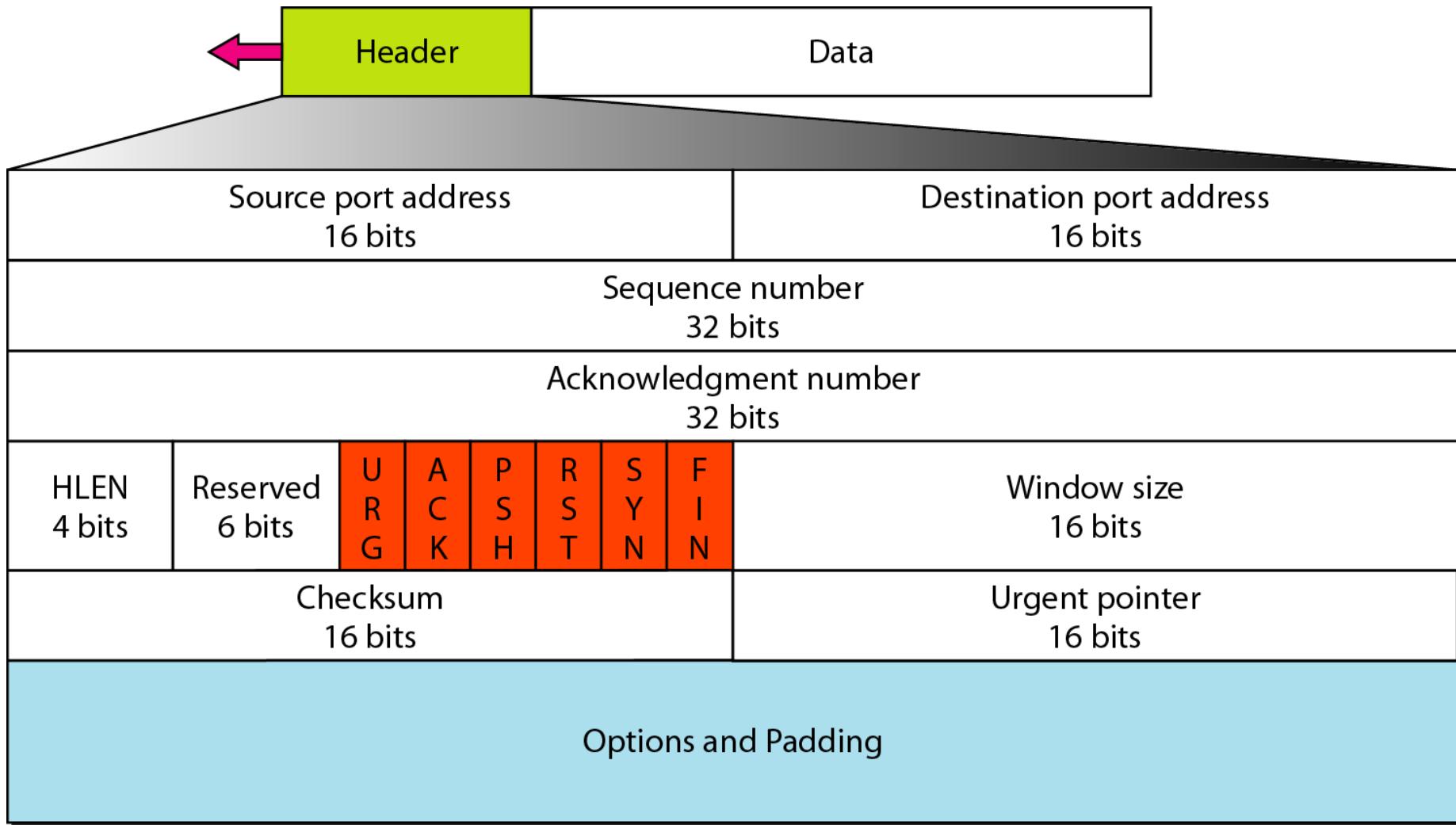
## **3 Error Control**

To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented, as we will see later.

## **4 Congestion Control**

TCP takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network

## TCP segment format



The segment consists of a 20- to 60-byte header.,.

**Source port address.** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

**Destination port address.** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

**Sequence number.** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

**Acknowledgment number.** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number  $x$  *from the other party*, it defines  $x + 1$  as the acknowledgment number. *Acknowledgment and data can be piggybacked together.*

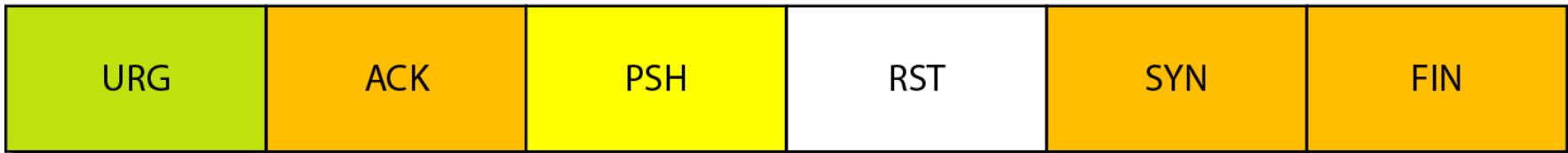
**Header length.** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 ( $5 \times 4 = 20$ ) and 15 ( $15 \times 4 = 60$ ).

**Reserved.** This is a 6-bit field reserved for future use.

**Control.** This field defines 6 different control bits or flags as shown in Figure. One or more of these bits can be set at a time.

URG: Urgent pointer is valid  
ACK: Acknowledgment is valid  
PSH: Request for push

RST: Reset the connection  
SYN: Synchronize sequence numbers  
FIN: Terminate the connection



These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.

**Window size.** This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

**Checksum.** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.

**Urgent pointer.** This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. This will be discussed later in this chapter.

**Options.** There can be up to 40 bytes of optional information in the TCP header. We will not discuss these options here; please refer to the reference list for more information.

## **A TCP Connection**

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All the segments belonging to a message are then sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.

In TCP, connection-oriented transmission requires three phases:

1. connection establishment,
2. data transfer,
3. connection termination.

# TCP connection establishment(3 way handshaking)

1 The client sends the first segment, a SYN segment, in which only the SYN flag is set.

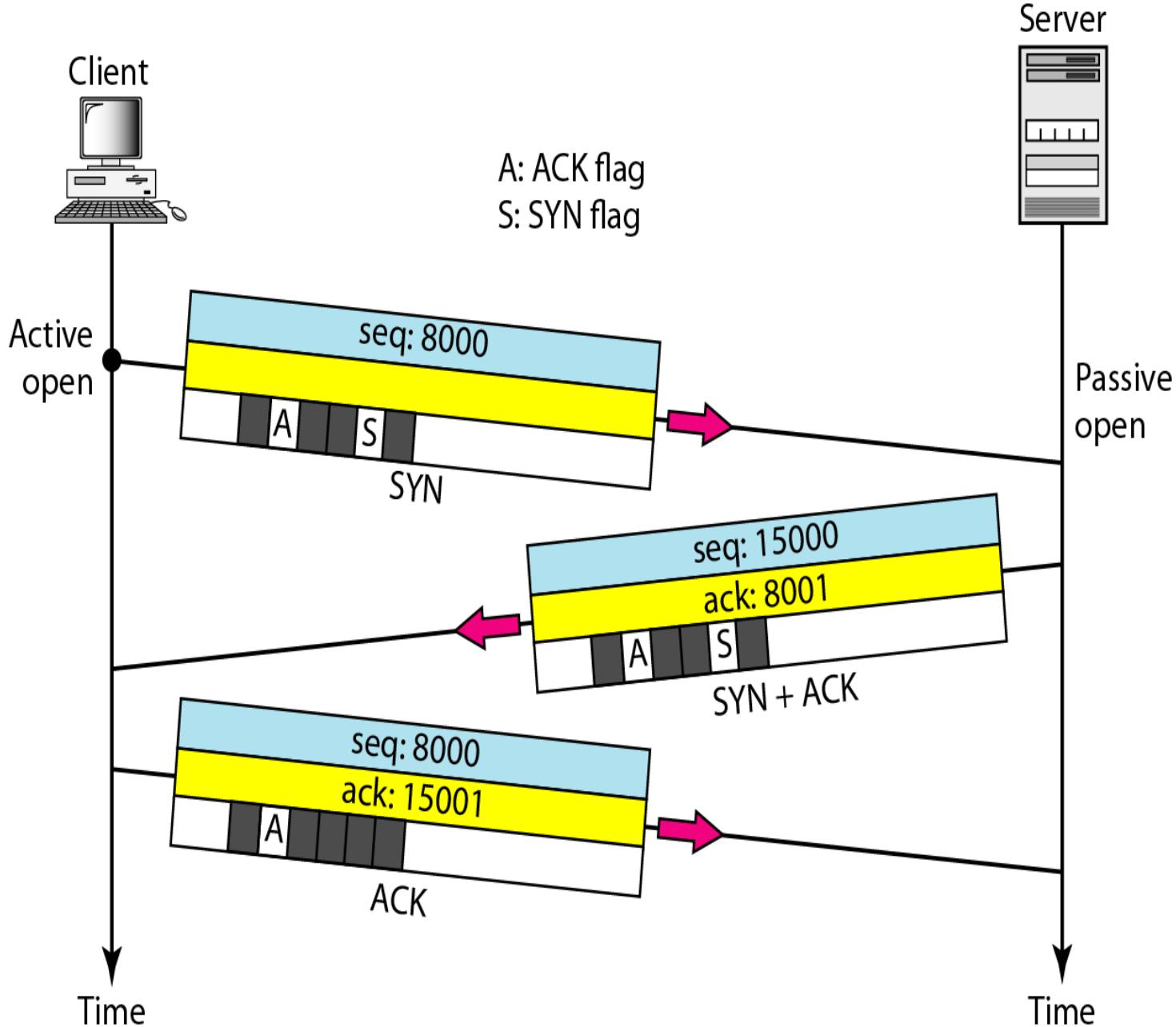
NOTE: A SYN segment cannot carry data, but it consumes one sequence number.

2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

NOTE: A SYN+ACK segment cannot carry data, but does consume one sequence number

3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

NOTE: An ACK segment, if carrying no data, consumes no sequence number



## SYN Flooding Attack

This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming from a different client by faking the source IP addresses in the datagram's.

The server, assuming that the clients are issuing an active open, allocates the necessary resources, such as creating communication tables and setting timers. The TCP server then sends the SYN +ACK segments to the fake clients, which are lost. During this time, however, a lot of resources are occupied without being used. If, during this short time, the number of SYN segments is large, the server eventually runs out of resources and may crash. This SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

### SOLUTIONS:

- 1 Some have imposed a limit on connection requests during a specified period of time.
- 2 Others filter out datagrams coming from unwanted source addresses.
- 3 One recent strategy is to postpone resource allocation until the entire connection is set up using what is called a cookie.

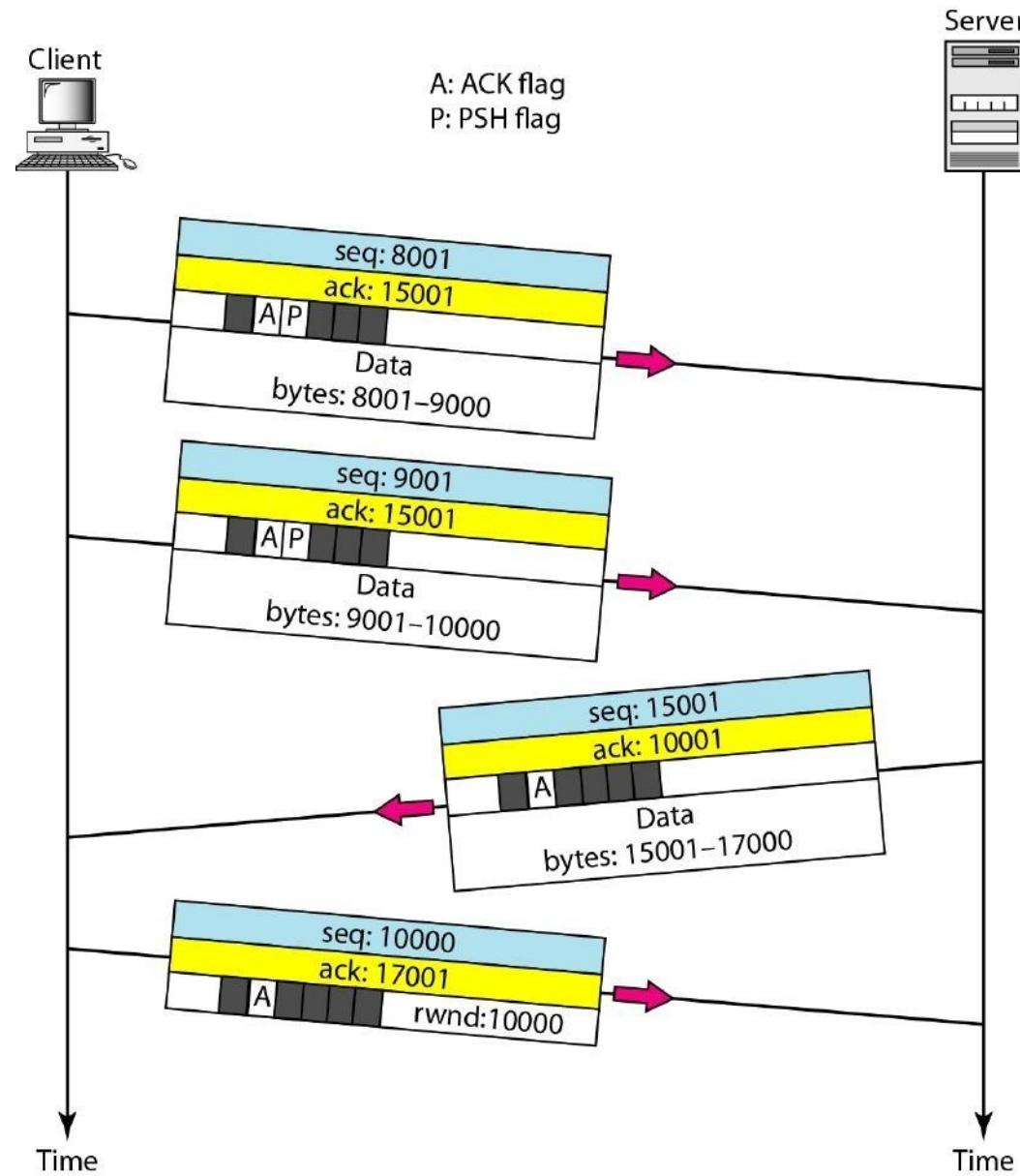
## Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments. Data traveling in the same direction as an acknowledgment are carried on the same segment. The acknowledgment is piggybacked with the data

In this example, after connection is established (not shown in the figure), the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.

Note the values of the sequence and acknowledgment numbers. The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.

## Data transfer



**PUSHING DATA:** Delayed transmission and delayed delivery of data may not be acceptable by the application program.

TCP can handle such a situation. The application program at the sending site can request a *push operation*. *This means that the sending TCP must not wait for the window to be filled.* It must create a segment and send it immediately. The sending TCP must also set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.

**Urgent Data :** TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream. However, sending application program wants a piece of data to be read out of order by the receiving application program.

**Connection Termination** (three-way handshaking and four-way handshaking with a half-close option.)

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set.

Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure. If it is only a control segment, it consumes only one sequence number.

NOTE: The FIN segment consumes one sequence number if it does not carry data.

2 The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN +ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

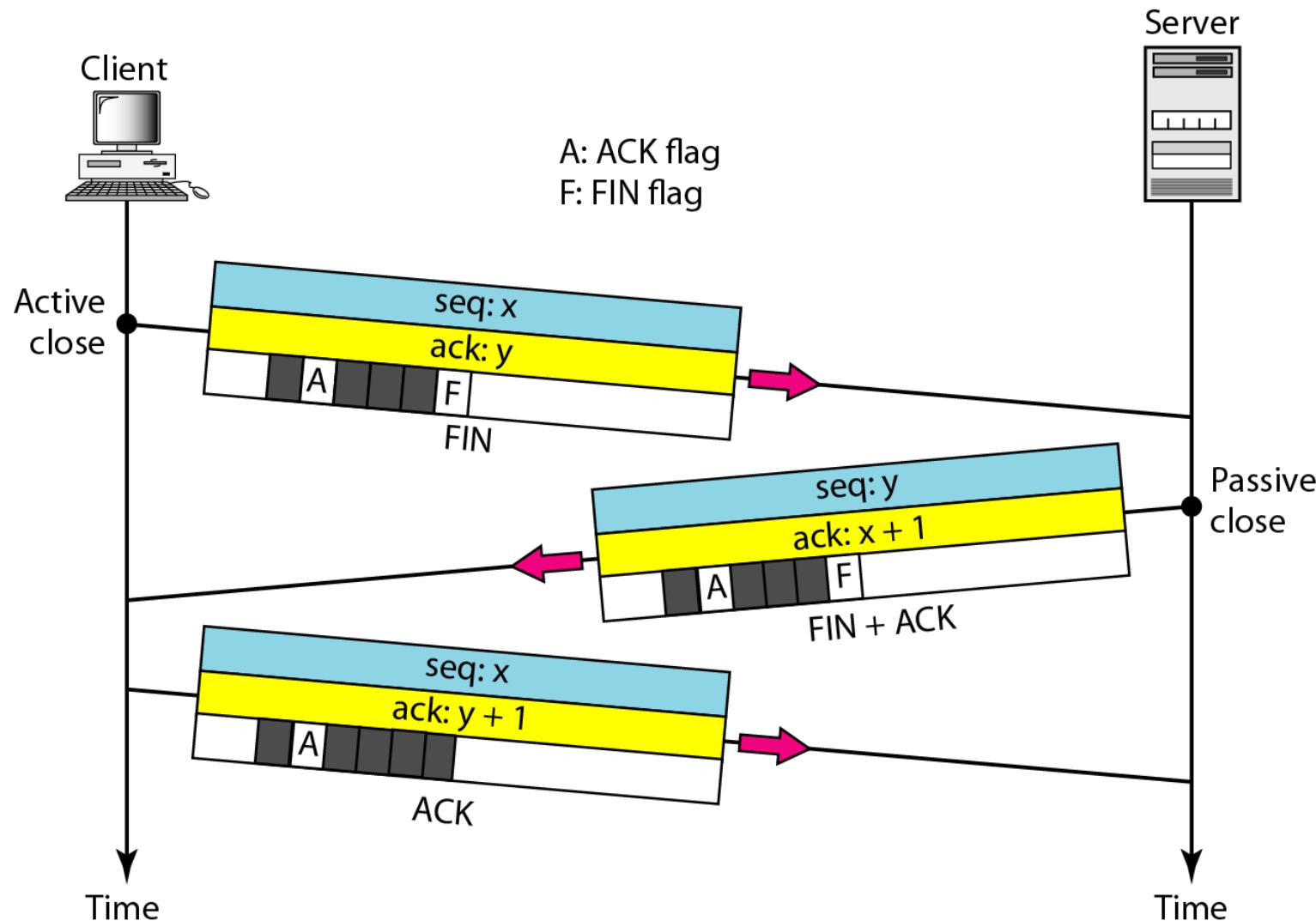
NOTE: The FIN +ACK segment consumes one sequence number if it does not carry data.

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

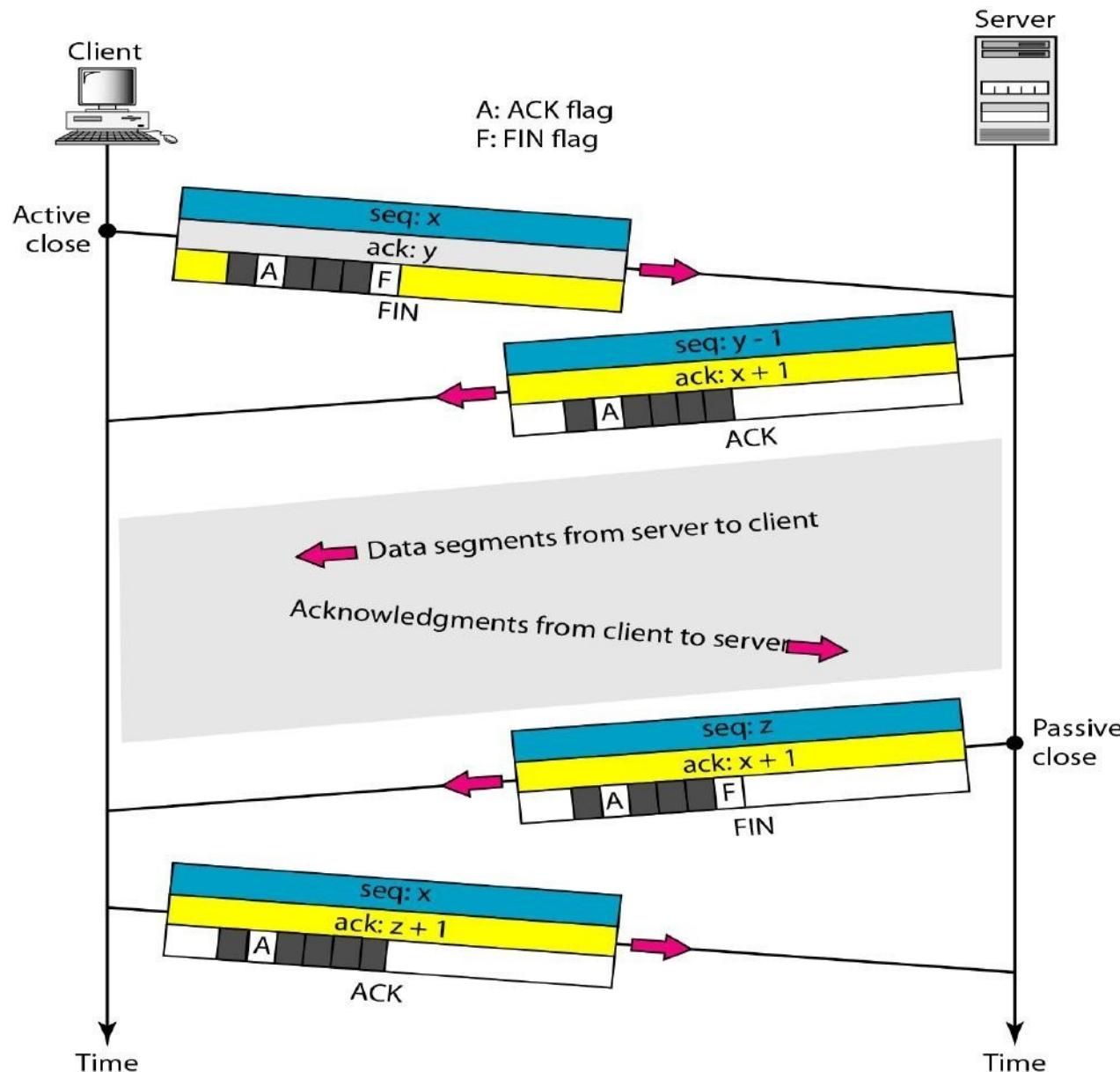
**Half-Close** In TCP, one end can stop sending data while still receiving data. This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin.

A good example is sorting. When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start. This means the client, after sending all the data, can close the connection in the outbound direction. However, the inbound direction must remain open to receive the sorted data. The server, after receiving the data, still needs time for sorting; its outbound direction must remain open

## Connection termination using three-way handshaking



## Figure 23.21 Half-close



## Flow Control or TCP Sliding Window

TCP uses a sliding window, to handle flow control. The sliding window protocol used by TCP, however, is something between the *Go-Back-N* and Selective Repeat sliding window.

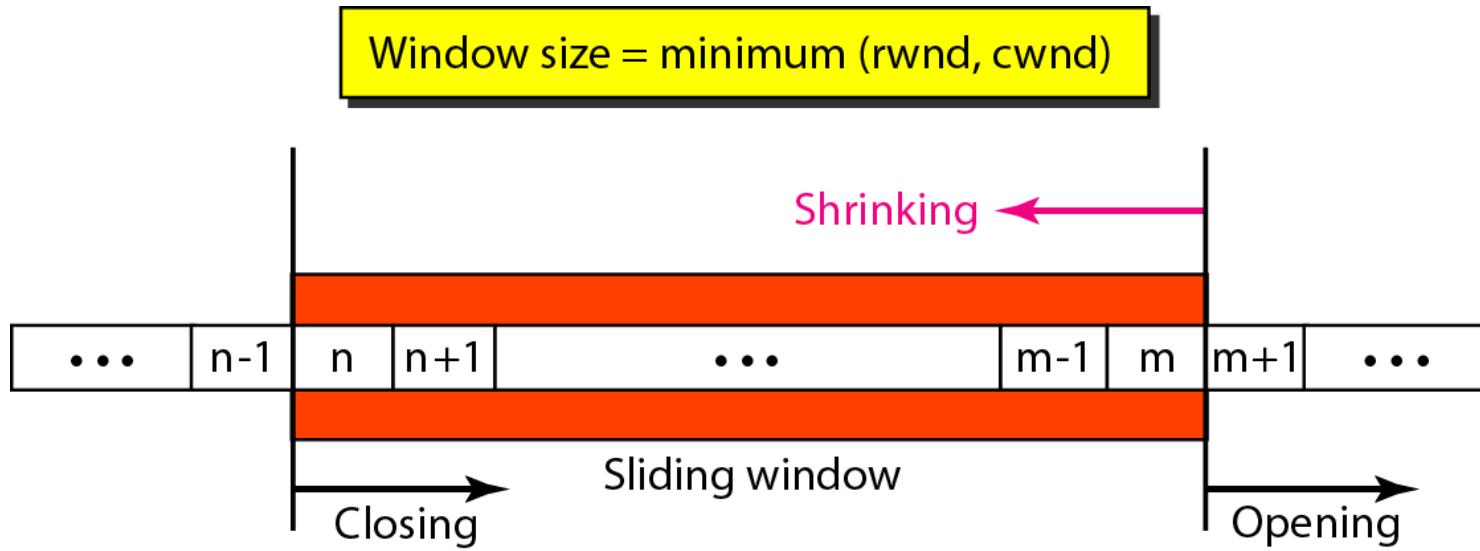
The sliding window protocol in TCP looks like the *Go-Back-N* protocol because it does not use NAKs;

it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.

There are two big differences between this sliding window and the one we used at the data link layer.

- 1 the sliding window of TCP is byte-oriented; the one we discussed in the data link layer is frame-oriented.
- 2 the TCP's sliding window is of variable size; the one we discussed in the data link layer was of fixed size

## *Sliding window*



The window is opened, closed, or shrunk. These three activities, as we will see, are in the control of the receiver (and depend on congestion in the network), not the sender.

The sender must obey the commands of the receiver in this matter.

Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending.

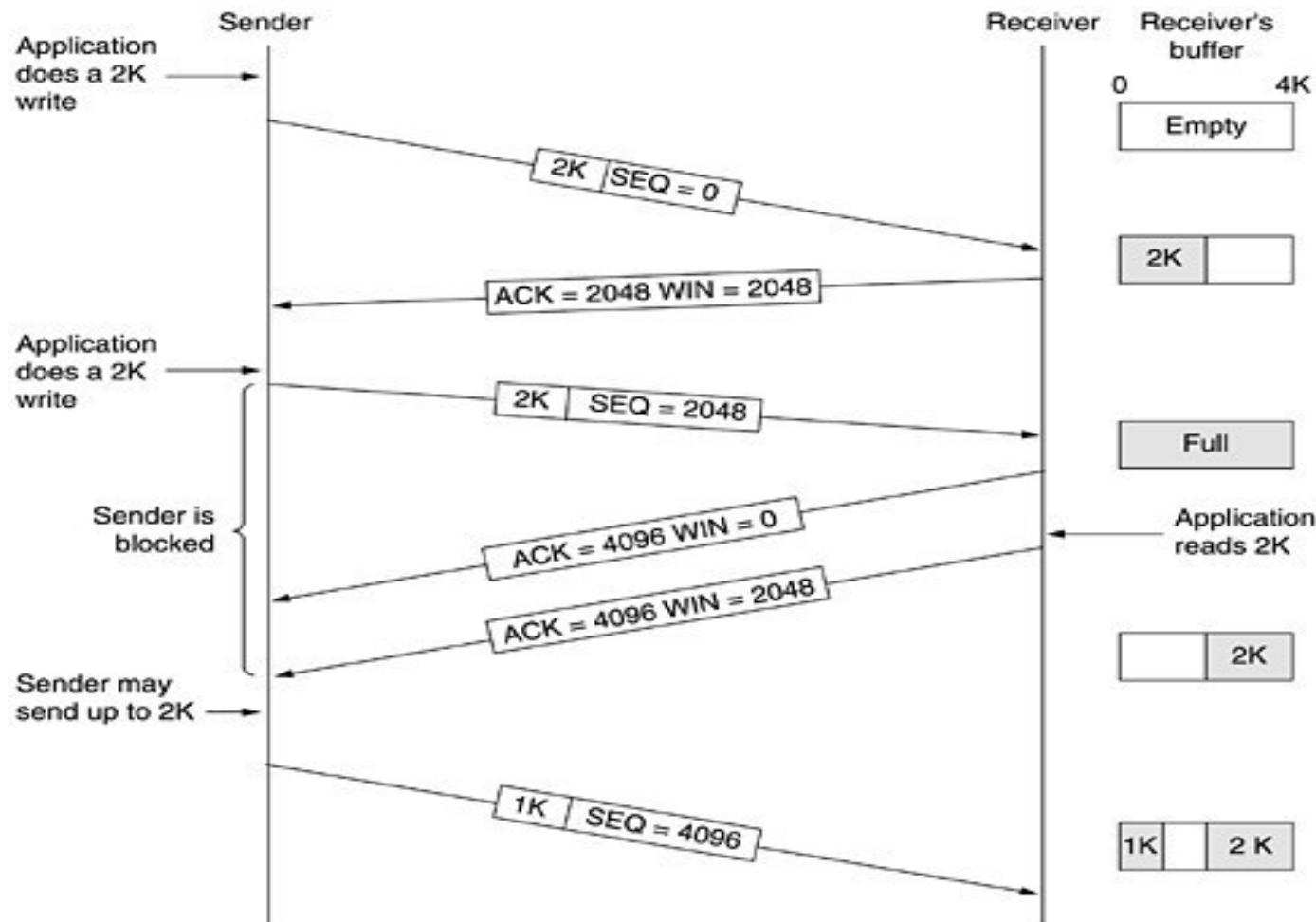
Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore.

Shrinking the window means moving the right wall to the left.

The size of the window at one end is determined by the lesser of two values: receiver window (rwnd) or congestion window (cwnd).

The receiver window is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded.

The congestion window is a value determined by the network to avoid congestion



Window management in TCP

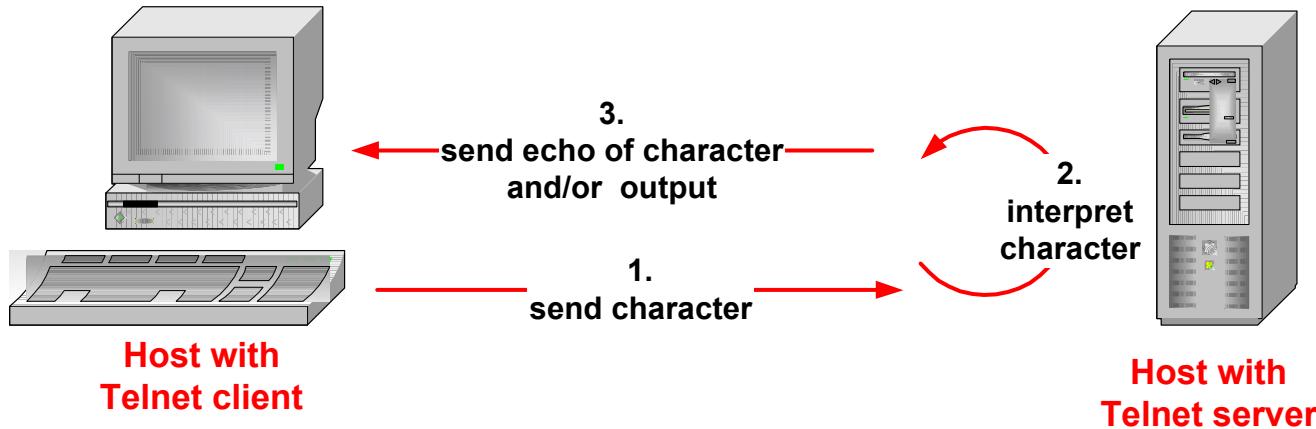
When the window is 0, the sender may not normally send segments, with two exceptions.

- 1) urgent data may be sent, for example, to allow the user to kill the process running on the remote machine.
- 2) the sender may send a 1-byte segment to force the receiver to reannounce the next byte expected and the window size. This packet is called a **window probe**.

The TCP standard explicitly provides this option to prevent deadlock if a window update ever gets lost.

Senders are not required to transmit data as soon as they come in from the application. Neither are receivers required to send acknowledgements as soon as possible.

For example, in Fig. when the first 2 KB of data came in, TCP, knowing that it had a 4-KB window, would have been completely correct in just buffering the data until another 2 KB came in, to be able to transmit a segment with a 4-KB payload. This freedom can be used to improve performance



Remote terminal applications (e.g., Telnet) send characters to a server. The server interprets the character and sends the output at the server to the client.

For each character typed, you see three packets:

**Client → Server:** Send typed character

**Server → Client:** Echo of character (or user output) and acknowledgement for first packet

**Client → Server:** Acknowledgement for second packet

# Delayed Acknowledgement

- TCP delays transmission of ACKs for up to 500ms
- Avoid to send ACK packets that do not carry data.
  - The hope is that, within the delay, the receiver will have data ready to be sent to the receiver. Then, the ACK can be piggybacked with a data segment

## Exceptions:

- ACK should be sent for every full sized segment
- Delayed ACK is not used when packets arrive out of order

Although delayed acknowledgements reduce the load placed on the network by the receiver, a sender that sends multiple short packets (e.g., 41-byte packets containing 1 byte of data) is still operating inefficiently. A way to reduce this usage is known as **Nagle's algorithm (Nagle, 1984)**.

## Nagel's Rule

Send one byte and buffer all subsequent bytes until acknowledgement is received. Then send all buffered bytes in a single TCP segment and start buffering again until the sent segment is acknowledged.

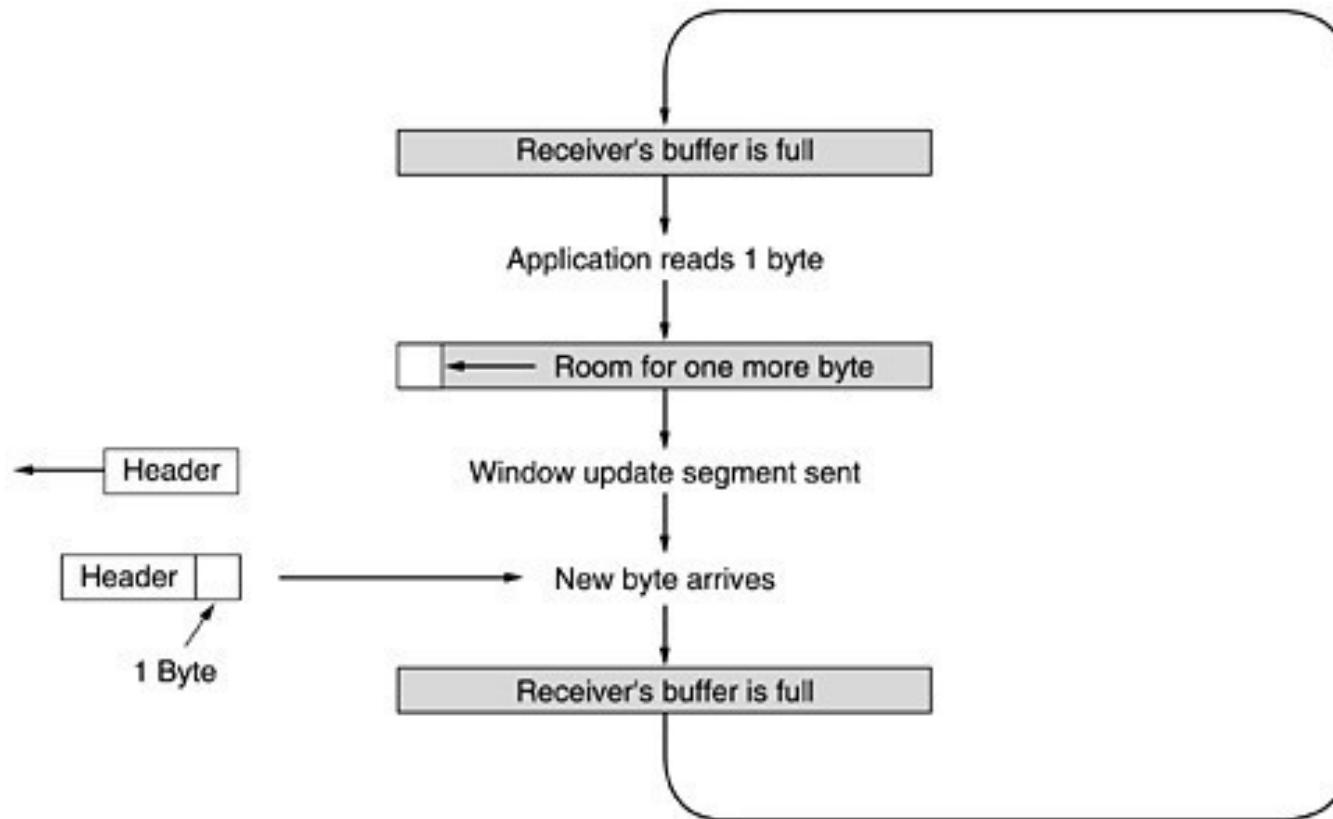
Nagle's algorithm will put the many pieces in one segment, greatly reducing the bandwidth used

Nagle's algorithm is widely used by TCP implementations, but there are times when it is better to disable it. In particular, in interactive games that are run over the Internet.

A more subtle problem is that Nagle's algorithm can sometimes interact with delayed acknowledgements to cause a temporary deadlock: the receiver waits for data on which to piggyback an acknowledgement, and the sender waits on the acknowledgement to send more data.

Because of these problems, Nagle's algorithm can be disabled (which is called the *TCP NODELAY option*).

Another problem that can degrade TCP performance is the **silly window syndrome** (Clark, 1982).



Clark's solution is to prevent the receiver from sending a window update for 1 byte. Instead, it is forced to wait until it has a decent amount of space available and advertise that instead. Specifically, the receiver should not send a window update until it can handle the maximum segment size it advertised when the connection was established or until its buffer is half empty, whichever is smaller.

Furthermore, the sender can also help by not sending tiny segments. Instead, it should wait until it can send a full segment, or at least one containing half of the receiver's buffer size.

The goal is for the sender not to send small segments and the receiver not to ask for them. (Nagel + Clark). Both are used to improve TCP performance

The receiver will buffer the data until it can be passed up to the application in order (handling out of order segments)

## **Cumulative acknowledgements**

## Error Control

TCP is a reliable transport layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.

TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools: **checksum, acknowledgment, and time-out.**

### Checksum

Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment

**Figure 23.11** *Checksum calculation of a simple UDP user datagram*

153.18.8.105		
171.2.14.10		
All 0s	17	15
1087		13
15		All 0s
T	E	S
I	N	G
All 0s		

10011001 00010010	→	153.18
00001000 01101001	→	8.105
10101011 00000010	→	171.2
00001110 00001010	→	14.10
00000000 00010001	→	0 and 17
00000000 00001111	→	15
00000100 00111111	→	1087
00000000 00001101	→	13
00000000 00001111	→	15
00000000 00000000	→	0 (checksum)
01010100 01000101	→	T and E
01010011 01010100	→	S and T
01001001 01001110	→	I and N
01000111 00000000	→	G and 0 (padding)
<hr/>		
10010110 11101011	→	Sum
01101001 00010100	→	Checksum

## **Acknowledgment**

TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged. ACK segments do not consume sequence numbers and are not acknowledged.

## **Retransmission**

The heart of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost, or delayed, it is retransmitted.

In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.

Retransmission After RTO (retransmission time out)

Retransmission After Three Duplicate ACK Segments (also called fast retransmission)

## **Out-of-Order Segments**

Data may arrive out of order and be temporarily stored by the receiving TCP, but yet guarantees that no out-of-order segment is delivered to the process

# TCP Congestion Control

When the load offered to any network is more than it can handle, congestion builds up.

The network layer detects congestion when queues grow large at routers and tries to manage it, if only by dropping packets. It is up to the transport layer to receive congestion feedback from the network layer and slow down the rate of traffic that it is sending into the network.

For Congestion control, transport protocol uses an AIMD (Additive Increase Multiplicative Decrease) control law.

TCP congestion control is based on implementing this approach using a window called **congestion window**. TCP adjusts the size of the window according to the AIMD rule.

The window size at the sender is set as follows:

**Send Window = MIN (flow control window, congestion window)**

where

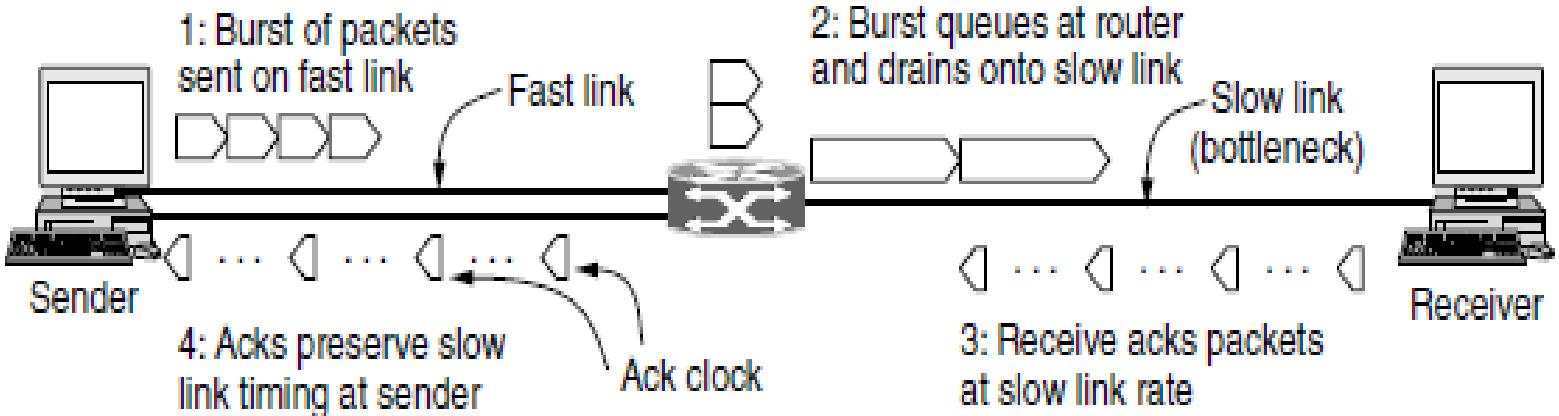
**flow control window** is advertised by the receiver (rwnd)

**congestion window** is adjusted based on feedback from the

Modern congestion control was added to TCP largely through the efforts of Van Jacobson (1988). It is a fascinating story. Starting in 1986, the growing popularity of the early Internet led to the first occurrence of what became known as a **congestion collapse**, a prolonged period during which good put dropped suddenly (i.e., by more than a factor of 100) due to congestion in the network. Jacobson (and many others) set out to understand what was happening and remedy the situation.

To start, he observed that packet loss is a suitable signal of congestion. This signal comes a little late (as the network is already congested) but it is quite dependable

At the beginning how sender knows at what speed receiver can receive the packets?

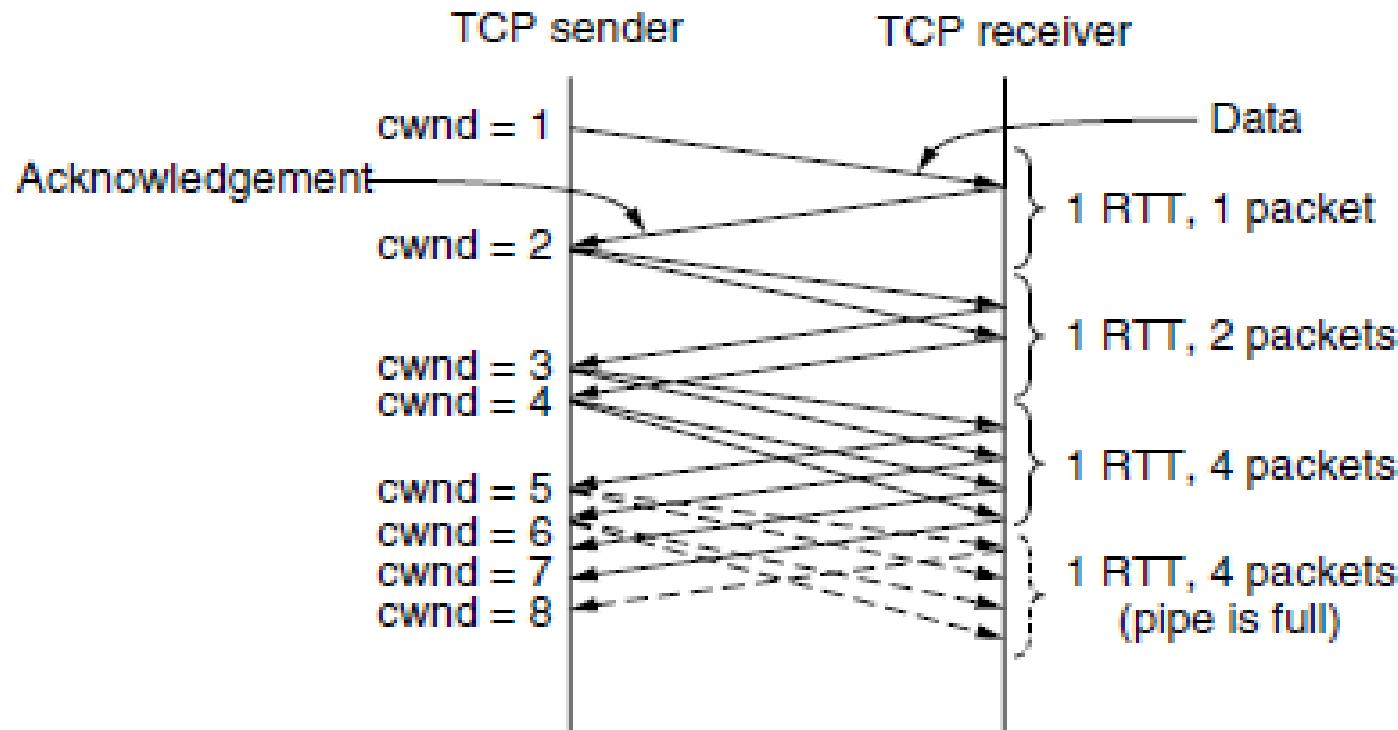


The key observation is this: the acknowledgements return to the sender at about the rate that packets can be sent over the slowest link in the path. This is precisely the rate that the sender wants to use. If it injects new packets into the network at this rate, they will be sent as fast as the slow link permits, but they will not queue up and congest any router along the path. This timing is known as an **ack clock**. It is an essential part of TCP. By using an ack clock, TCP smoothes out traffic and avoids unnecessary queues at routers. This is first consideration

A second consideration is that the AIMD rule will take a very long time to reach a good operating point on fast networks if the congestion window is started from a small size

Instead, the solution Jacobson chose to handle both of these considerations is a mix of linear and multiplicative increase.

## **SLOW-START**



# TCP Congestion Control

## Slow Start

- Additive Increase / Multiplicative Decrease is only suitable for source, that is operating close to the available capacity of the network, but it takes too long to ramp up a connection when it is starting from scratch.
- slow start, that is used to increase the congestion window rapidly from a cold start.
- Slow start effectively **increases the congestion window exponentially**, rather than linearly.
  - the source starts out by setting CongestionWindow to one packet.
  - When the ACK for this packet arrives, TCP adds 1 to CongestionWindow and then sends two packets.
  - Upon receiving the corresponding two ACKs, TCP increments CongestionWindow by 2—one for each ACK—and next sends four packets.
  - The end result is that TCP effectively doubles the number of packets it has in transit every RTT.

Whenever a packet loss is detected, for example, by a timeout, the slow start threshold is set to be half of the congestion window and the entire process is restarted.

Congestion avoidance phase is started if cwnd has reached the slow start threshold value

Whenever the slow start threshold is crossed, TCP switches from slow start to additive increase. In this mode, the congestion window is increased by one segment every round-trip time.

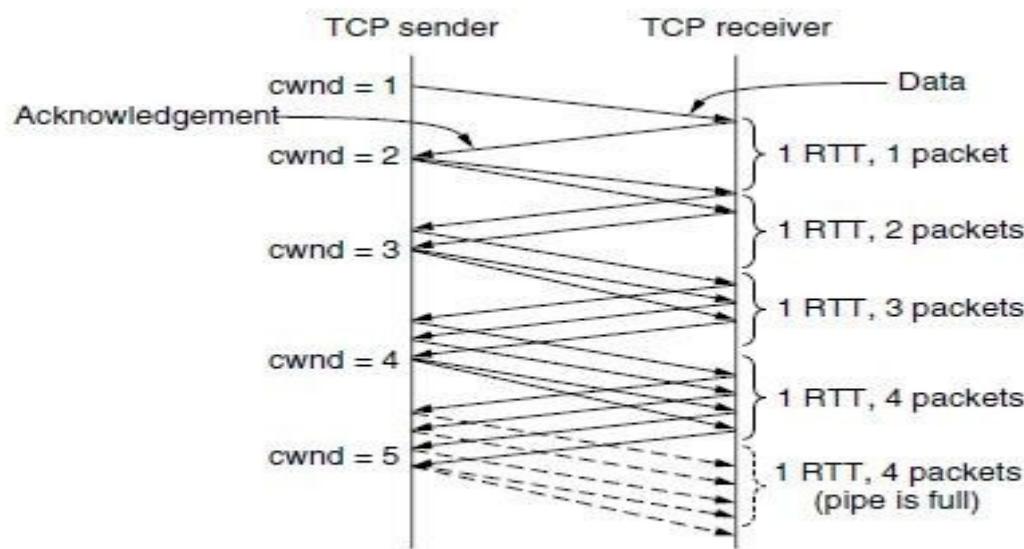


Figure 6-45. Additive increase from an initial congestion window of one segment.

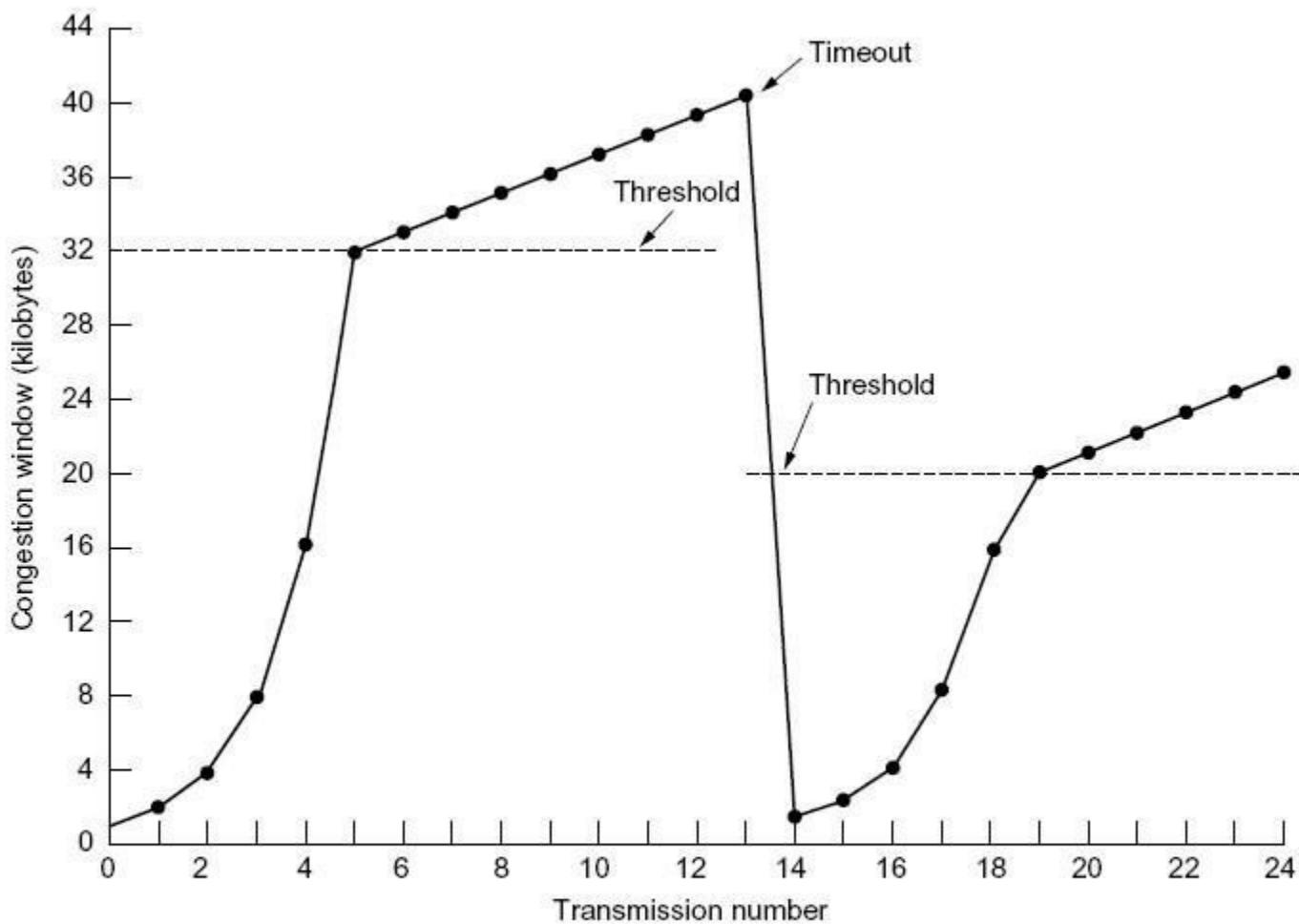


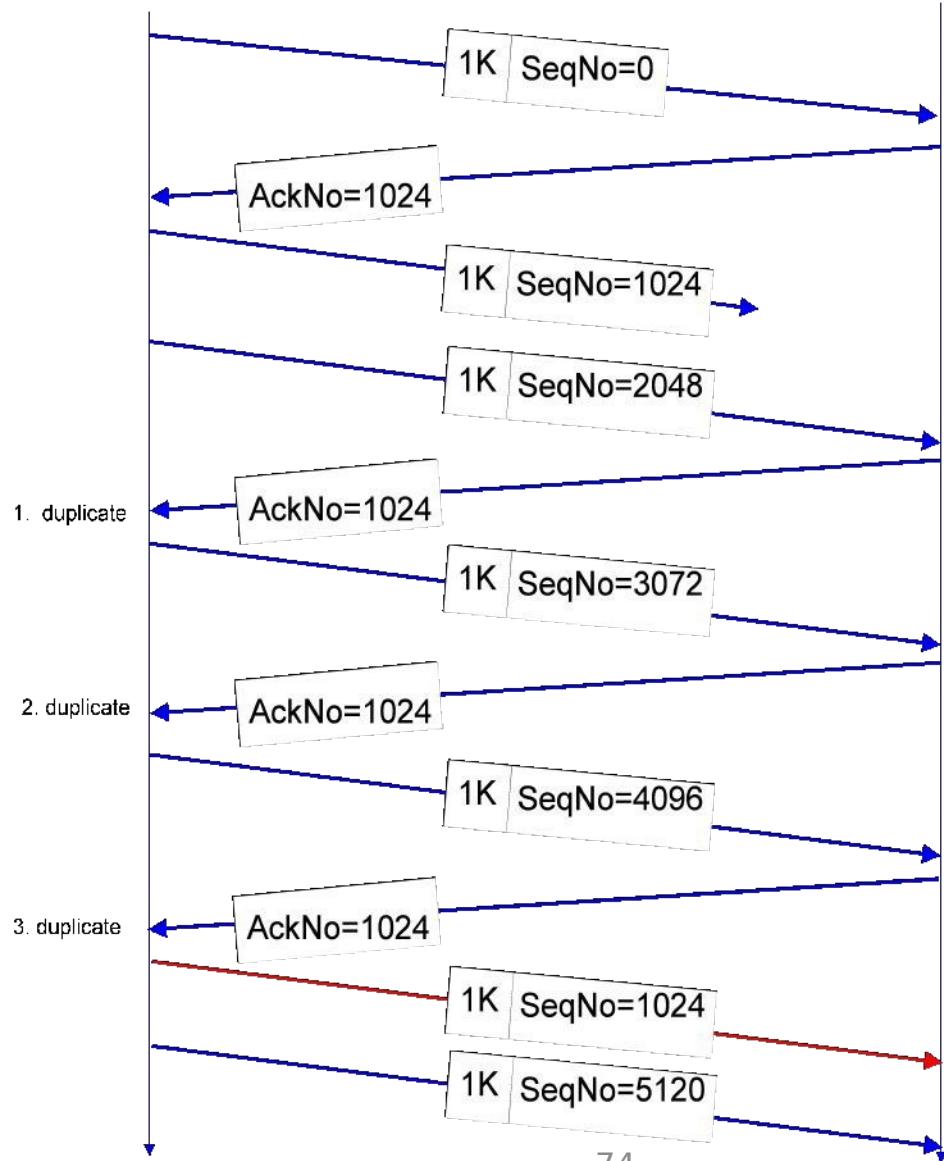
Fig. 6-37. An example of the Internet congestion algorithm.

# Responses to Congestion

- So, TCP assumes there is congestion if it detects a packet loss
- A TCP sender can detect lost packets via:
  - Timeout of a retransmission timer
  - Receipt of a duplicate ACK
- TCP interprets a Timeout as a binary congestion signal. When a timeout occurs, the sender performs:
  - cwnd is reset to one:  
 $cwnd = 1$
  - ssthresh is set to half the current size of the congestion window:  
 $ssthresh = cwnd / 2$
  - and slow-start is entered

# Fast Retransmit

- If three or more duplicate ACKs are received in a row, the TCP sender believes that a segment has been lost.
- Then TCP performs a retransmission of what seems to be the missing segment, without waiting for a timeout to happen.
- Enter slow start:  
 $\text{ssthresh} = \text{cwnd}/2$   
 $\text{cwnd} = 1$



# Flavors of TCP Congestion Control

- **TCP Tahoe** (1988)
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
- **TCP Reno** (1990) (TCP Tahoe+FR)
  - Fast Recovery
- **New Reno** (1996)
- **SACK** (1996) (**SACK (Selective ACKnowledgements)**)
- **RED** (Floyd and Jacobson 1993)

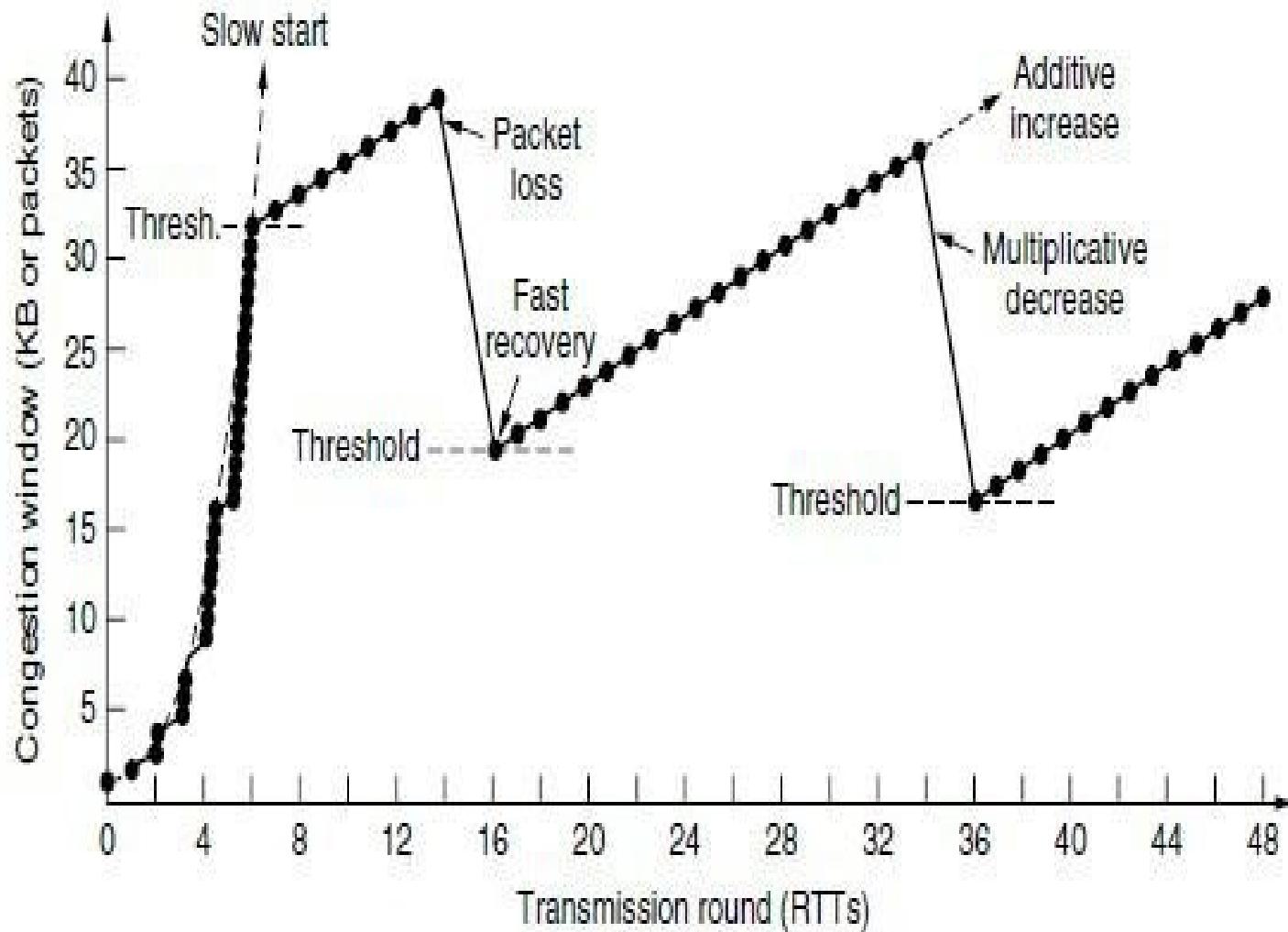


Figure 6-47. Fast recovery and the sawtooth pattern of TCP Reno.

The use of ECN (Explicit Congestion Notification) in addition to packet loss as a congestion signal. ECN is an IP layer mechanism to notify hosts of congestion.

The sender tells the receiver that it has heard the signal by using the CWR (*Congestion Window Reduced*) flag.

# USER DATAGRAM PROTOCOL (UDP)

*The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.*

## Topics discussed in this section:

Well-Known Ports for UDP

User Datagram

Checksum

UDP Operation

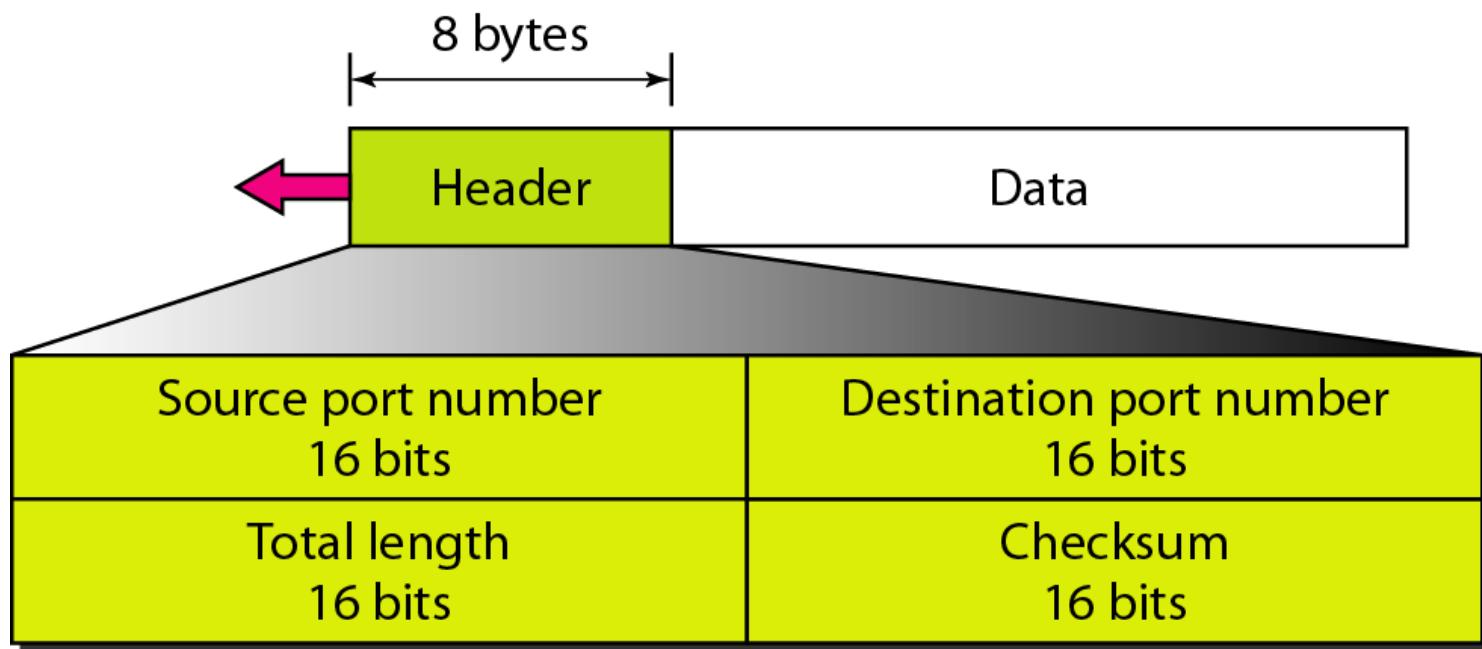
Use of UDP

**Table 23.1** *Well-known ports used with UDP*

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Figure 23.9 *User datagram format*

---



## **Checksum** (OPTIONAL, IF NOT USED SET ALL 1'S DEFAULT)

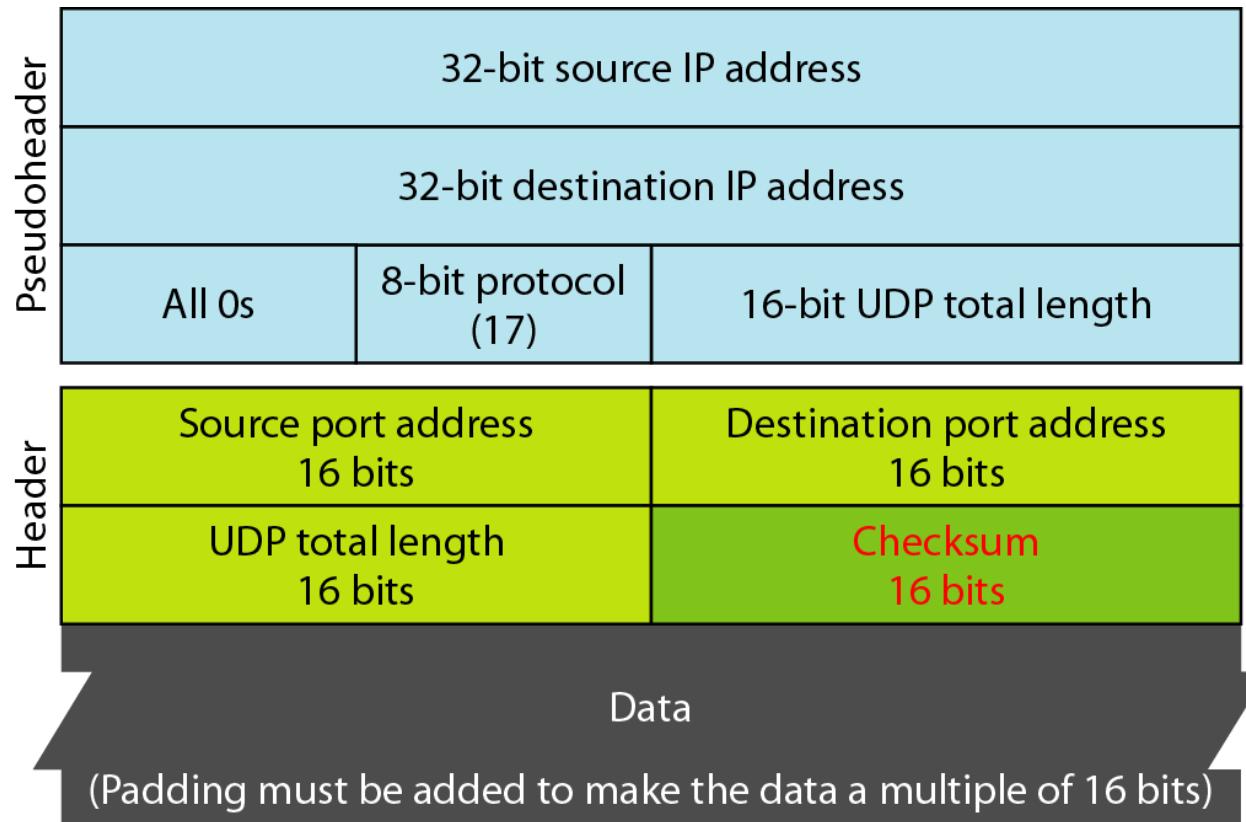
The UDP checksum calculation is different from the one for IP and ICMP. Here the checksum includes three sections: a pseudo header, the UDP header and the data coming from the application layer.

The pseudo header is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with Os

If the checksum does not include the pseudo header, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.

The protocol field is added to ensure that the packet belongs to UDP, and not to other transport-layer protocols.

**Figure 23.10 Pseudoheader for checksum calculation**



# **UDP Operation**

## **Connectionless Services**

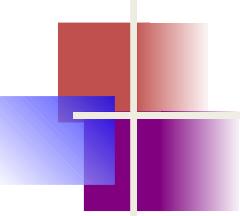
UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program. The user datagrams are not numbered. Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path.

## **Flow and Error Control**

UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages. There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded. The lack of flow control and error control

## **Encapsulation and Decapsulation**

To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.



## *Example 23.2*

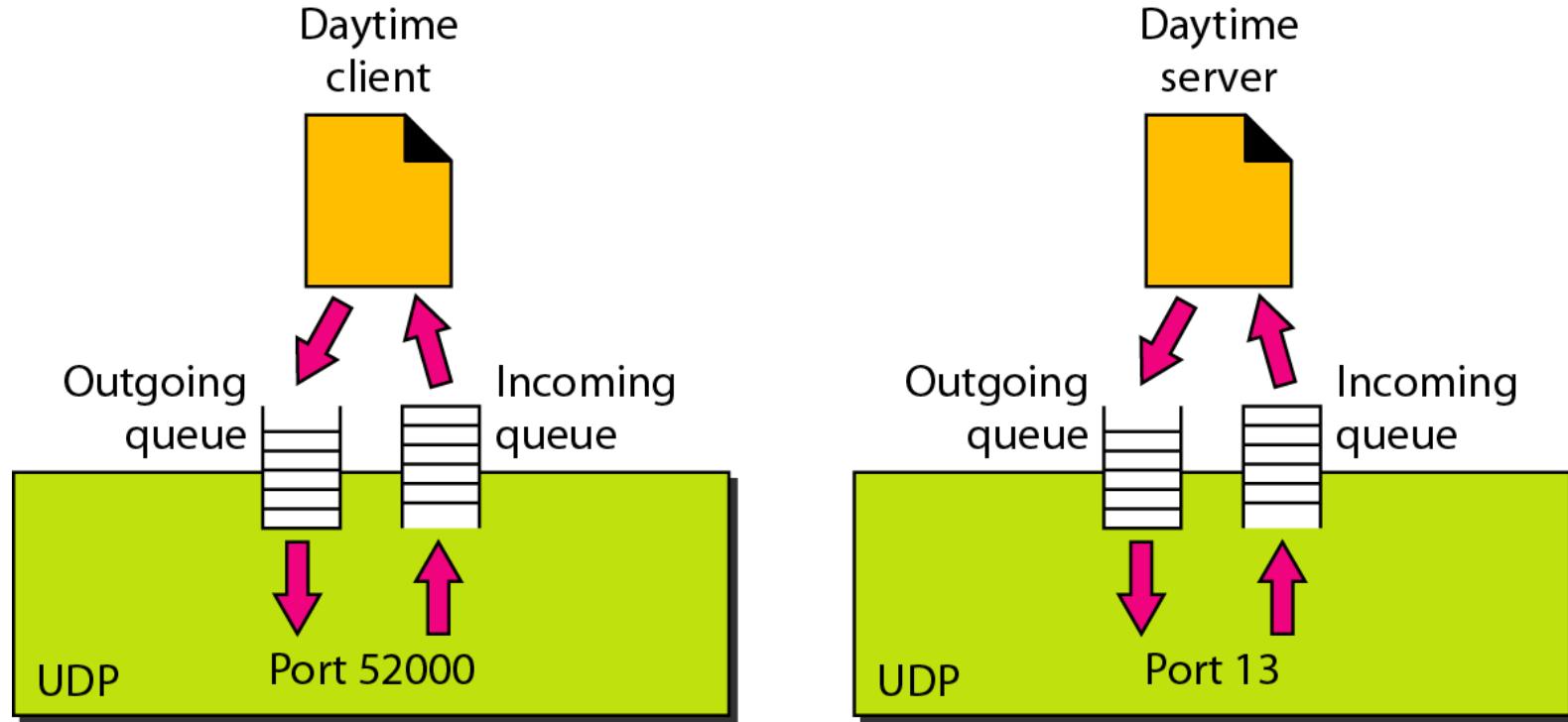
*Figure 23.11 shows the checksum calculation for a very small user datagram with only 7 bytes of data. Because the number of bytes of data is odd, padding is added for checksum calculation. The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.*

**Figure 23.11** Checksum calculation of a simple UDP user datagram

153.18.8.105		
171.2.14.10		
All 0s	17	15
1087		13
15		All 0s
T	E	S
I	N	G
All 0s		

10011001 00010010	→	153.18
00001000 01101001	→	8.105
10101011 00000010	→	171.2
00001110 00001010	→	14.10
00000000 00010001	→	0 and 17
00000000 00001111	→	15
00000100 00111111	→	1087
00000000 00001101	→	13
00000000 00001111	→	15
00000000 00000000	→	0 (checksum)
01010100 01000101	→	T and E
01010011 01010100	→	S and T
01001001 01001110	→	I and N
01000111 00000000	→	G and 0 (padding)
<hr/>		
10010110 11101011	→	Sum
01101001 00010100	→	Checksum

Figure 23.12 *Queues in UDP*



# Remote Procedure Call

The key work was done by Birrell and Nelson (1984). In a nutshell, what Birrell and Nelson suggested was allowing programs to call procedures located on remote hosts. When 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)**. Traditionally, the calling procedure is known as the client and the called procedure is known as the server, and we will use those names here too.

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**. These procedures hide the fact that the procedure call from the client to the server is not local

Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.

Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.

Step 3 is the operating system sending the message from the client machine to the server machine.

Step 4 is the operating system passing the incoming packet to the server stub.

Finally, step 5 is the server stub calling the server procedure with the unmarshaled parameters.

The reply traces the same path in the other direction.

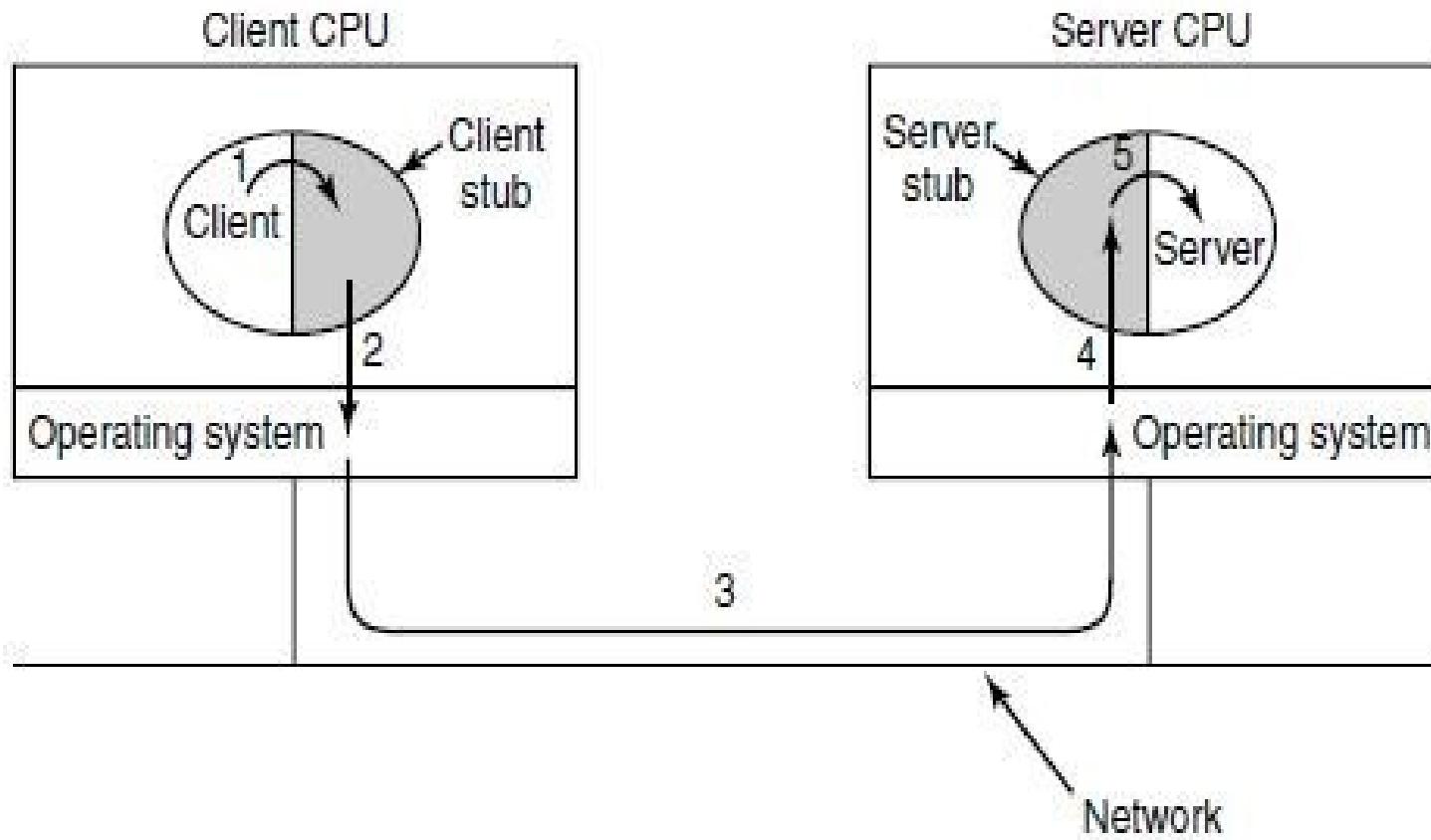


Figure 6-29. Steps in making a remote procedure call. The stubs are shaded.

## Problems with RPC:

- 1 With RPC, passing pointers is impossible because the client and server are in different address spaces.
- 2 It is essentially impossible for the client stub to marshal the parameters: it has no way of determining how large they are.
- 3 A third problem is that it is not always possible to deduce the types of parameters, not even from a formal specification or the code itself .(exa: printf)
- 4 A fourth problem relates to the use of global variables. Normally, the calling and called procedure can communicate by using global variables, in addition to communicating via parameters. But if the called procedure is moved to a remote machine, the code will fail because the global variables are no longer shared

# TCP

TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level. In brief, TCP is called a *connection-oriented, reliable transport protocol. It adds* connection-oriented and reliability features to the services of IP.

## *Topics discussed in this section:*

TCP Services

TCP Features

Segment

A TCP Connection

Flow Control

Error Control

# TCP Services

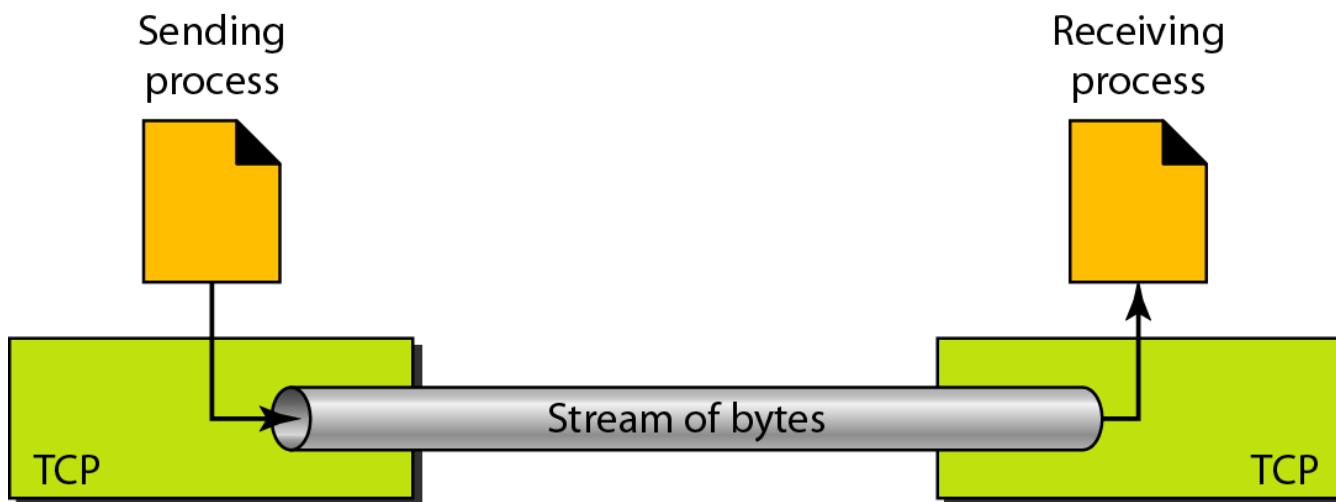
## ***1 Process-to-Process Communication***

TCP provides process-to-process communication using port numbers. Below Table lists some well-known port numbers used by TCP.

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

## **2 Stream Delivery Service**

TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their data across the Internet. This imaginary environment is showed in below Figure. The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them



**3 Sending and Receiving Buffers** Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction. One way to implement a buffer is to use a circular array of 1-byte locations as shown in Figure. For simplicity, we have shown two buffers of 20 bytes each. Normally the buffers are hundreds or thousands of bytes, depending on the implementation. We also show the buffers as the same size, which is not always the case.

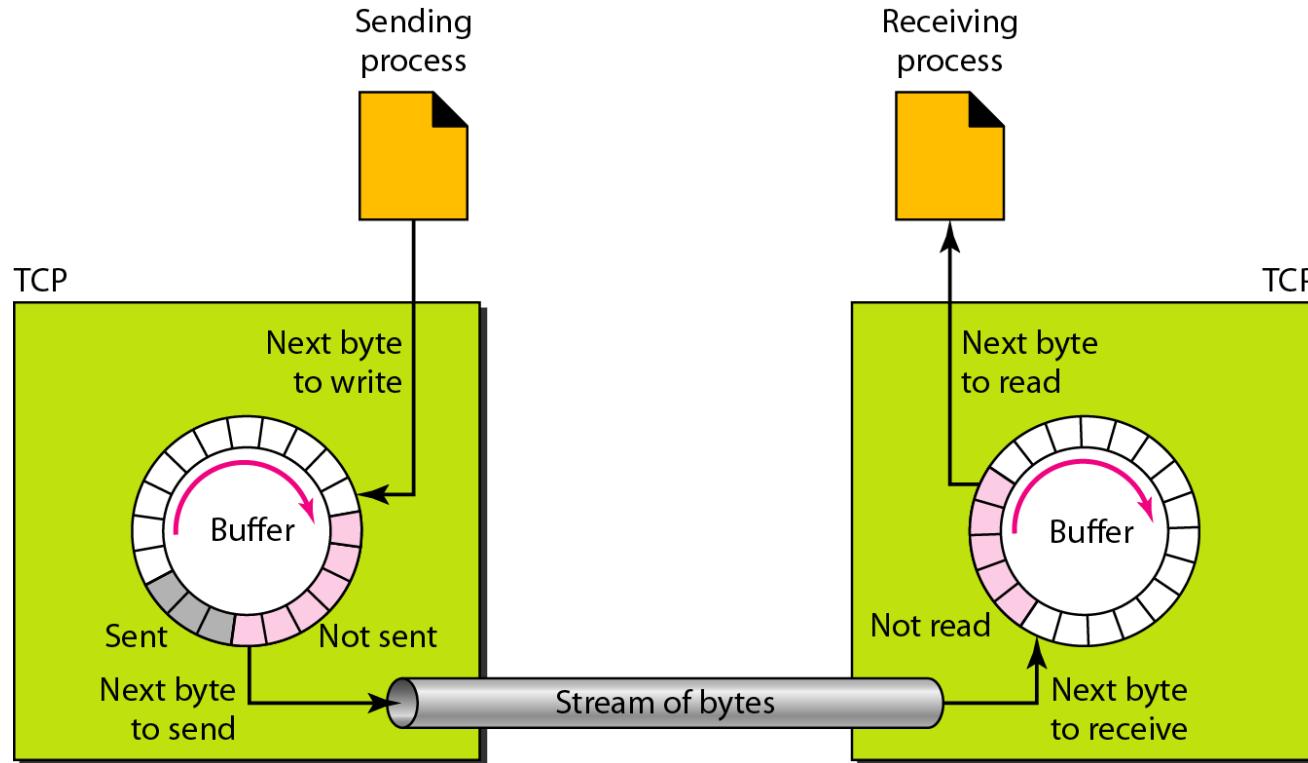


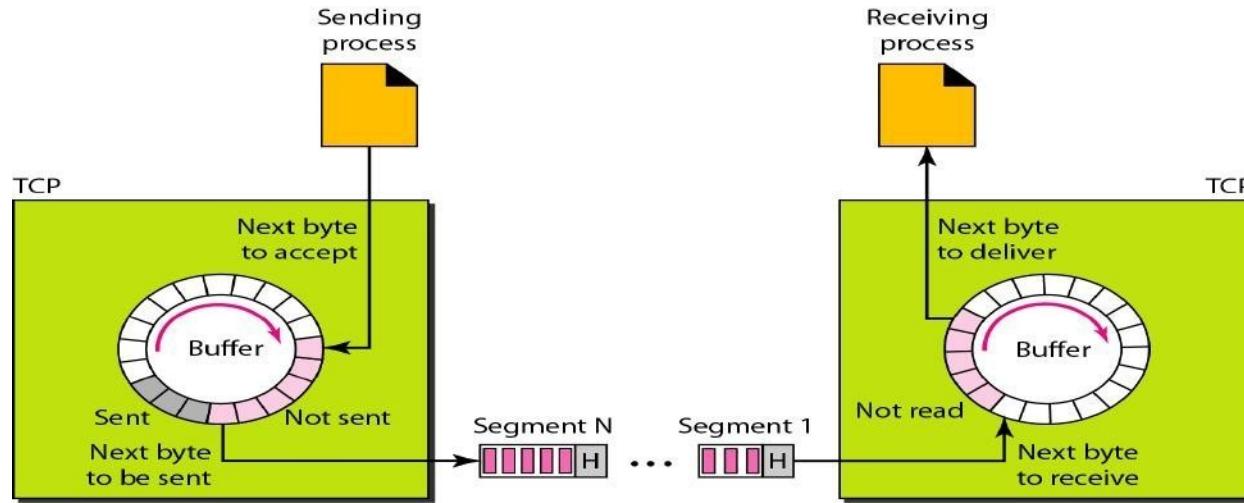
Figure shows the movement of the data in one direction. At the sending site, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer). The gray area holds bytes that have been sent but not yet acknowledged. TCP keeps these bytes in the buffer until it receives an acknowledgment. The colored area contains bytes to be sent by the sending TCP.

However, as we will see later in this chapter, TCP may be able to send only part of this colored section. This could be due to the slowness of the receiving process or perhaps to congestion in the network. Also note that after the bytes in the gray chambers are acknowledged, the chambers are recycled and available for use by the sending process.

This is why we show a circular buffer.

The operation of the buffer at the receiver site is simpler. The circular buffer is divided into two areas (shown as white and colored). The white area contains empty chambers to be filled by bytes received from the network. The colored sections contain received bytes that can be read by the receiving process. When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

## 4 TCP segments



At the transport layer, TCP groups a number of bytes together into a packet called a segment. TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission. The segments are encapsulated in IP datagrams and transmitted.

This entire operation is transparent to the receiving process. Later we will see that segments may be received out of order, lost, or corrupted and resent. All these are handled by TCP with the receiving process unaware of any activities. Above fig shows how segments are created from the bytes in the buffers

## **5 Full-Duplex Communication**

TCP offers full-duplex service, in which data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer, and segments move in both directions

## **6 Connection-Oriented Service**

TCP is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two TCPs establish a connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

## **7 Reliable Service**

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data. We will discuss this feature further in the section on error control.

# TCP Features

## 1 Numbering System

There are two fields called the sequence number and the acknowledgment number. These two fields refer to the byte number and not the segment number.

**Byte Number** The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number. For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056. We will see that byte numbering is used for flow and error control.

**Sequence Number** After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.

**Acknowledgment Number** The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

## **2 Flow Control**

TCP, provides *flow control*. *The receiver of the data controls the amount of data that are to be sent by the sender.* This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

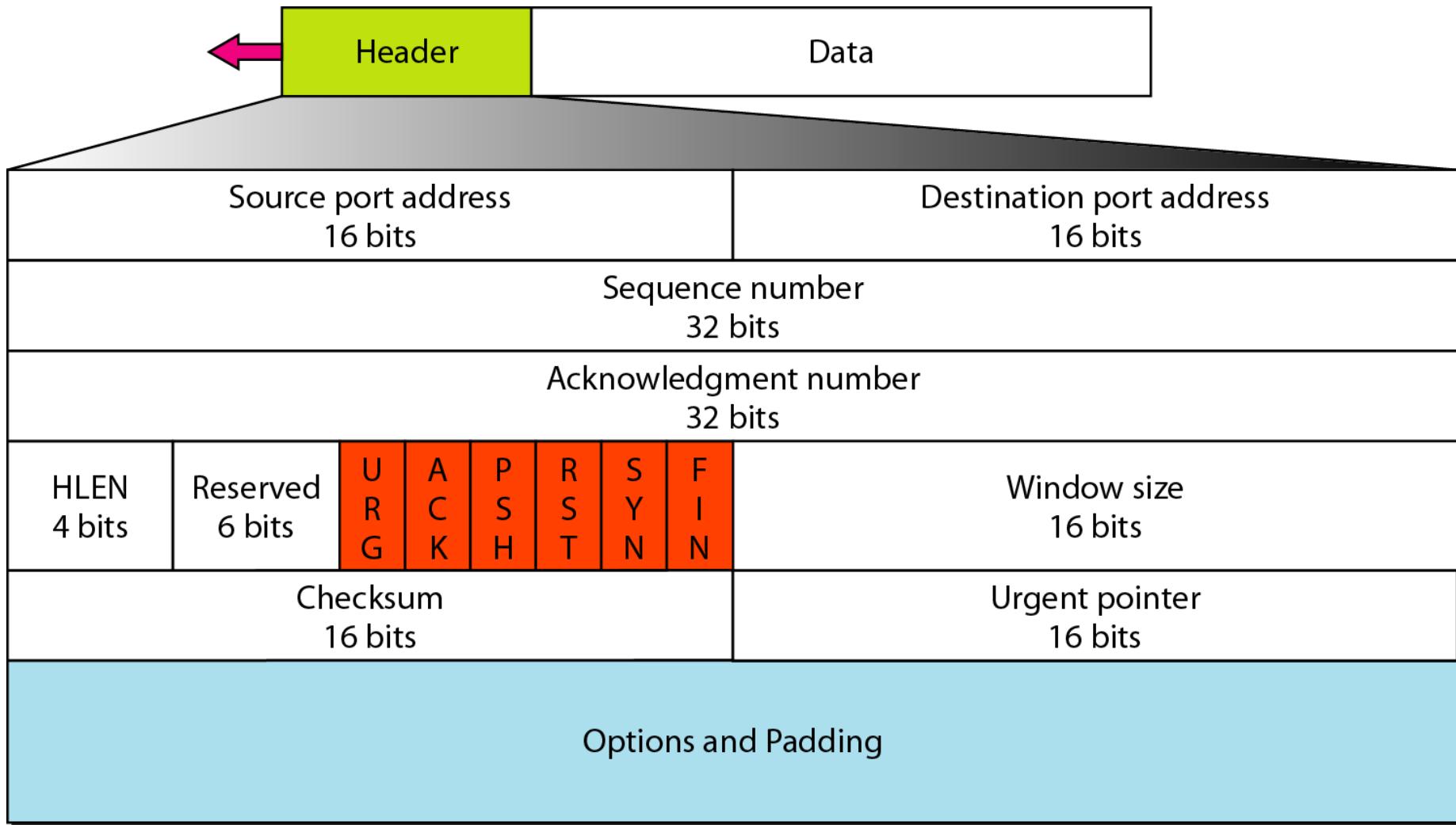
## **3 Error Control**

To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented, as we will see later.

## **4 Congestion Control**

TCP takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network

## TCP segment format



The segment consists of a 20- to 60-byte header.,.

**Source port address.** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

**Destination port address.** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

**Sequence number.** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

**Acknowledgment number.** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number  $x$  *from the other party*, it defines  $x + 1$  as the acknowledgment number. *Acknowledgment and data can be piggybacked together.*

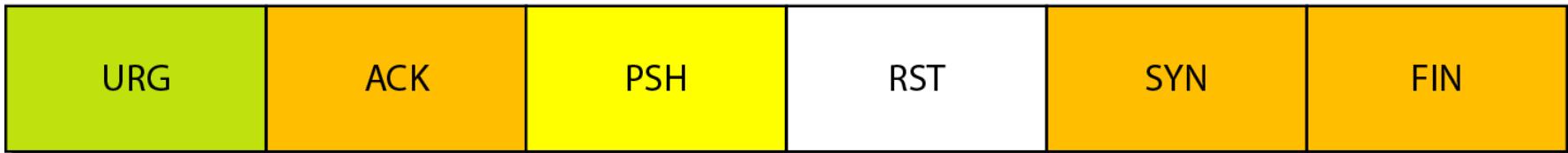
**Header length.** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 ( $5 \times 4 = 20$ ) and 15 ( $15 \times 4 = 60$ ).

**Reserved.** This is a 6-bit field reserved for future use.

**Control.** This field defines 6 different control bits or flags as shown in Figure. One or more of these bits can be set at a time.

URG: Urgent pointer is valid  
ACK: Acknowledgment is valid  
PSH: Request for push

RST: Reset the connection  
SYN: Synchronize sequence numbers  
FIN: Terminate the connection



These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.

**Window size.** This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

**Checksum.** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.

**Urgent pointer.** This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. This will be discussed later in this chapter.

**Options.** There can be up to 40 bytes of optional information in the TCP header. We will not discuss these options here; please refer to the reference list for more information.

## **A TCP Connection**

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All the segments belonging to a message are then sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.

In TCP, connection-oriented transmission requires three phases:

1. connection establishment,
2. data transfer,
3. connection termination.

# TCP connection establishment(3 way handshaking)

1 The client sends the first segment, a SYN segment, in which only the SYN flag is set.

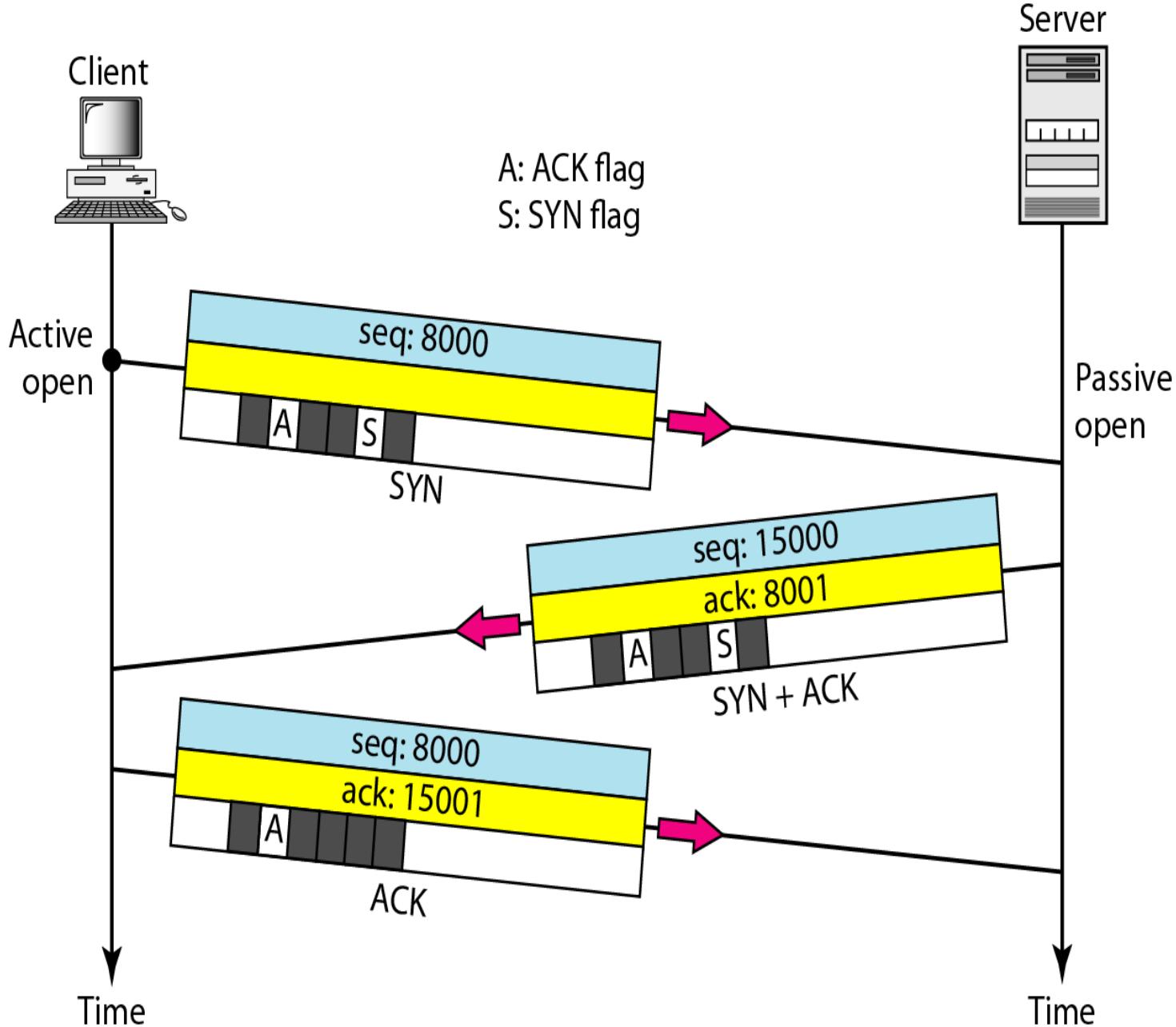
NOTE: A SYN segment cannot carry data, but it consumes one sequence number.

2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

NOTE: A SYN+ACK segment cannot carry data, but does consume one sequence number

3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

NOTE: An ACK segment, if carrying no data, consumes no sequence number



## SYN Flooding Attack

This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming from a different client by faking the source IP addresses in the datagram's.

The server, assuming that the clients are issuing an active open, allocates the necessary resources, such as creating communication tables and setting timers. The TCP server then sends the SYN +ACK segments to the fake clients, which are lost. During this time, however, a lot of resources are occupied without being used. If, during this short time, the number of SYN segments is large, the server eventually runs out of resources and may crash. This SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

### SOLUTIONS:

- 1 Some have imposed a limit on connection requests during a specified period of time.
- 2 Others filter out datagrams coming from unwanted source addresses.
- 3 One recent strategy is to postpone resource allocation until the entire connection is set up using what is called a cookie.

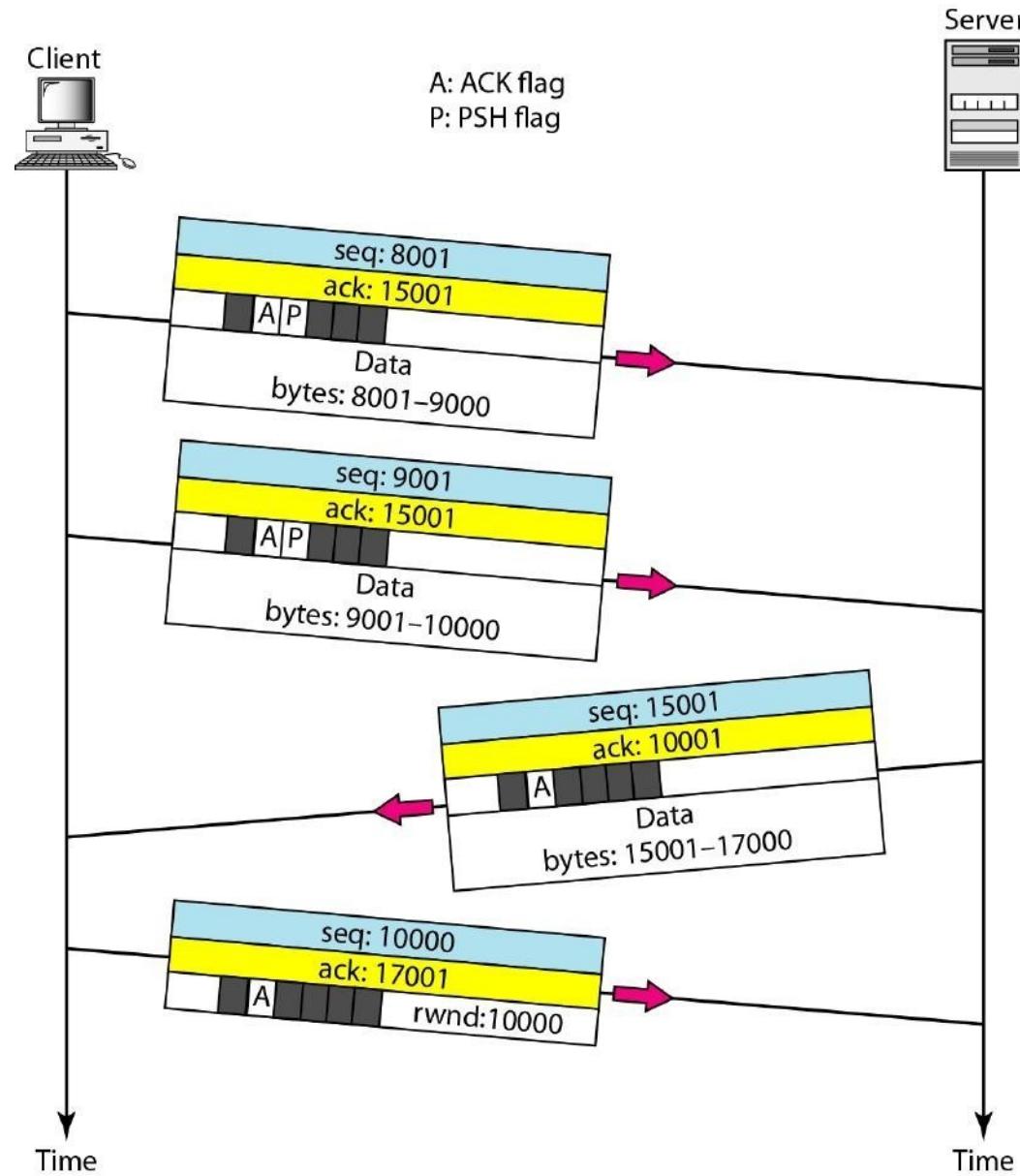
## Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments. Data traveling in the same direction as an acknowledgment are carried on the same segment. The acknowledgment is piggybacked with the data

In this example, after connection is established (not shown in the figure), the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.

Note the values of the sequence and acknowledgment numbers. The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.

## Data transfer



**PUSHING DATA:** Delayed transmission and delayed delivery of data may not be acceptable by the application program.

TCP can handle such a situation. The application program at the sending site can request a *push operation*. *This means that the sending TCP must not wait for the window to be filled.* It must create a segment and send it immediately. The sending TCP must also set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.

**Urgent Data :** TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream. However, sending application program wants a piece of data to be read out of order by the receiving application program.

**Connection Termination** (three-way handshaking and four-way handshaking with a half-close option.)

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set.

Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure. If it is only a control segment, it consumes only one sequence number.

NOTE: The FIN segment consumes one sequence number if it does not carry data.

2 The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN +ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

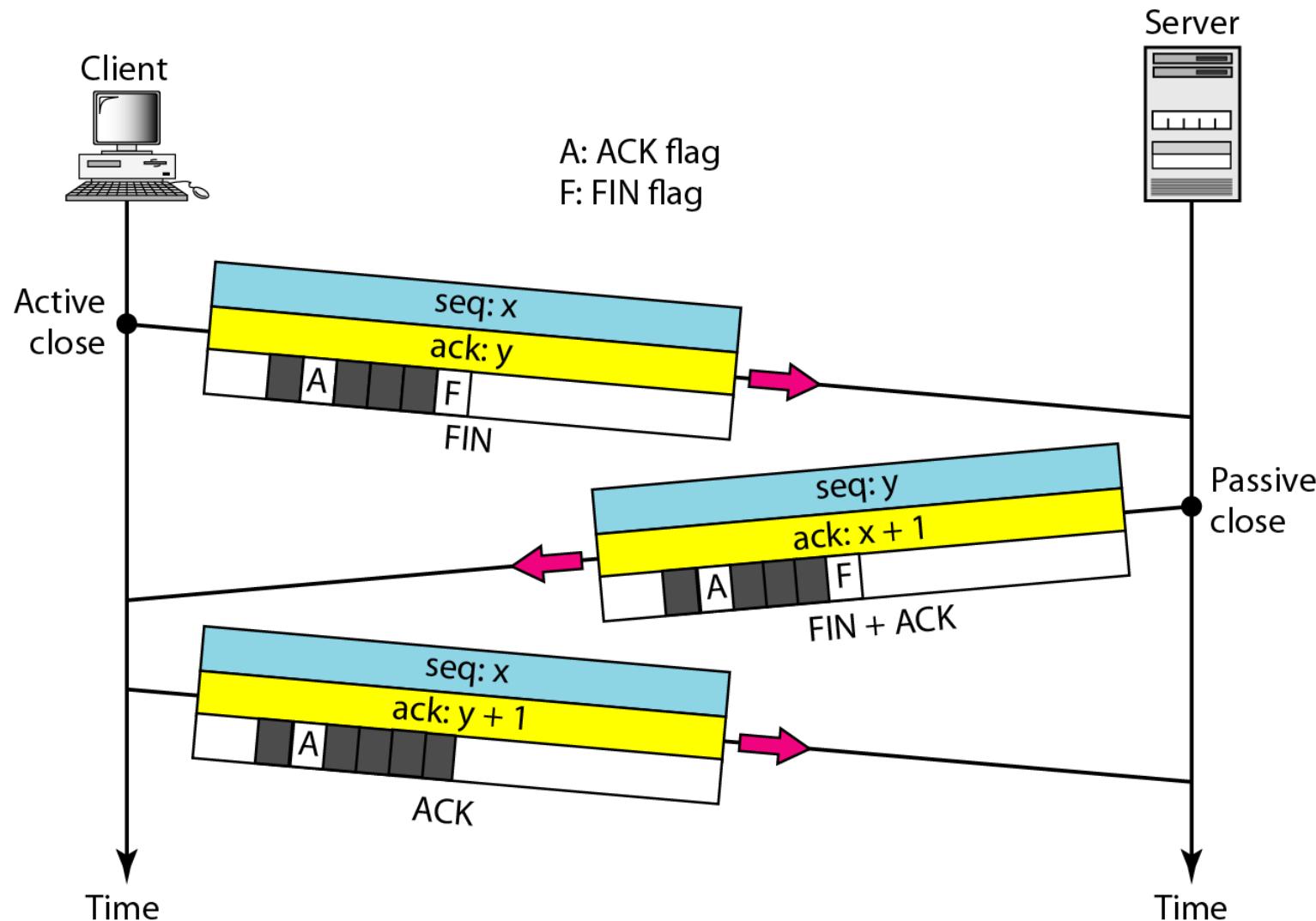
NOTE: The FIN +ACK segment consumes one sequence number if it does not carry data.

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

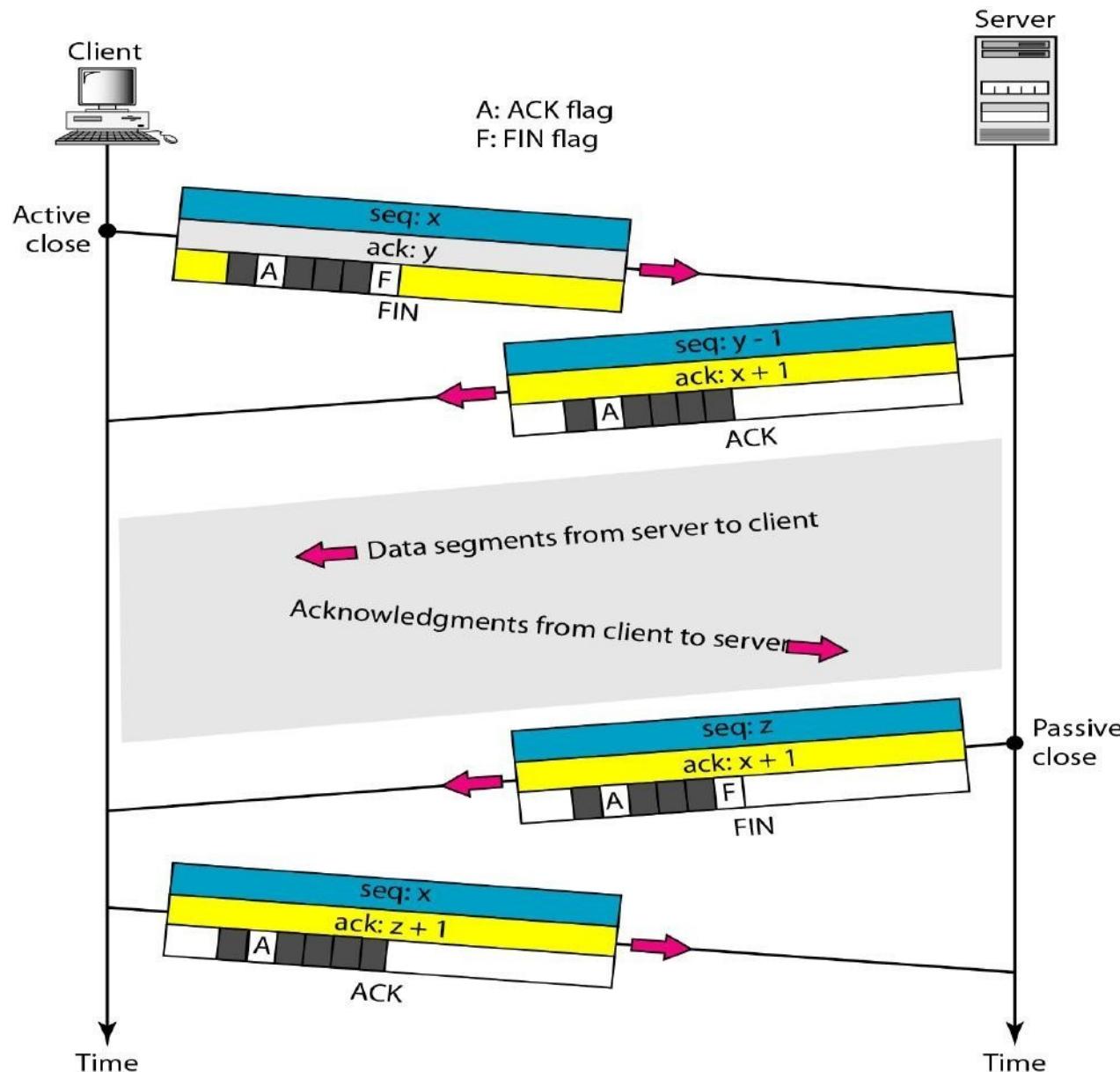
**Half-Close** In TCP, one end can stop sending data while still receiving data. This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin.

A good example is sorting. When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start. This means the client, after sending all the data, can close the connection in the outbound direction. However, the inbound direction must remain open to receive the sorted data. The server, after receiving the data, still needs time for sorting; its outbound direction must remain open

## Connection termination using three-way handshaking



## Figure 23.21 Half-close



## Flow Control or TCP Sliding Window

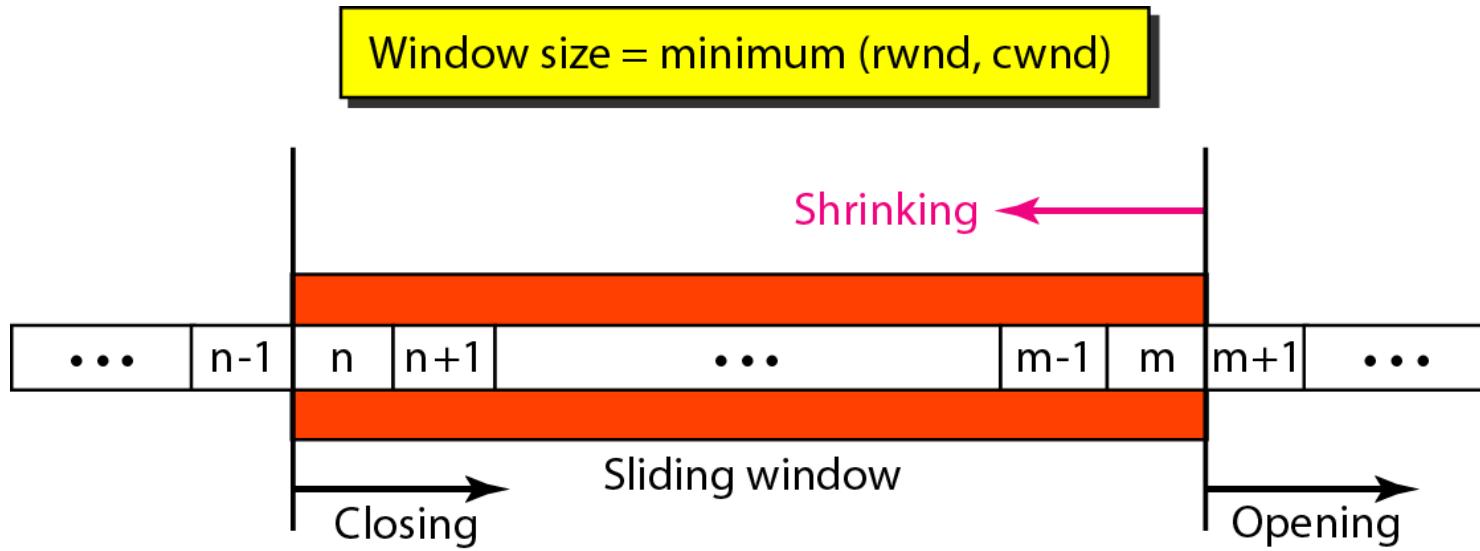
TCP uses a sliding window, to handle flow control. The sliding window protocol used by TCP, however, is something between the *Go-Back-N* and Selective Repeat sliding window.

The sliding window protocol in TCP looks like the *Go-Back-N* protocol because it does not use NAKs;  
it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.

There are two big differences between this sliding window and the one we used at the data link layer.

- 1 the sliding window of TCP is byte-oriented; the one we discussed in the data link layer is frame-oriented.
- 2 the TCP's sliding window is of variable size; the one we discussed in the data link layer was of fixed size

## *Sliding window*



The window is opened, closed, or shrunk. These three activities, as we will see, are in the control of the receiver (and depend on congestion in the network), not the sender.

The sender must obey the commands of the receiver in this matter.

Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending.

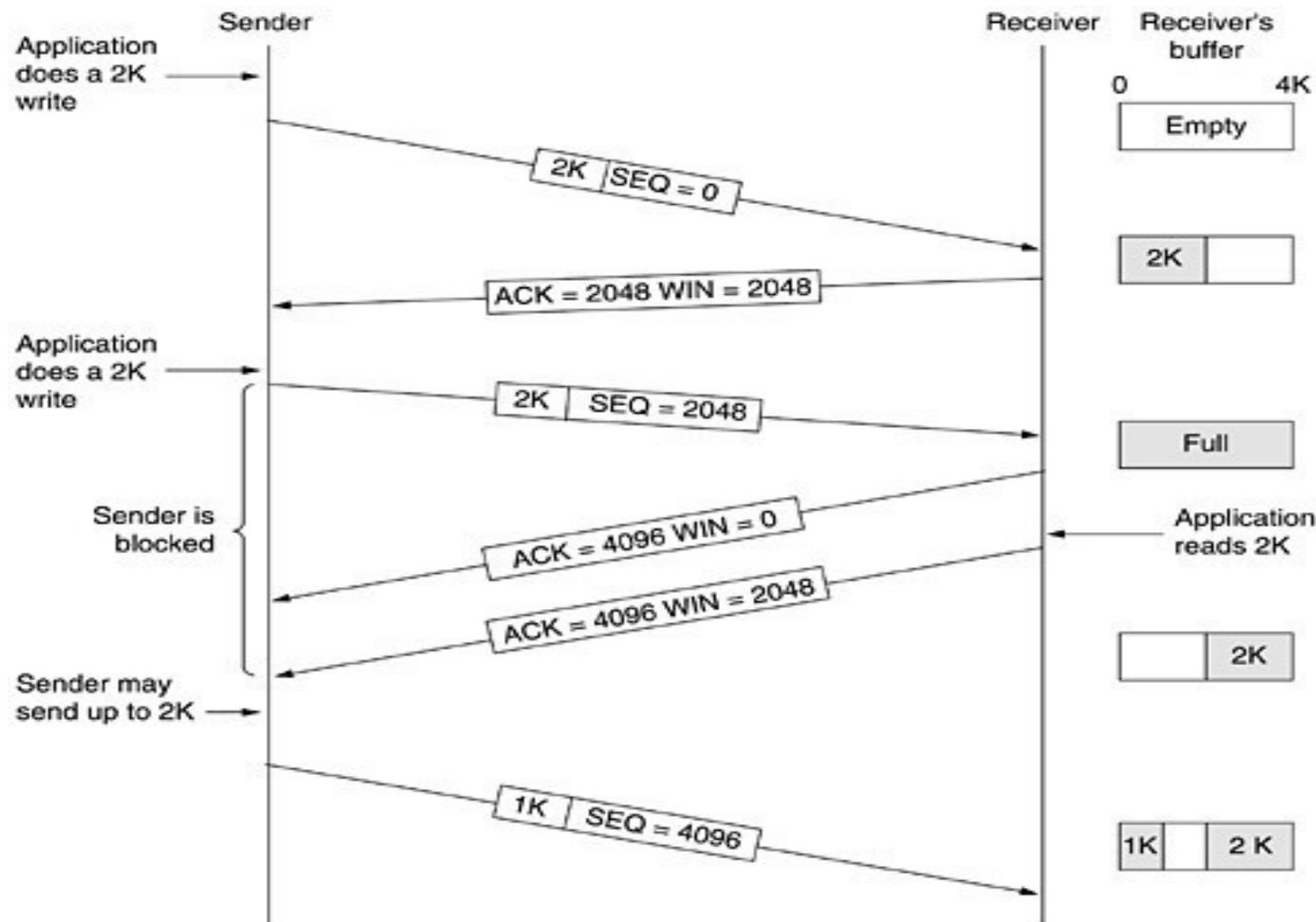
Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore.

Shrinking the window means moving the right wall to the left.

The size of the window at one end is determined by the lesser of two values: receiver window (rwnd) or congestion window (cwnd).

The receiver window is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded.

The congestion window is a value determined by the network to avoid congestion



## Window management in TCP

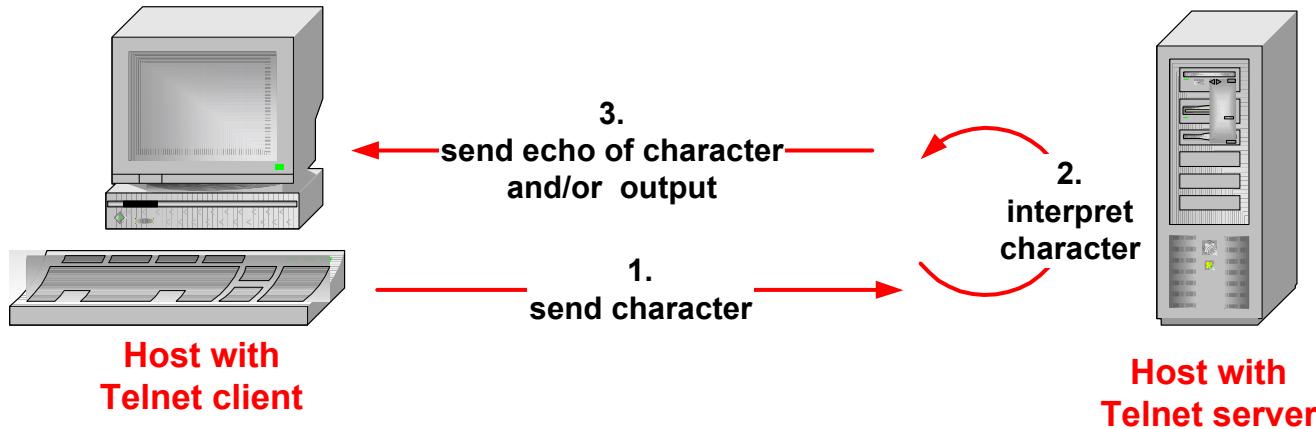
When the window is 0, the sender may not normally send segments, with two exceptions.

- 1) urgent data may be sent, for example, to allow the user to kill the process running on the remote machine.
- 2) the sender may send a 1-byte segment to force the receiver to reannounce the next byte expected and the window size. This packet is called a **window probe**.

The TCP standard explicitly provides this option to prevent deadlock if a window update ever gets lost.

Senders are not required to transmit data as soon as they come in from the application. Neither are receivers required to send acknowledgements as soon as possible.

For example, in Fig. when the first 2 KB of data came in, TCP, knowing that it had a 4-KB window, would have been completely correct in just buffering the data until another 2 KB came in, to be able to transmit a segment with a 4-KB payload. This freedom can be used to improve performance



Remote terminal applications (e.g., Telnet) send characters to a server. The server interprets the character and sends the output at the server to the client.

For each character typed, you see three packets:

**Client → Server:** Send typed character

**Server → Client:** Echo of character (or user output) and acknowledgement for first packet

**Client → Server:** Acknowledgement for second packet

# Delayed Acknowledgement

- TCP delays transmission of ACKs for up to 500ms
- Avoid to send ACK packets that do not carry data.
  - The hope is that, within the delay, the receiver will have data ready to be sent to the receiver. Then, the ACK can be piggybacked with a data segment

## Exceptions:

- ACK should be sent for every full sized segment
- Delayed ACK is not used when packets arrive out of order

Although delayed acknowledgements reduce the load placed on the network by the receiver, a sender that sends multiple short packets (e.g., 41-byte packets containing 1 byte of data) is still operating inefficiently. A way to reduce this usage is known as **Nagle's algorithm (Nagle, 1984)**.

## Nagel's Rule

Send one byte and buffer all subsequent bytes until acknowledgement is received. Then send all buffered bytes in a single TCP segment and start buffering again until the sent segment is acknowledged.

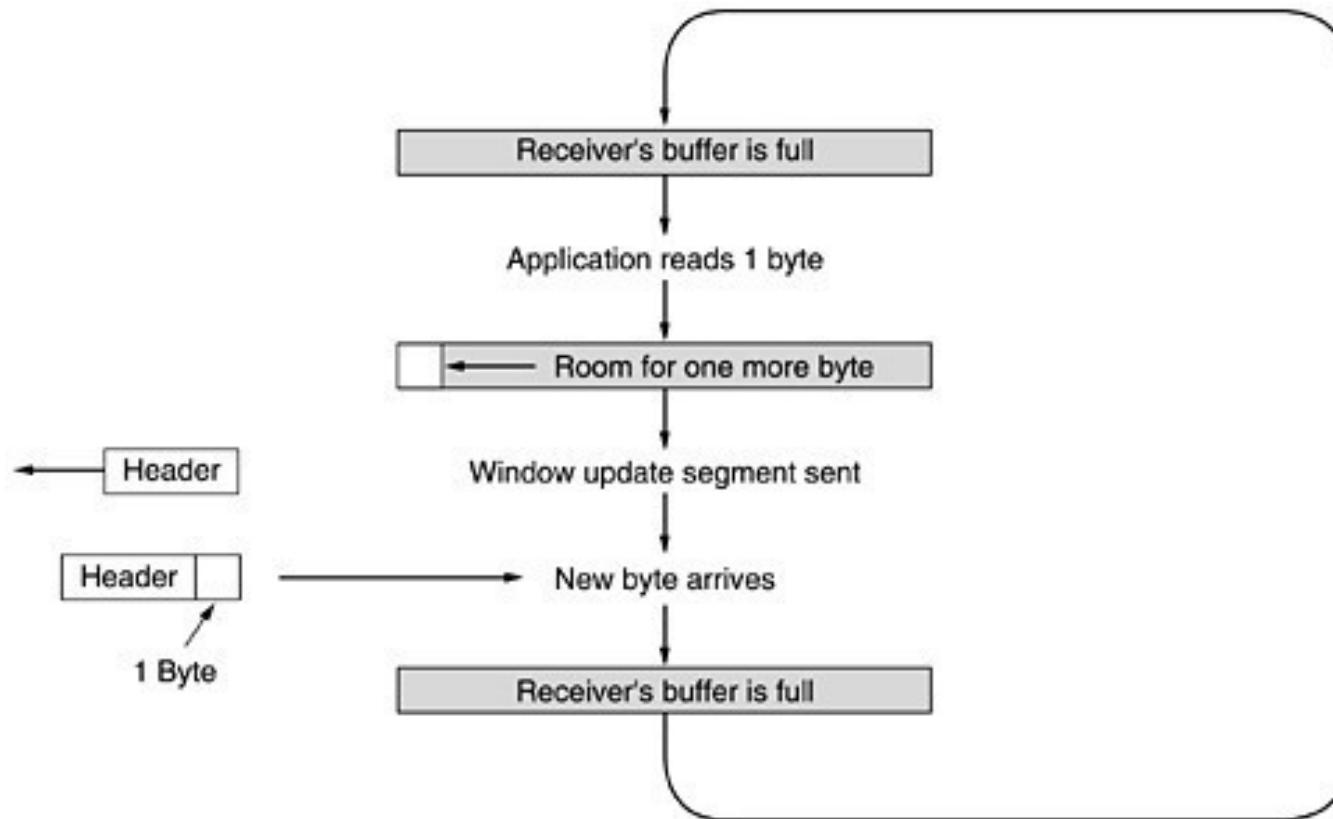
Nagle's algorithm will put the many pieces in one segment, greatly reducing the bandwidth used

Nagle's algorithm is widely used by TCP implementations, but there are times when it is better to disable it. In particular, in interactive games that are run over the Internet.

A more subtle problem is that Nagle's algorithm can sometimes interact with delayed acknowledgements to cause a temporary deadlock: the receiver waits for data on which to piggyback an acknowledgement, and the sender waits on the acknowledgement to send more data.

Because of these problems, Nagle's algorithm can be disabled (which is called the *TCP NODELAY option*).

Another problem that can degrade TCP performance is the **silly window syndrome** (Clark, 1982).



Clark's solution is to prevent the receiver from sending a window update for 1 byte. Instead, it is forced to wait until it has a decent amount of space available and advertise that instead. Specifically, the receiver should not send a window update until it can handle the maximum segment size it advertised when the connection was established or until its buffer is half empty, whichever is smaller.

Furthermore, the sender can also help by not sending tiny segments. Instead, it should wait until it can send a full segment, or at least one containing half of the receiver's buffer size.

The goal is for the sender not to send small segments and the receiver not to ask for them. (Nagel + Clark). Both are used to improve TCP performance

The receiver will buffer the data until it can be passed up to the application in order (handling out of order segments)

## **Cumulative acknowledgements**

## Error Control

TCP is a reliable transport layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.

TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools: **checksum, acknowledgment, and time-out.**

### Checksum

Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment

**Figure 23.11** *Checksum calculation of a simple UDP user datagram*

153.18.8.105		
171.2.14.10		
All 0s	17	15
1087		13
15		All 0s
T	E	S
I	N	G
All 0s		

10011001 00010010	→	153.18
00001000 01101001	→	8.105
10101011 00000010	→	171.2
00001110 00001010	→	14.10
00000000 00010001	→	0 and 17
00000000 00001111	→	15
00000100 00111111	→	1087
00000000 00001101	→	13
00000000 00001111	→	15
00000000 00000000	→	0 (checksum)
01010100 01000101	→	T and E
01010011 01010100	→	S and T
01001001 01001110	→	I and N
01000111 00000000	→	G and 0 (padding)
<hr/>		
10010110 11101011	→	Sum
01101001 00010100	→	Checksum

## **Acknowledgment**

TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged. ACK segments do not consume sequence numbers and are not acknowledged.

## **Retransmission**

The heart of the error control mechanism is the retransmission of segments. When a segment is corrupted, lost, or delayed, it is retransmitted.

In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.

Retransmission After RTO (retransmission time out)

Retransmission After Three Duplicate ACK Segments (also called fast retransmission)

## **Out-of-Order Segments**

Data may arrive out of order and be temporarily stored by the receiving TCP, but yet guarantees that no out-of-order segment is delivered to the process

# TCP Congestion Control

When the load offered to any network is more than it can handle, congestion builds up.

The network layer detects congestion when queues grow large at routers and tries to manage it, if only by dropping packets. It is up to the transport layer to receive congestion feedback from the network layer and slow down the rate of traffic that it is sending into the network.

For Congestion control, transport protocol uses an AIMD (Additive Increase Multiplicative Decrease) control law.

TCP congestion control is based on implementing this approach using a window called **congestion window**. TCP adjusts the size of the window according to the AIMD rule.

The window size at the sender is set as follows:

**Send Window = MIN (flow control window, congestion window)**

where

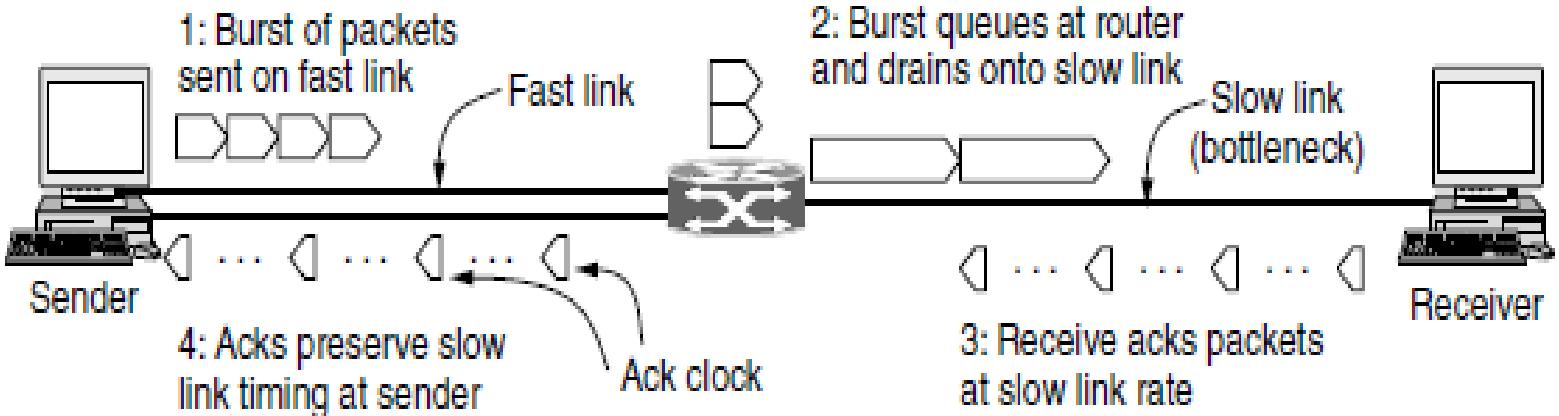
**flow control window** is advertised by the receiver (rwnd)

**congestion window** is adjusted based on feedback from the

Modern congestion control was added to TCP largely through the efforts of Van Jacobson (1988). It is a fascinating story. Starting in 1986, the growing popularity of the early Internet led to the first occurrence of what became known as a **congestion collapse**, a prolonged period during which good put dropped suddenly (i.e., by more than a factor of 100) due to congestion in the network. Jacobson (and many others) set out to understand what was happening and remedy the situation.

To start, he observed that packet loss is a suitable signal of congestion. This signal comes a little late (as the network is already congested) but it is quite dependable

At the beginning how sender knows at what speed receiver can receive the packets?

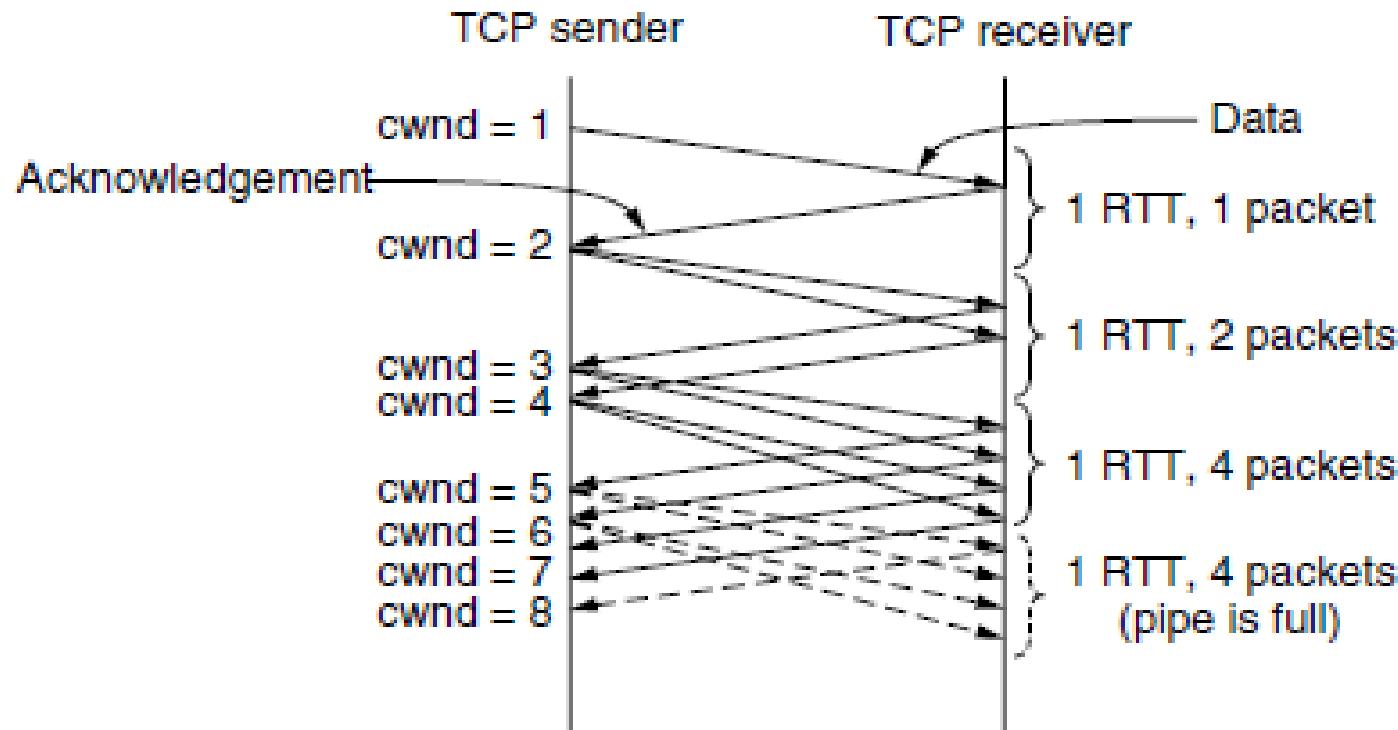


The key observation is this: the acknowledgements return to the sender at about the rate that packets can be sent over the slowest link in the path. This is precisely the rate that the sender wants to use. If it injects new packets into the network at this rate, they will be sent as fast as the slow link permits, but they will not queue up and congest any router along the path. This timing is known as an **ack clock**. It is an essential part of TCP. By using an ack clock, TCP smoothes out traffic and avoids unnecessary queues at routers. This is first consideration

A second consideration is that the AIMD rule will take a very long time to reach a good operating point on fast networks if the congestion window is started from a small size

Instead, the solution Jacobson chose to handle both of these considerations is a mix of linear and multiplicative increase.

## **SLOW-START**



# TCP Congestion Control

## Slow Start

- Additive Increase / Multiplicative Decrease is only suitable for source, that is operating close to the available capacity of the network, but it takes too long to ramp up a connection when it is starting from scratch.
- slow start, that is used to increase the congestion window rapidly from a cold start.
- Slow start effectively **increases the congestion window exponentially**, rather than linearly.
  - the source starts out by setting CongestionWindow to one packet.
  - When the ACK for this packet arrives, TCP adds 1 to CongestionWindow and then sends two packets.
  - Upon receiving the corresponding two ACKs, TCP increments CongestionWindow by 2—one for each ACK—and next sends four packets.
  - The end result is that TCP effectively doubles the number of packets it has in transit every RTT.

Whenever a packet loss is detected, for example, by a timeout, the slow start threshold is set to be half of the congestion window and the entire process is restarted.

Congestion avoidance phase is started if cwnd has reached the slow start threshold value

Whenever the slow start threshold is crossed, TCP switches from slow start to additive increase. In this mode, the congestion window is increased by one segment every round-trip time.

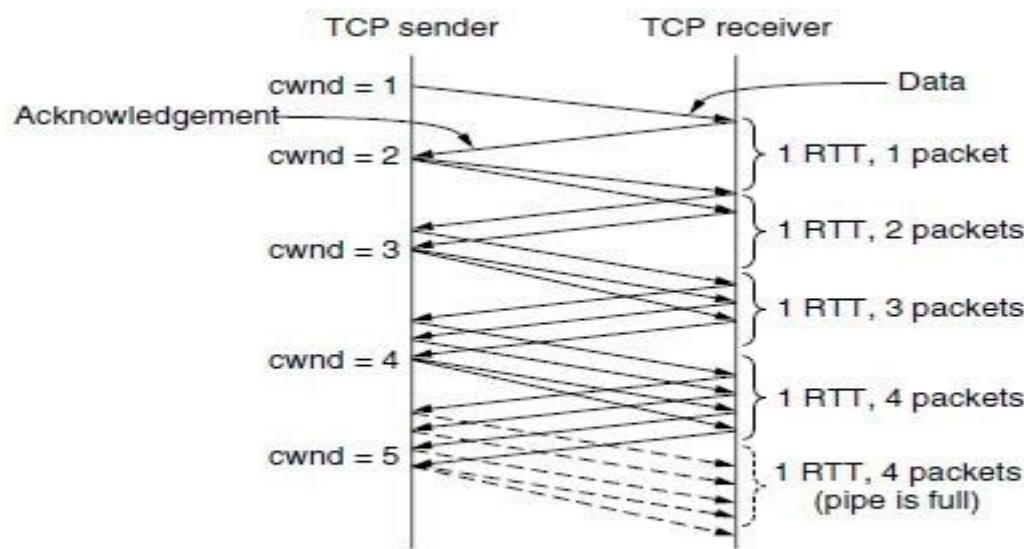


Figure 6-45. Additive increase from an initial congestion window of one segment.

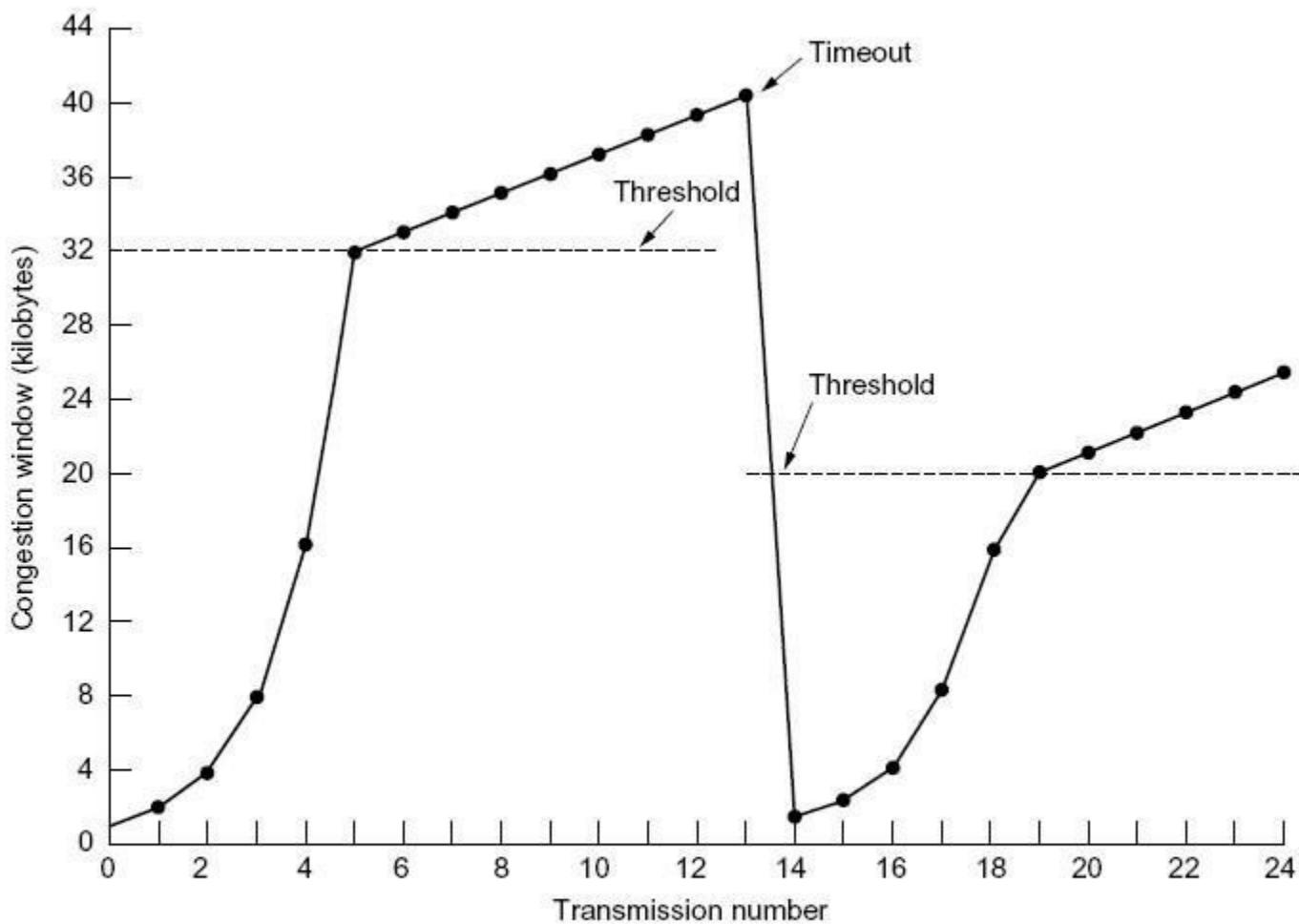


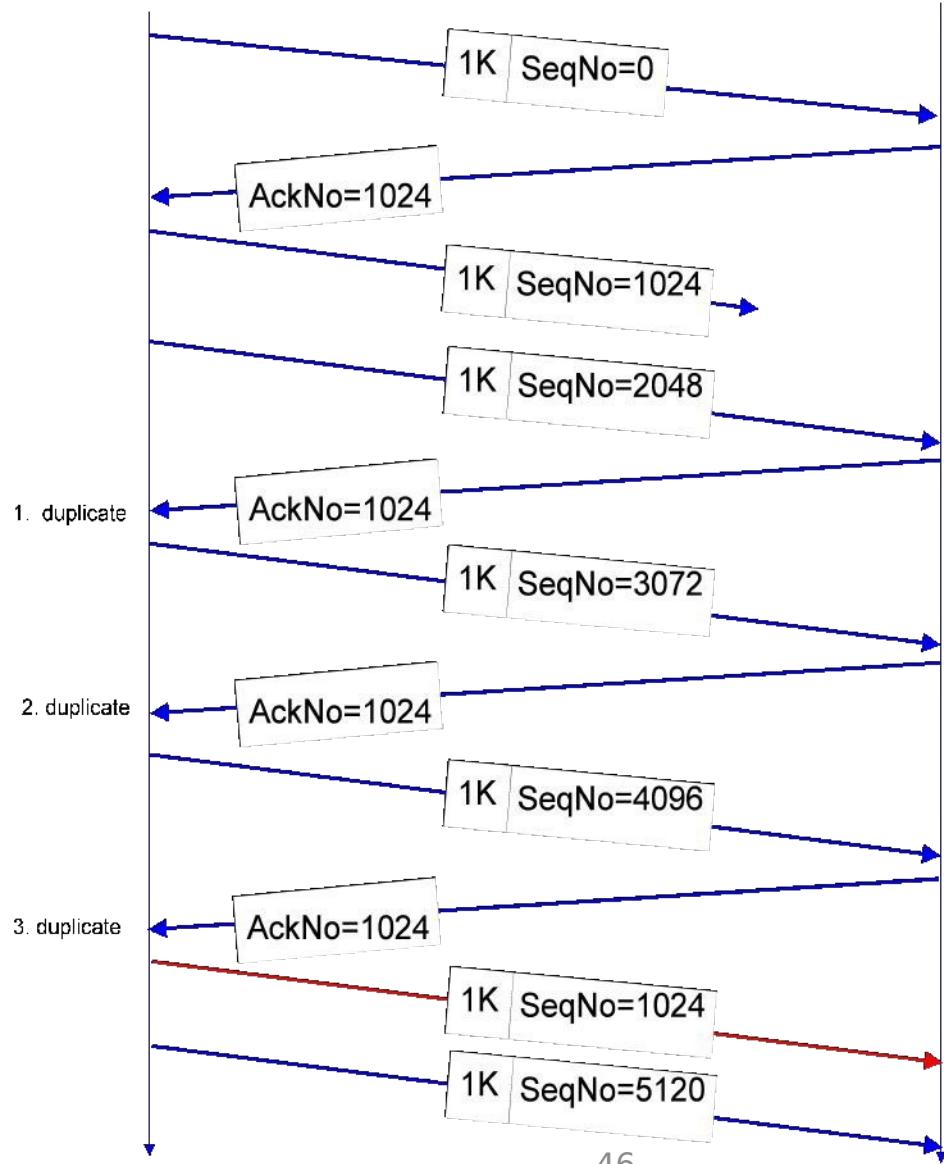
Fig. 6-37. An example of the Internet congestion algorithm.

# Responses to Congestion

- So, TCP assumes there is congestion if it detects a packet loss
- A TCP sender can detect lost packets via:
  - Timeout of a retransmission timer
  - Receipt of a duplicate ACK
- TCP interprets a Timeout as a binary congestion signal. When a timeout occurs, the sender performs:
  - cwnd is reset to one:  
 $cwnd = 1$
  - ssthresh is set to half the current size of the congestion window:  
 $ssthresh = cwnd / 2$
  - and slow-start is entered

# Fast Retransmit

- If three or more duplicate ACKs are received in a row, the TCP sender believes that a segment has been lost.
- Then TCP performs a retransmission of what seems to be the missing segment, without waiting for a timeout to happen.
- Enter slow start:  
 $\text{ssthresh} = \text{cwnd}/2$   
 $\text{cwnd} = 1$



# Flavors of TCP Congestion Control

- **TCP Tahoe** (1988)
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
- **TCP Reno** (1990) (TCP Tahoe+FR)
  - Fast Recovery
- **New Reno** (1996)
- **SACK** (1996) (**SACK (Selective ACKnowledgements)**)
- **RED** (Floyd and Jacobson 1993)

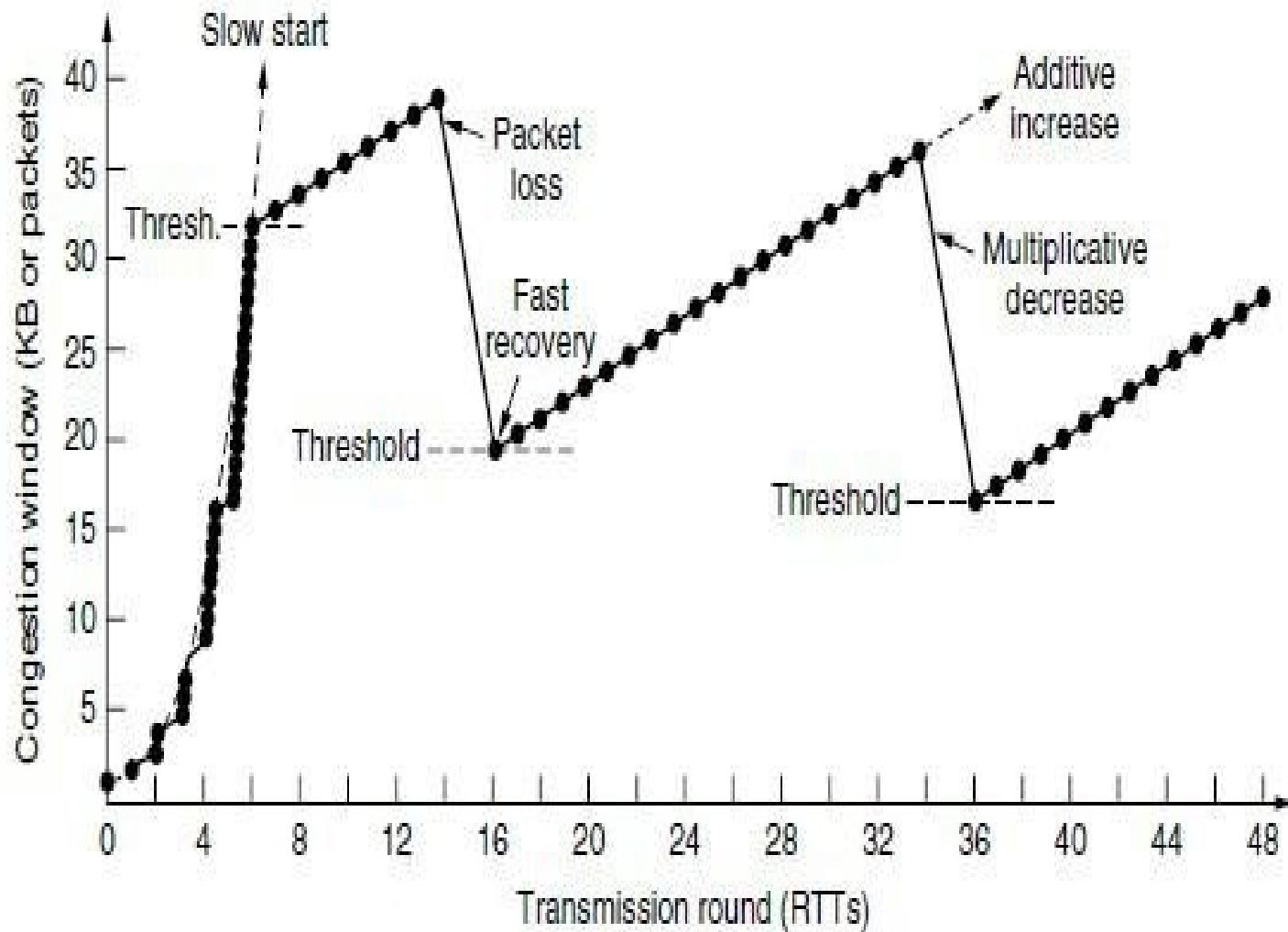


Figure 6-47. Fast recovery and the sawtooth pattern of TCP Reno.

The use of ECN (Explicit Congestion Notification) in addition to packet loss as a congestion signal. ECN is an IP layer mechanism to notify hosts of congestion.

The sender tells the receiver that it has heard the signal by using the CWR (*Congestion Window Reduced*) flag.

# USER DATAGRAM PROTOCOL (UDP)

*The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.*

## Topics discussed in this section:

Well-Known Ports for UDP

User Datagram

Checksum

UDP Operation

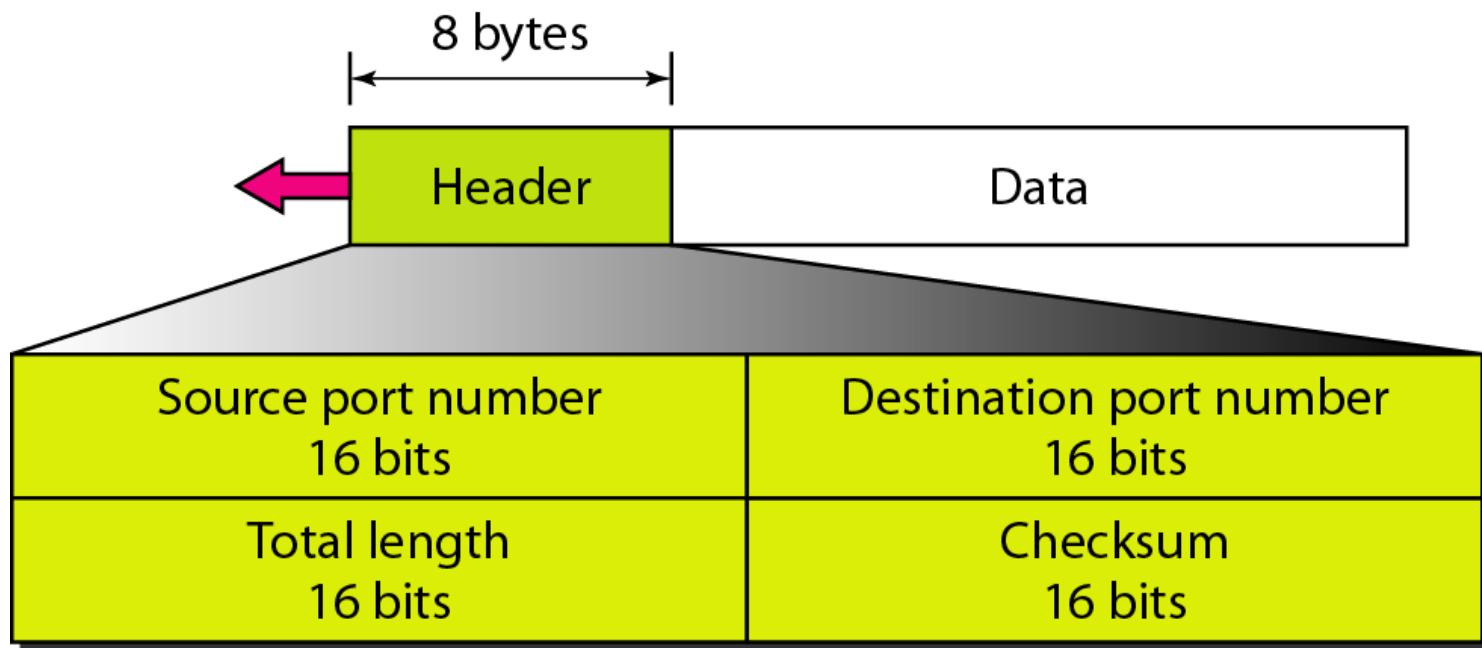
Use of UDP

**Table 23.1** *Well-known ports used with UDP*

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Figure 23.9 *User datagram format*

---



## **Checksum** (OPTIONAL, IF NOT USED SET ALL 1'S DEFAULT)

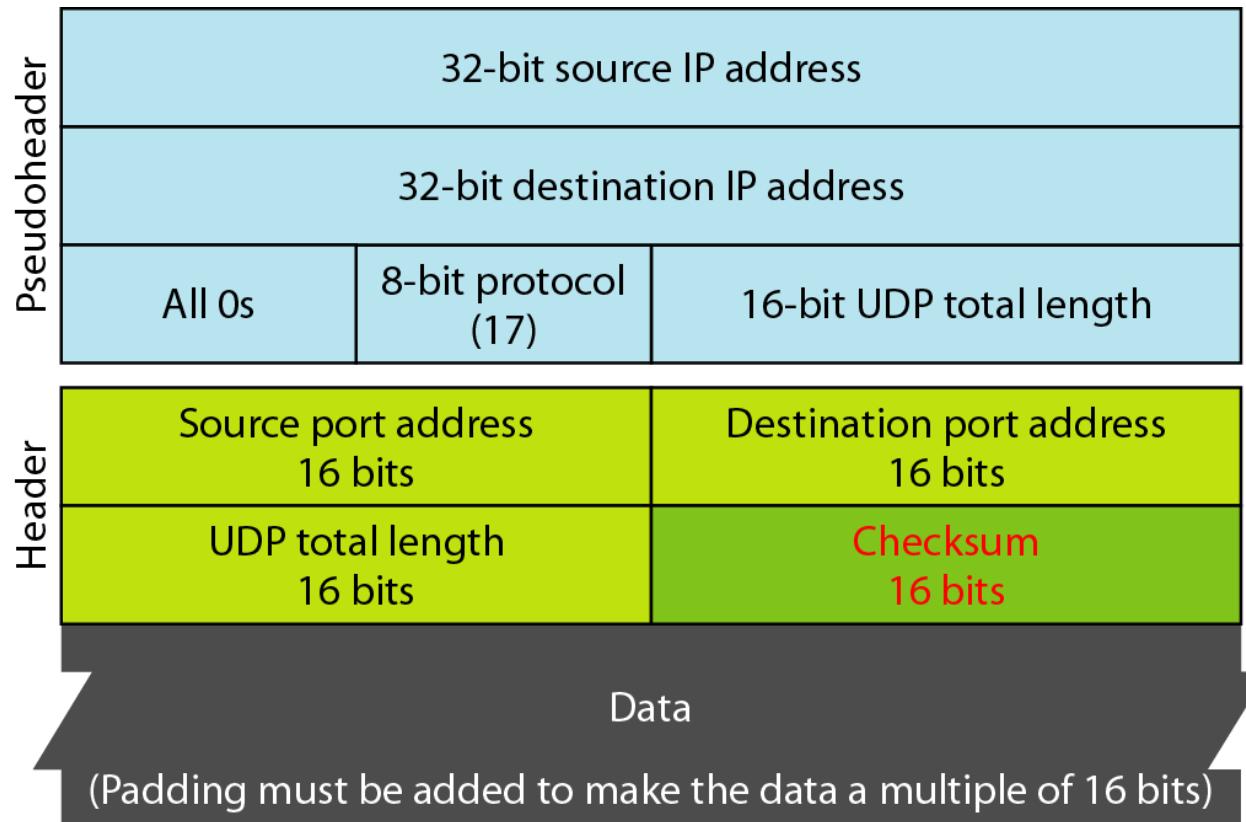
The UDP checksum calculation is different from the one for IP and ICMP. Here the checksum includes three sections: a pseudo header, the UDP header and the data coming from the application layer.

The pseudo header is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with Os

If the checksum does not include the pseudo header, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.

The protocol field is added to ensure that the packet belongs to UDP, and not to other transport-layer protocols.

**Figure 23.10 Pseudoheader for checksum calculation**



# **UDP Operation**

## **Connectionless Services**

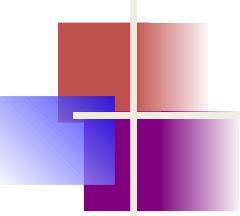
UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program. The user datagrams are not numbered. Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path.

## **Flow and Error Control**

UDP is a very simple, unreliable transport protocol. There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages. There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded. The lack of flow control and error control

## **Encapsulation and Decapsulation**

To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.



## *Example 23.2*

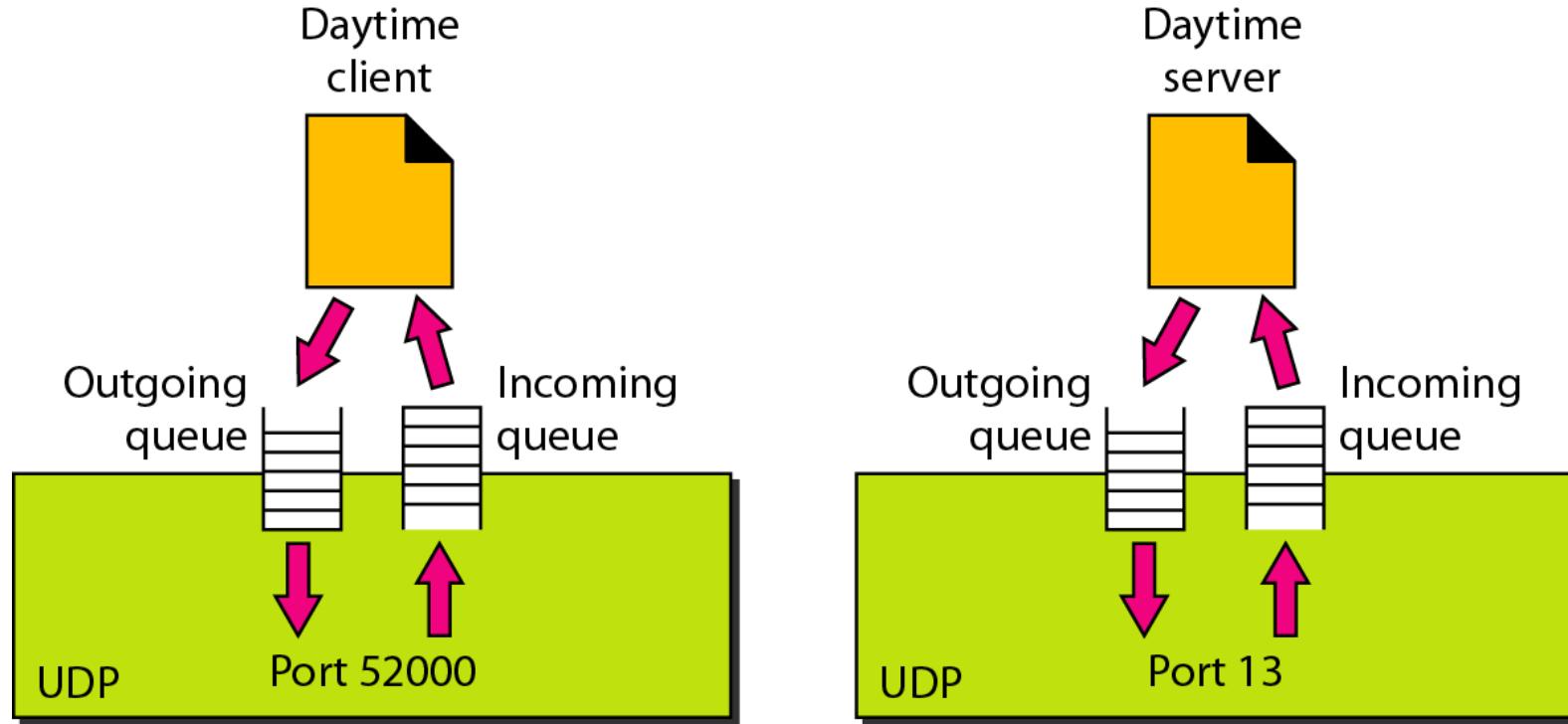
*Figure 23.11 shows the checksum calculation for a very small user datagram with only 7 bytes of data. Because the number of bytes of data is odd, padding is added for checksum calculation. The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.*

**Figure 23.11** *Checksum calculation of a simple UDP user datagram*

153.18.8.105		
171.2.14.10		
All 0s	17	15
1087		13
15		All 0s
T	E	S
I	N	G
All 0s		

10011001 00010010	→	153.18
00001000 01101001	→	8.105
10101011 00000010	→	171.2
00001110 00001010	→	14.10
00000000 00010001	→	0 and 17
00000000 00001111	→	15
00000100 00111111	→	1087
00000000 00001101	→	13
00000000 00001111	→	15
00000000 00000000	→	0 (checksum)
01010100 01000101	→	T and E
01010011 01010100	→	S and T
01001001 01001110	→	I and N
01000111 00000000	→	G and 0 (padding)
<hr/>		
10010110 11101011	→	Sum
01101001 00010100	→	Checksum

Figure 23.12 *Queues in UDP*



# Remote Procedure Call

The key work was done by Birrell and Nelson (1984). In a nutshell, what Birrell and Nelson suggested was allowing programs to call procedures located on remote hosts. When 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)**. Traditionally, the calling procedure is known as the client and the called procedure is known as the server, and we will use those names here too.

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**. These procedures hide the fact that the procedure call from the client to the server is not local

Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.

Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.

Step 3 is the operating system sending the message from the client machine to the server machine.

Step 4 is the operating system passing the incoming packet to the server stub.

Finally, step 5 is the server stub calling the server procedure with the unmarshaled parameters.

The reply traces the same path in the other direction.

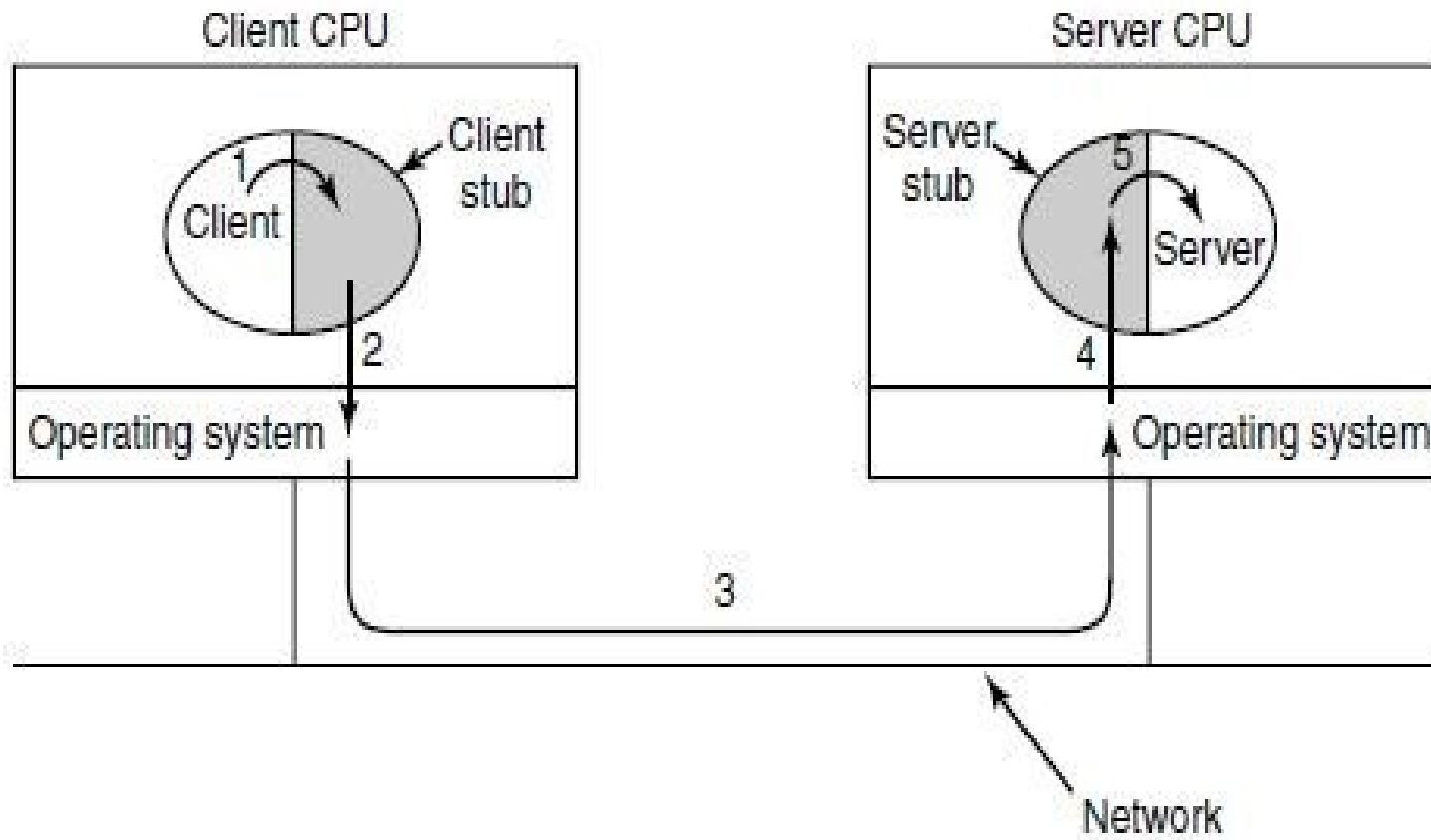


Figure 6-29. Steps in making a remote procedure call. The stubs are shaded.

## Problems with RPC:

- 1 With RPC, passing pointers is impossible because the client and server are in different address spaces.
- 2 It is essentially impossible for the client stub to marshal the parameters: it has no way of determining how large they are.
- 3 A third problem is that it is not always possible to deduce the types of parameters, not even from a formal specification or the code itself .(exa: printf)
- 4 A fourth problem relates to the use of global variables. Normally, the calling and called procedure can communicate by using global variables, in addition to communicating via parameters. But if the called procedure is moved to a remote machine, the code will fail because the global variables are no longer shared

## Real-Time Transport Protocols

Client-server RPC is one area in which UDP is widely used.

Another one is for real-time multimedia applications.

- Internet radio,
- Internet telephony,
- music-on-demand,
- videoconferencing,
- video-on-demand,

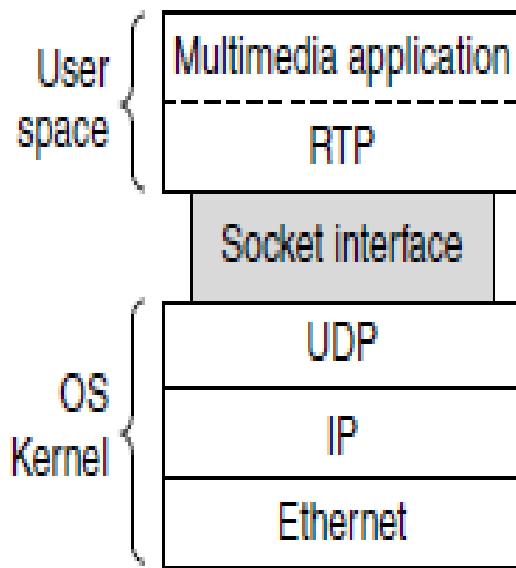
and other multimedia applications became more commonplace, people have discovered that each application was reinventing more or less the same real-time transport protocol.

It gradually became clear that having a generic real-time transport protocol for multiple applications would be a good idea.

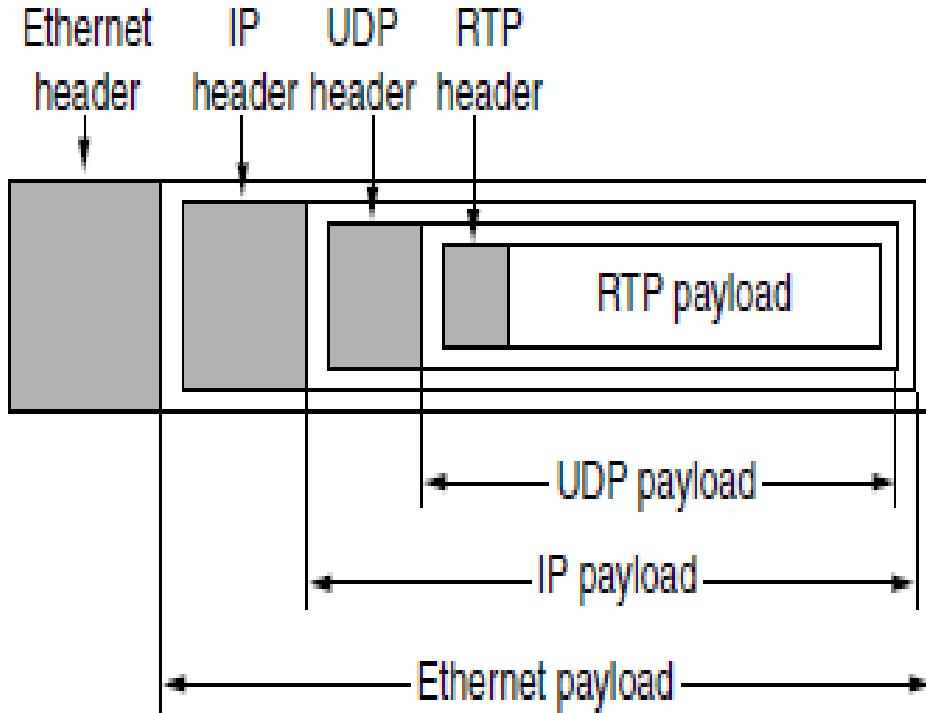
Thus was **RTP (Real-time Transport Protocol)** born. It is described in RFC 3550 and is now in widespread use for multimedia applications. We will describe two aspects of real-time transport.

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..



(a)



(b)

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

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. These are fed into the RTP library, which is in user space along with the application. This library multiplexes the streams and encodes them in RTP packets, which it stuffs into a socket.

On the operating system side of the socket, UDP packets are generated to wrap the RTP packets and handed to IP for transmission over a link such as Ethernet.

The reverse process happens at the receiver. The multimedia application eventually receives multimedia data from the RTP library. It is responsible for playing out the media. The protocol stack for this situation is shown in Fig. 6-30(a). The packet nesting is shown in Fig. 6-30(b).

## 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. In particular, there are no special guarantees about delivery, and packets may be lost, delayed, corrupted, etc.

The RTP format contains several features.

Each packet sent in an RTP stream is given a number one higher than its predecessor. This numbering allows the destination to determine if any packets are missing.

RTP has no acknowledgements, and no mechanism to request retransmissions.

Each RTP payload may contain multiple samples, and they may be coded any way that the application wants. To allow for interworking, RTP defines several profiles (e.g., a single audio stream), and for each profile, multiple encoding formats may be allowed

Another facility many real-time applications need is time stamping. Not only does time stamping reduce the effects of variation in network delay, but it

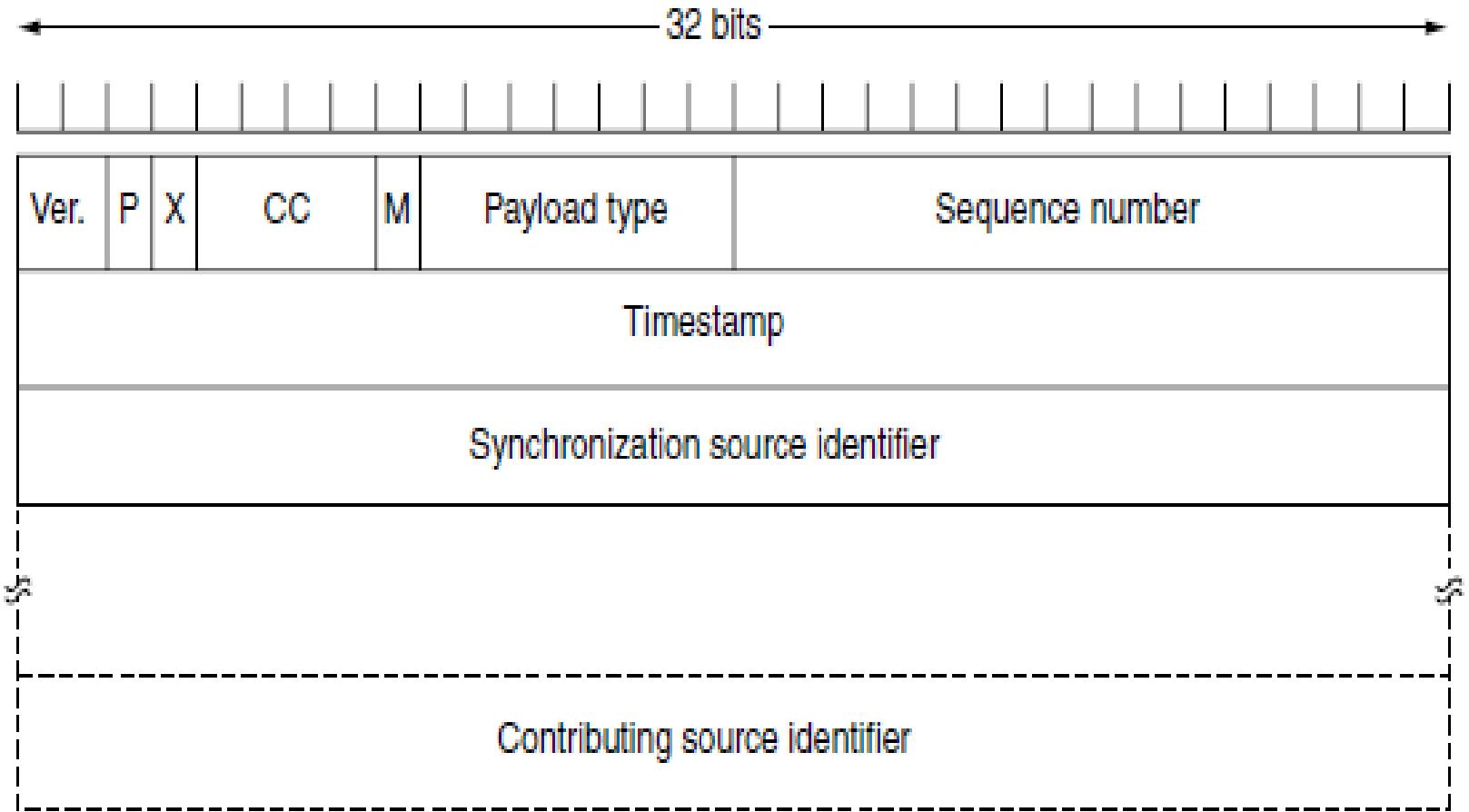


Figure 6-31. The RTP header.

It consists of three 32-bit words and potentially some extensions. The first word contains the *Version field*, which is already at 2. The *P bit* indicates that the packet has been padded to a multiple of 4 bytes. The last padding byte tells how many bytes were added. The *X bit* indicates that an extension header is present. The *CC field* tells how many contributing sources are present, from 0 to 15. The *M bit* is an application-specific marker bit. It can be used to mark the start of a video frame, the start of a word in an audio channel, or something else that the application understands. 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*, this value can help reduce timing variability called jitter at the receiver by decoupling the playback from the packet arrival time. 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 a packet contains multiple streams that are mixed together.

## RTCP—The Real-time Transport Control Protocol

RTP has a little sister protocol (little sibling protocol?) called RTCP (Real time Transport Control Protocol). It is defined along with RTP in RFC 3550 and handles feedback, synchronization, and the user interface. It does not transport any media samples.

The first function can be used to provide feedback on delay, variation in delay or jitter, bandwidth, congestion, and other network properties to the sources.

This information can be used by the encoding process to increase the data rate (and give better quality) when the network is functioning well and to cut back the data rate when there is trouble in the network. By providing continuous feedback, It provides the best quality

The *Payload type field* is used to tell the destination what encoding algorithm is used for the current packet, making it possible to vary it on demand.

RTCP also handles inter stream synchronization. The problem is that different streams may use different clocks, with different granularities and different drift rates. RTCP can be used to keep them in sync.

Finally, RTCP provides a way for naming the various sources (e.g., in ASCII text). This information can be displayed on the receiver's screen to indicate who is talking at the moment.

## Playout with Buffering and Jitter Control

Once the media information reaches the receiver, it must be played out at the right time. Even 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.

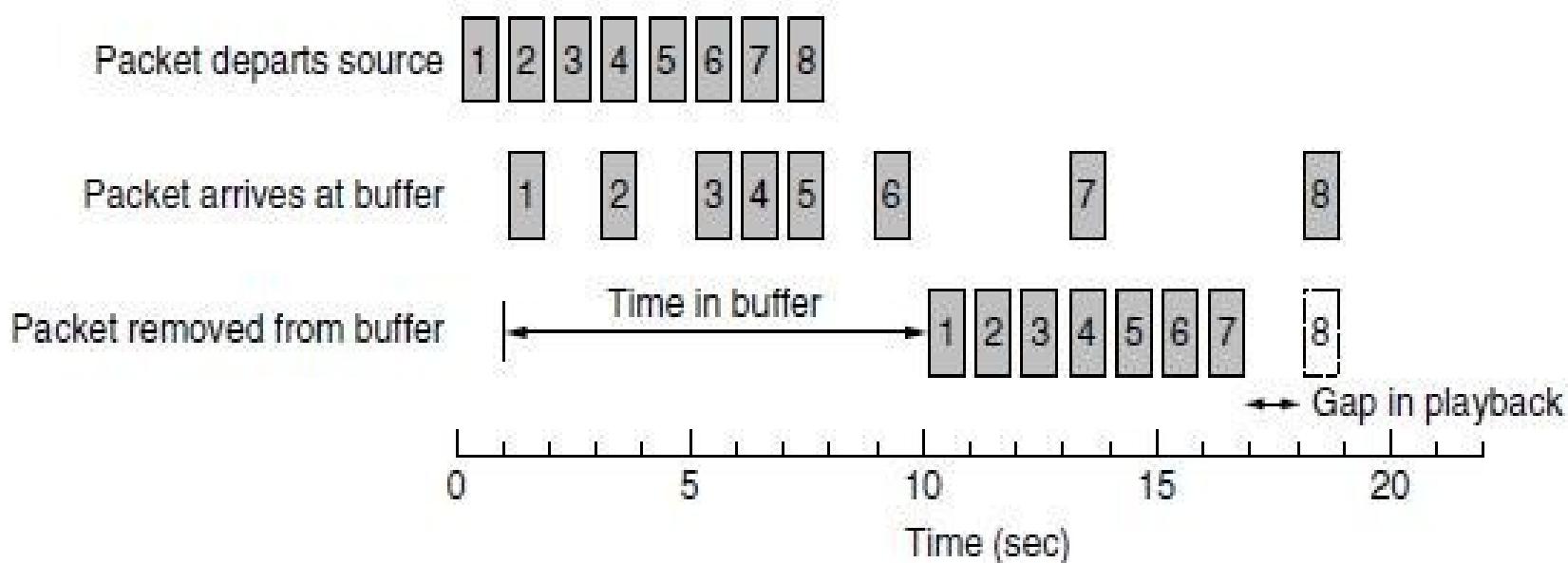


Figure 6-32. Smoothing the output stream by buffering packets.

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. The difference between a low-jitter and high-jitter connection is shown in Fig. The average delay may not differ greatly between the two, but if there is high jitter the playback point may need to be much further out to capture 99% of the packets than if there is low jitter.

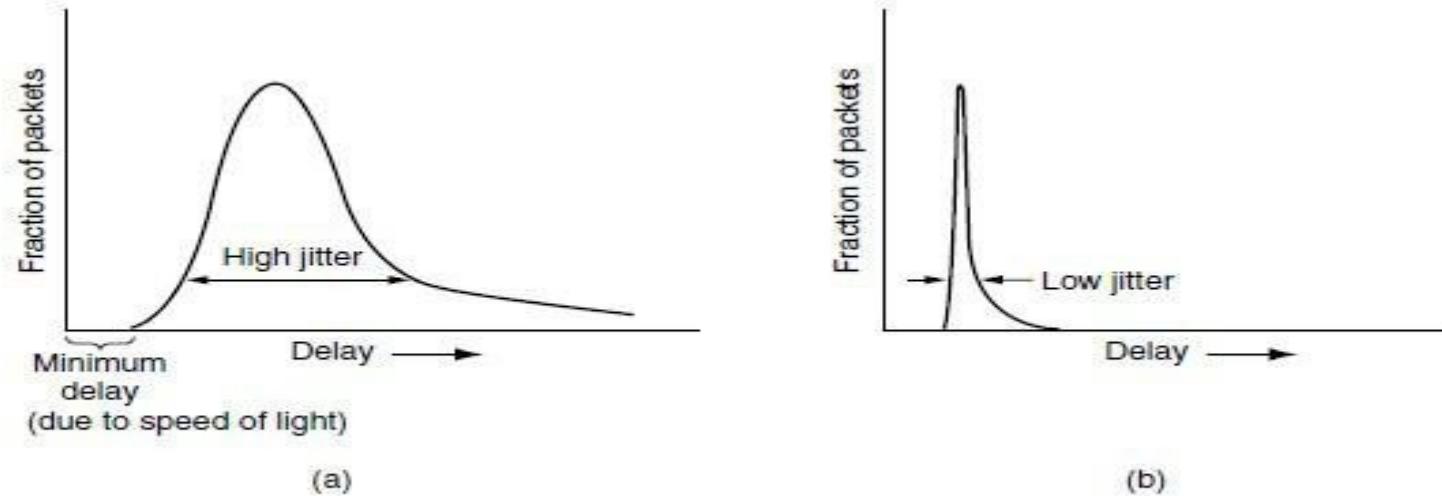


Figure 6-33. (a) High jitter. (b) Low jitter.

One way to avoid this problem for audio is to adapt the playback point between **talk spurts**, in the gaps in a conversation. No one will notice the difference between a short and slightly longer silence

## **TELNET**

It is client/server application program. TELNET is an abbreviation for *TERminaL NETwork*. TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

### *Timesharing Environment*

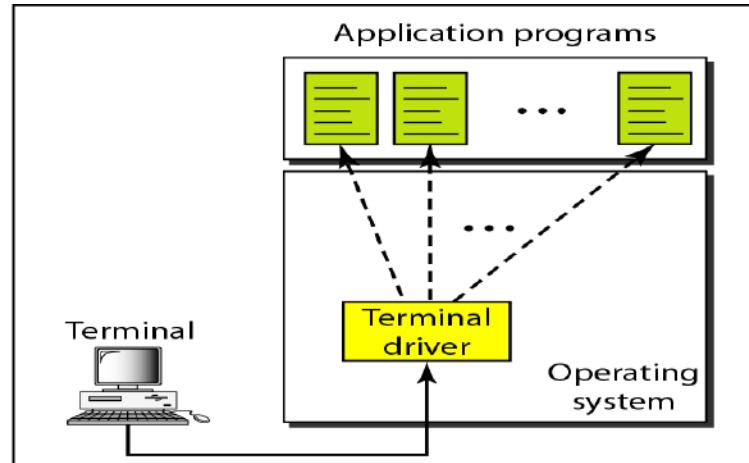
A large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse.

### *Logging*

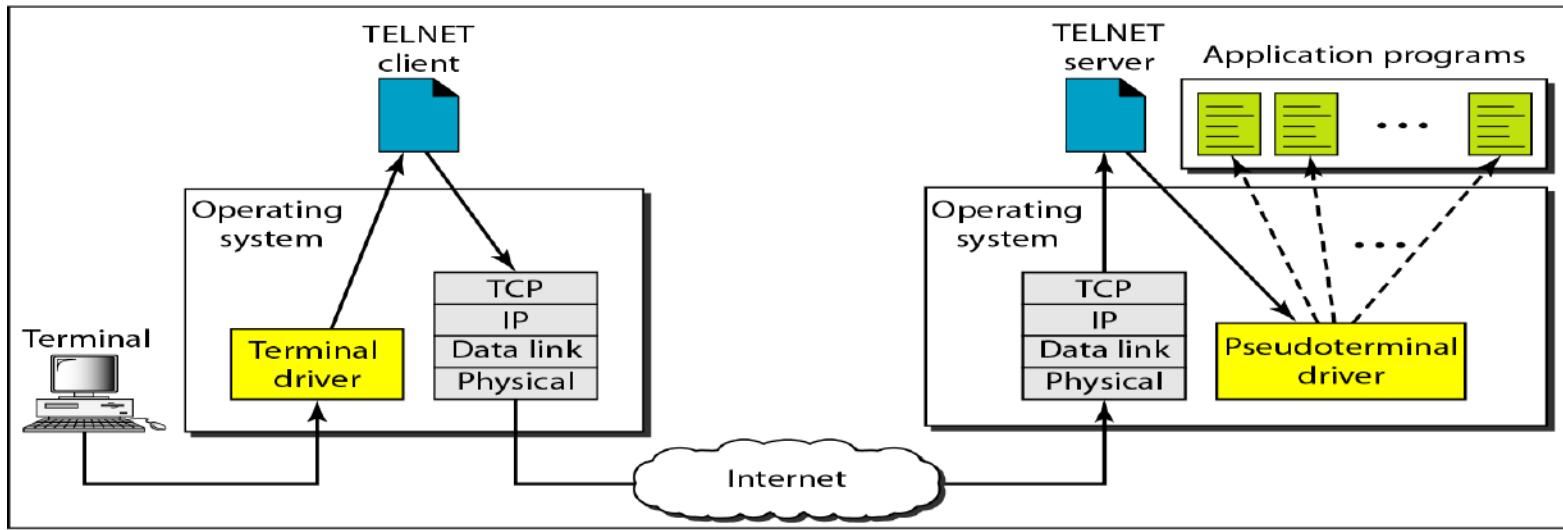
To access the system, the user logs into the system with a user id or log-in name. The system also includes password checking to prevent an unauthorized user from accessing the resources.

Local login

Remote login



a. Local log-in



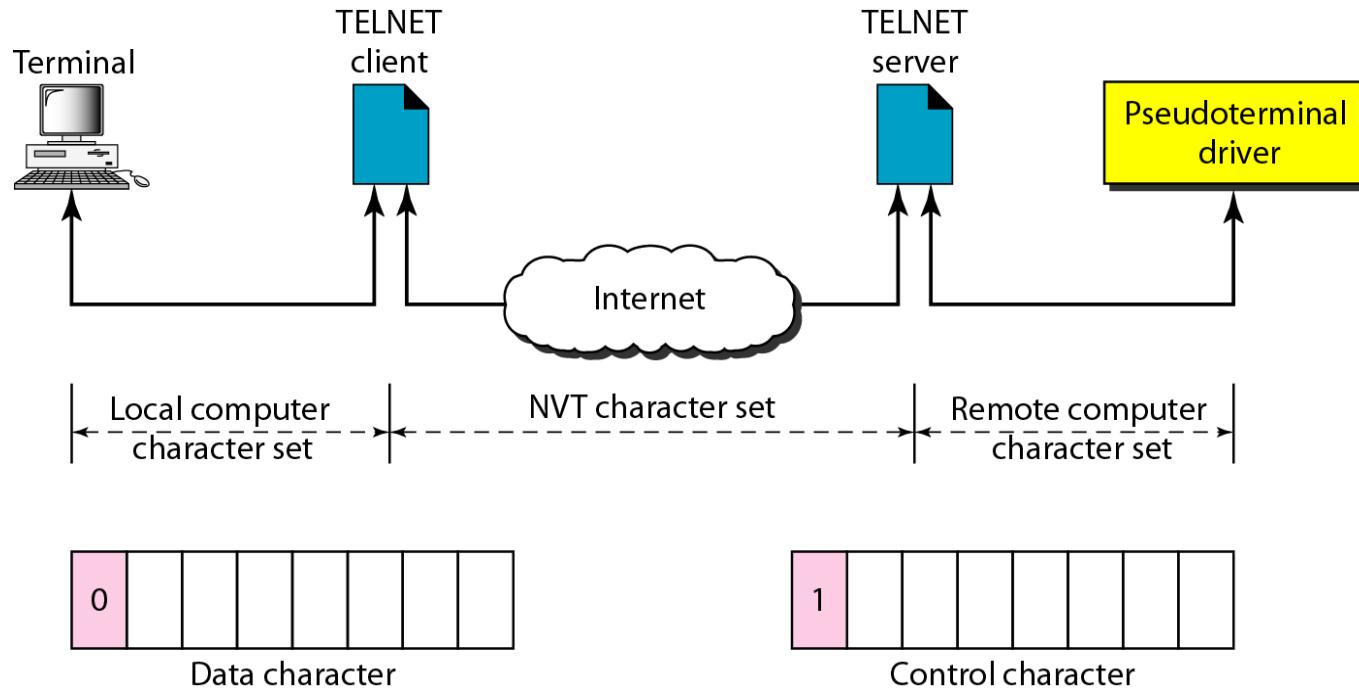
b. Remote log-in

When a user logs into a local timesharing system, it is called **local log-in**. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

When a user wants to access an application program or utility located on a remote Machine, it is called **remote log-in**. Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them. The characters are sent to the TELNET client, which transforms the characters to a universal character set called network virtual terminal (NVT) characters and delivers them to the local TCP/IP protocol stack.

The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver. The solution is to add a piece of

## *Concept of NVT(network virtual terminal)*





# Data Communications and Networking

Fourth Edition

Forouzan

## WWW and HTTP

*The WWW today is a distributed client/server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called sites as shown in fig.*

### Topics discussed in this section:

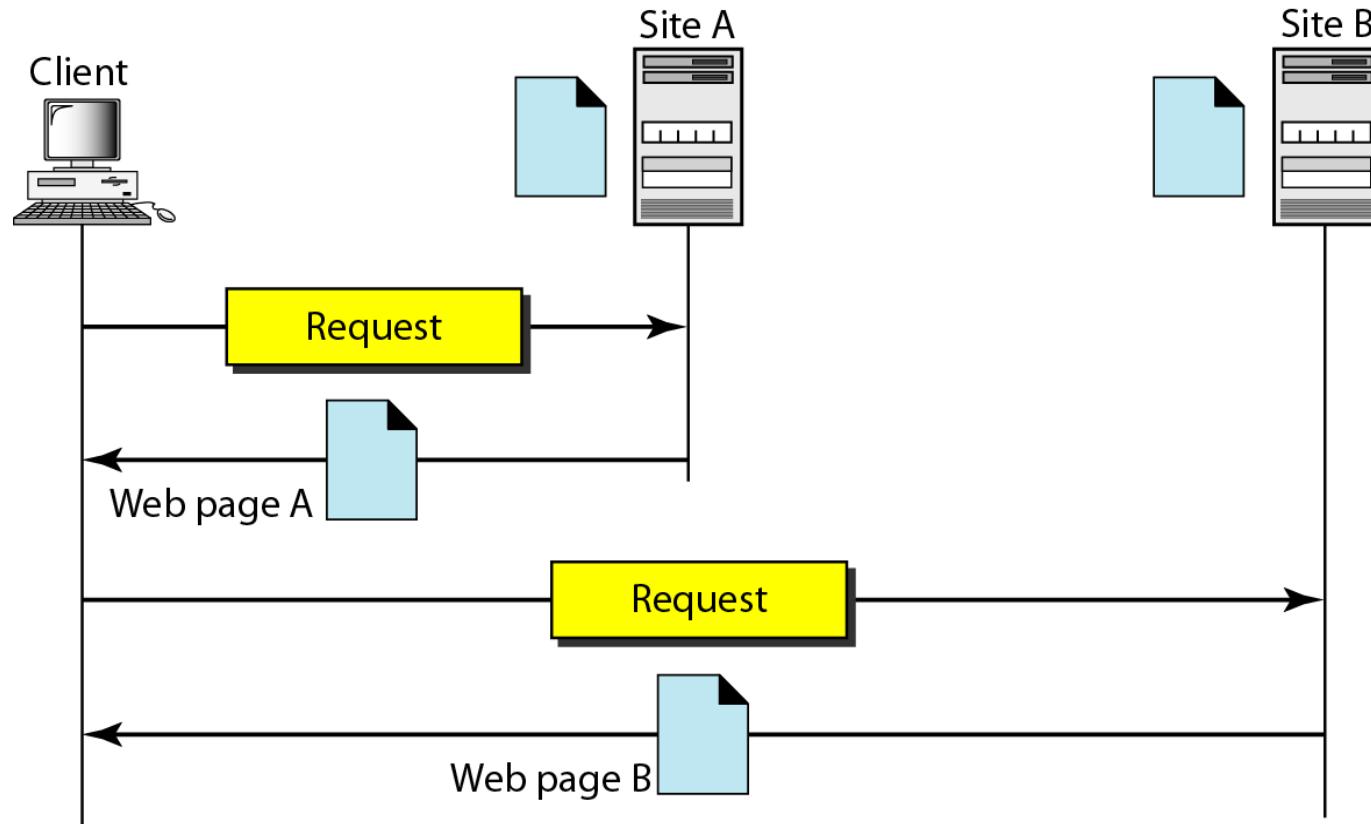
Client (Browser)

Server

Uniform Resource Locator

Cookies

**Figure 27.1** *Architecture of WWW*



## Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a Web document, and all use nearly the same architecture.

Each browser usually consists of three parts: a controller, client protocol, and interpreters.

The controller receives input from the keyboard or the mouse and uses the client programs to access the document.

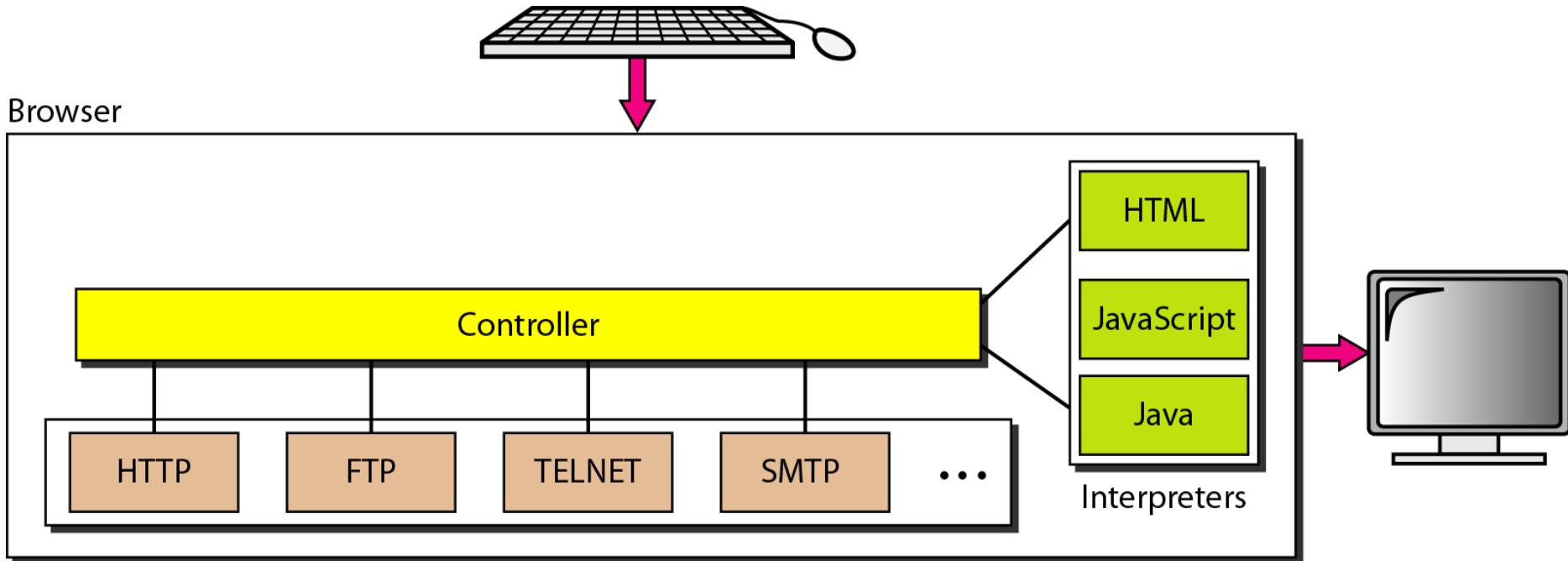
After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The interpreter can be HTML, Java, or JavaScript, depending on the type of document

The client protocol can be one of the protocols described previously such as FTP or HTTP.

## Server

The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk. A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time.

Figure 27.2 *Browser*



## Uniform Resource Locator

A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path.

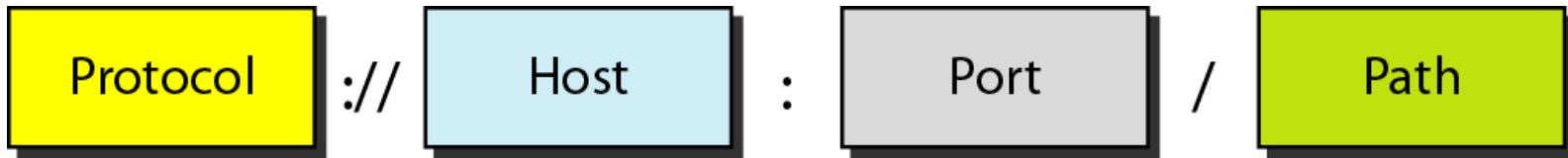
The *protocol* is the *client/server program used to retrieve the document*. Many different protocols can retrieve a document; among them are FTP or HTTP. The most common today is HTTP.

The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www".

The URL can optionally contain the port number of the server. If the *port* is included, it is inserted between the host and the path, and it is separated from the host by a colon.

Path is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

## Figure 27.3 URL



An HTTP cookie (also called web cookie, Internetcookie, browser cookie or simply cookie, the latter which is not to be confused with the literal definition), is a small piece of data sent from a website and stored in a user's web browser while the user is browsing that website

*The documents in the WWW can be grouped into three broad categories: static, dynamic, and active. The category is based on the time at which the contents of the document are determined.*

### Topics discussed in this section:

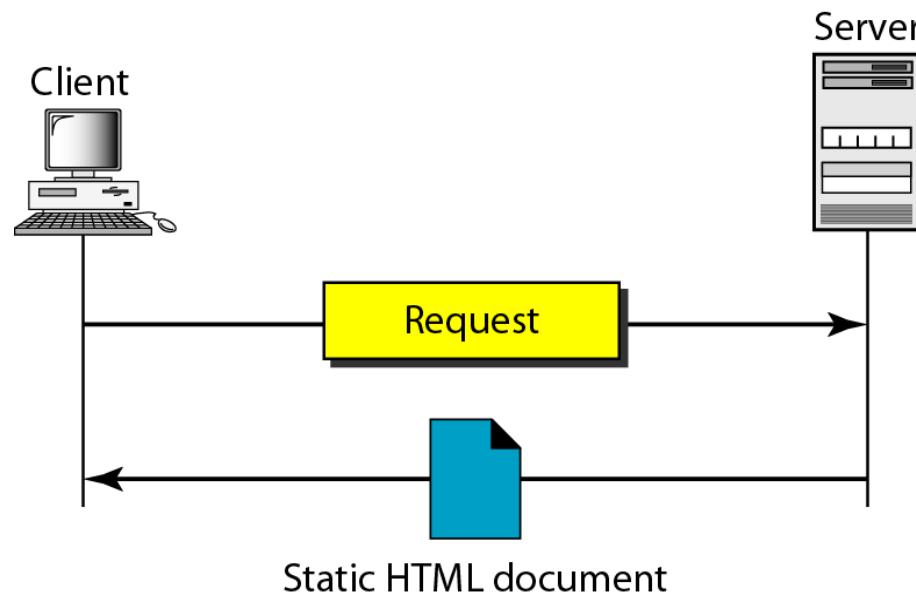
Static Documents

Dynamic Documents

Active Documents

## Static Documents

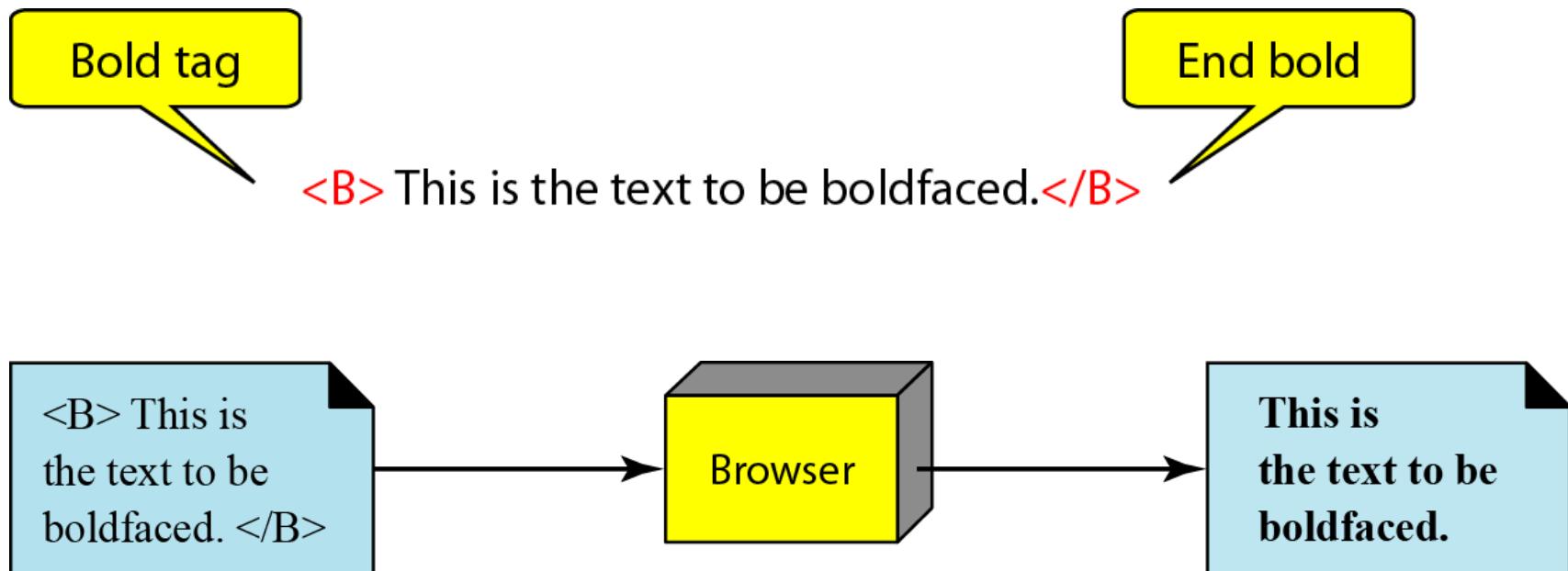
Static documents are fixed-content documents that are created and stored in a server. The client can get only a copy of the document. When a client accesses the document, a copy of the document is sent. The user can then use a browsing program to display the document



**Figure 27.5** *Boldface tags*

## *HTML*

Hypertext Markup Language (HTML) is a language for creating Web pages.



## Figure 27.7 Beginning and ending tags

< TagName Attribute = Value Attribute = Value ... >

a. Beginning tag

< /TagName >

b. Ending tag

## Dynamic Documents

A dynamic document is created by a Web server whenever a browser requests the document. When a request arrives, the Web server runs an application program or a script that creates the dynamic document. The server returns the output of the program or script as a response to the browser that requested the document.

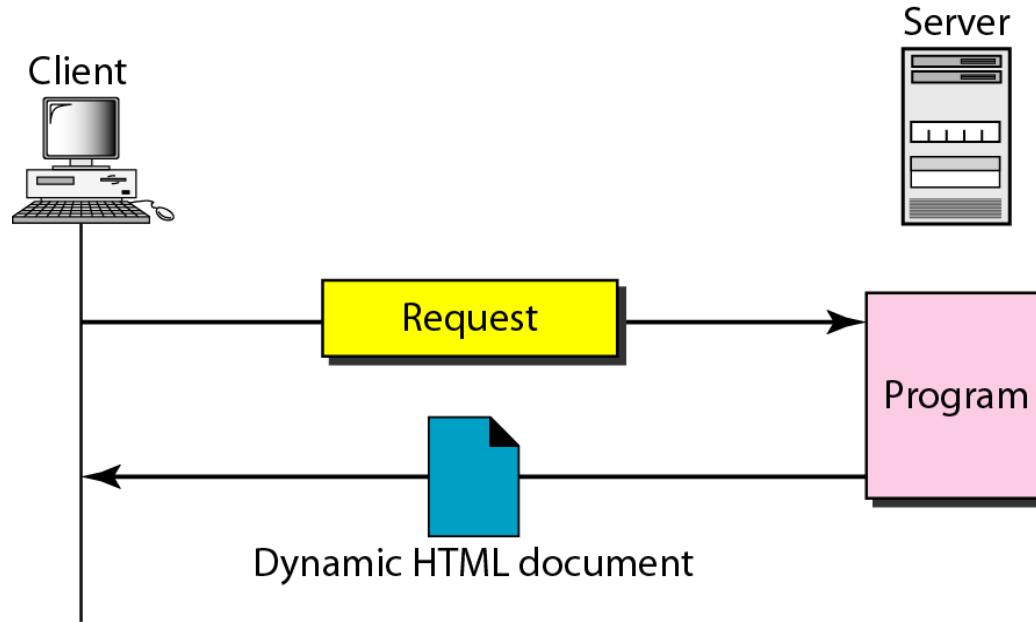
A very simple example of a dynamic document is the retrieval of the time and date from a server. Time and date are kinds of information that are dynamic in that they change from moment to moment. The client can ask the server to run a program such as the *date program in UNIX and send the result of the program to the client*.

### Common Gateway Interface (CGI)

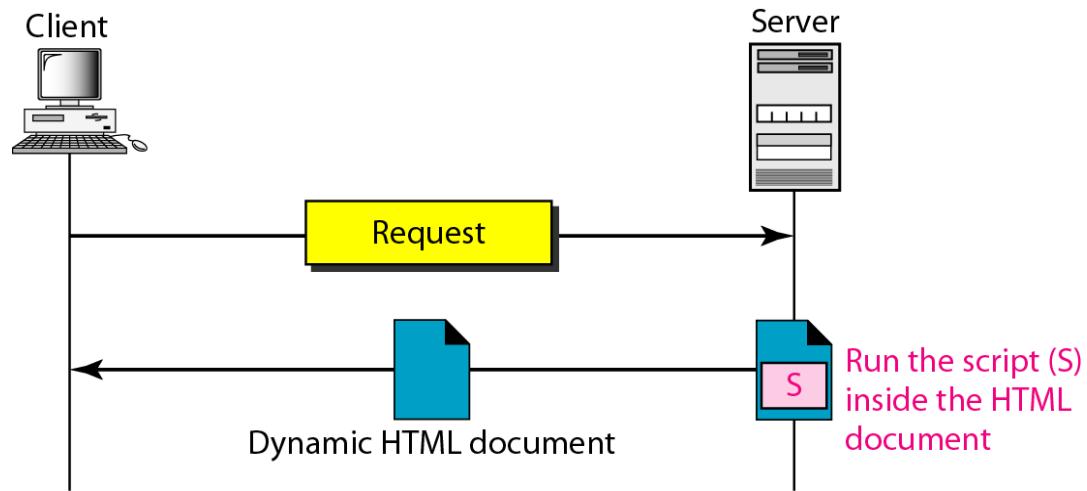
The Common Gateway Interface (CGI) is a technology that creates and handles dynamic documents.

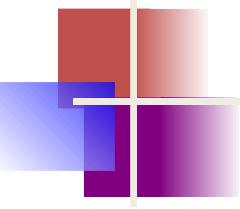
Hypertext Preprocessor (pHP), which uses the Perl language; Java Server Pages (JSP), which uses the Java language for scripting; Active Server Pages (ASP), a Microsoft product which uses Visual Basic language for scripting; and ColdFusion, which embeds SQL database queries in the HTML document

**Figure 27.8** *Dynamic document using CGI*



**Figure 27.9** Dynamic document using server-site script





## *Note*

---

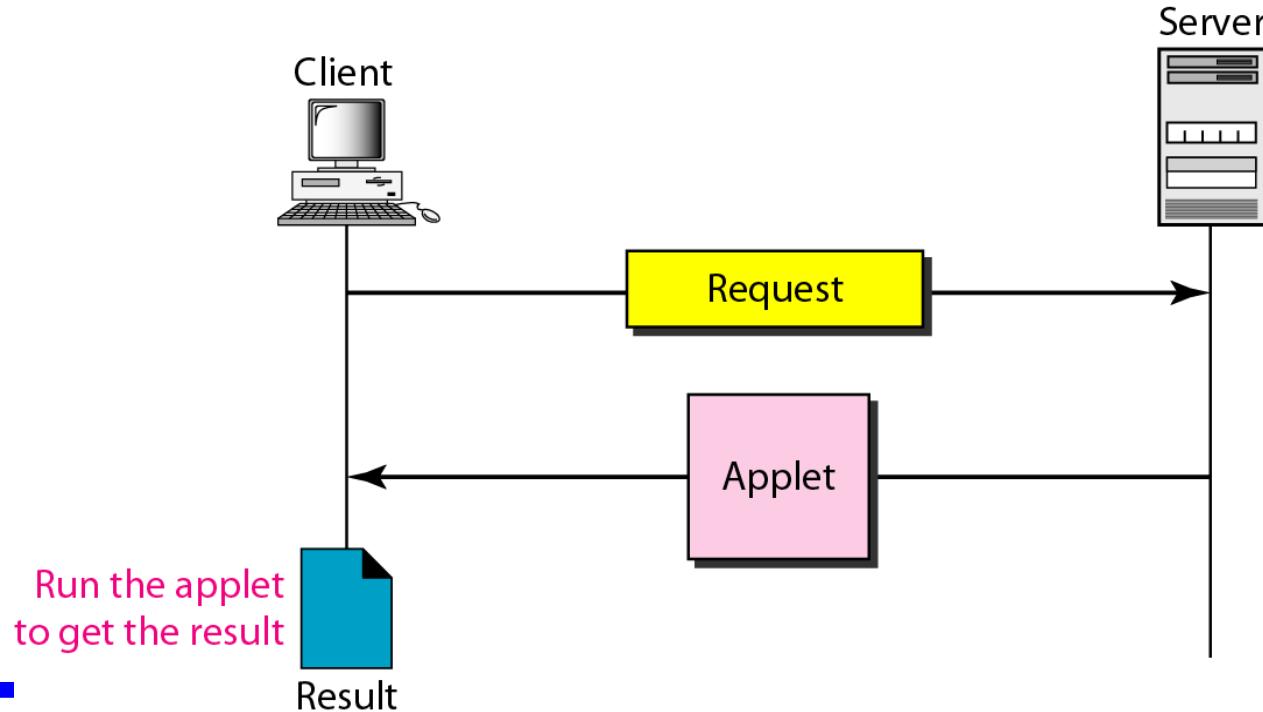
Dynamic documents are sometimes referred to as server-site dynamic documents.

---

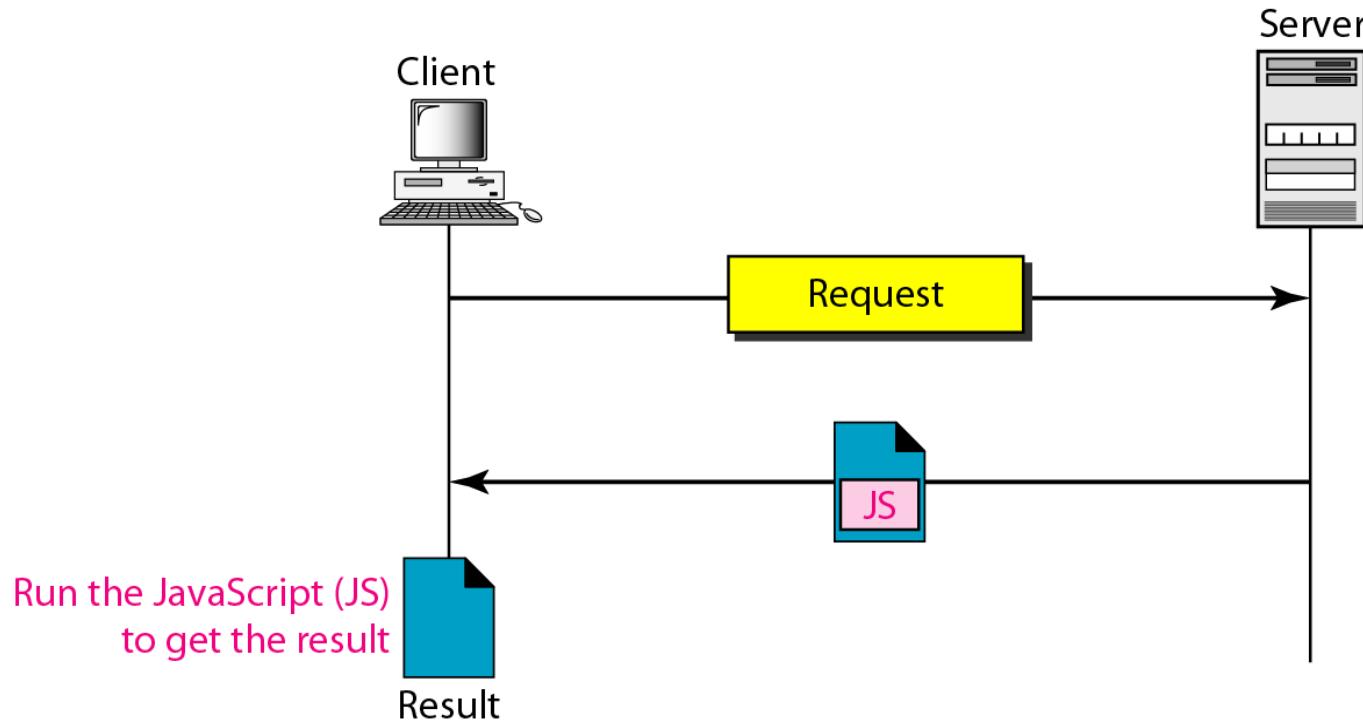
**Figure 27.10** Active document using Java applet

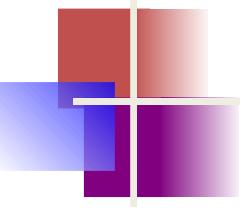
## Active Documents

For many applications, we need a program or a script to be run at the client site. These are called active documents



**Figure 27.11** Active document using client-site script



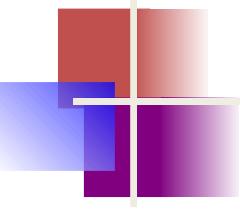


## *Note*

---

Active documents are sometimes referred to as client-site dynamic documents.

---



## *Note*

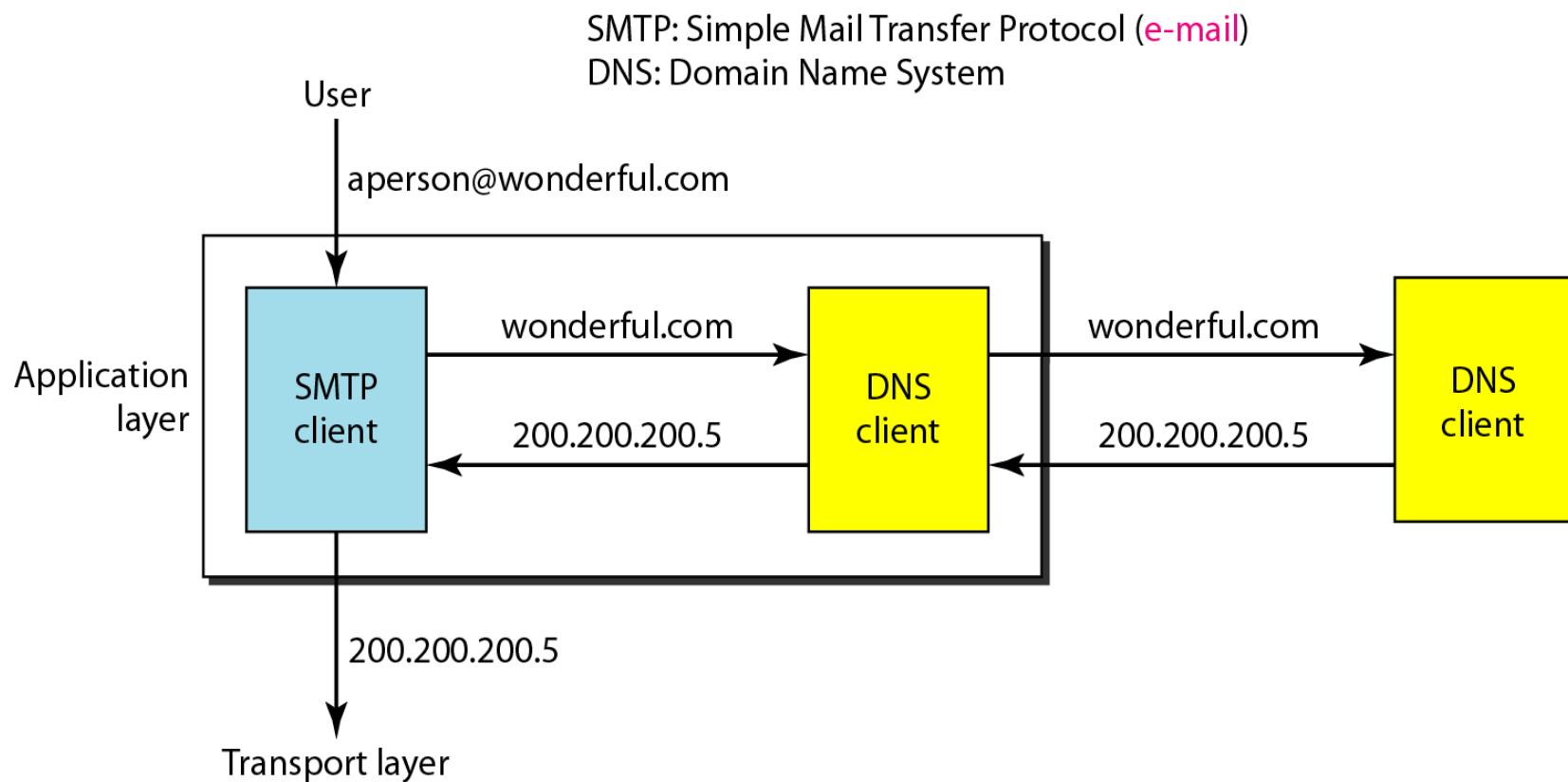
---

HTTP version 1.1 specifies a persistent connection by default.

---

## DNS (Domain Name System)

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name.



## NAME SPACE

A name space that maps each address to a unique name can be organized in two ways: flat or hierarchical.

### Flat Name Space

In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure.

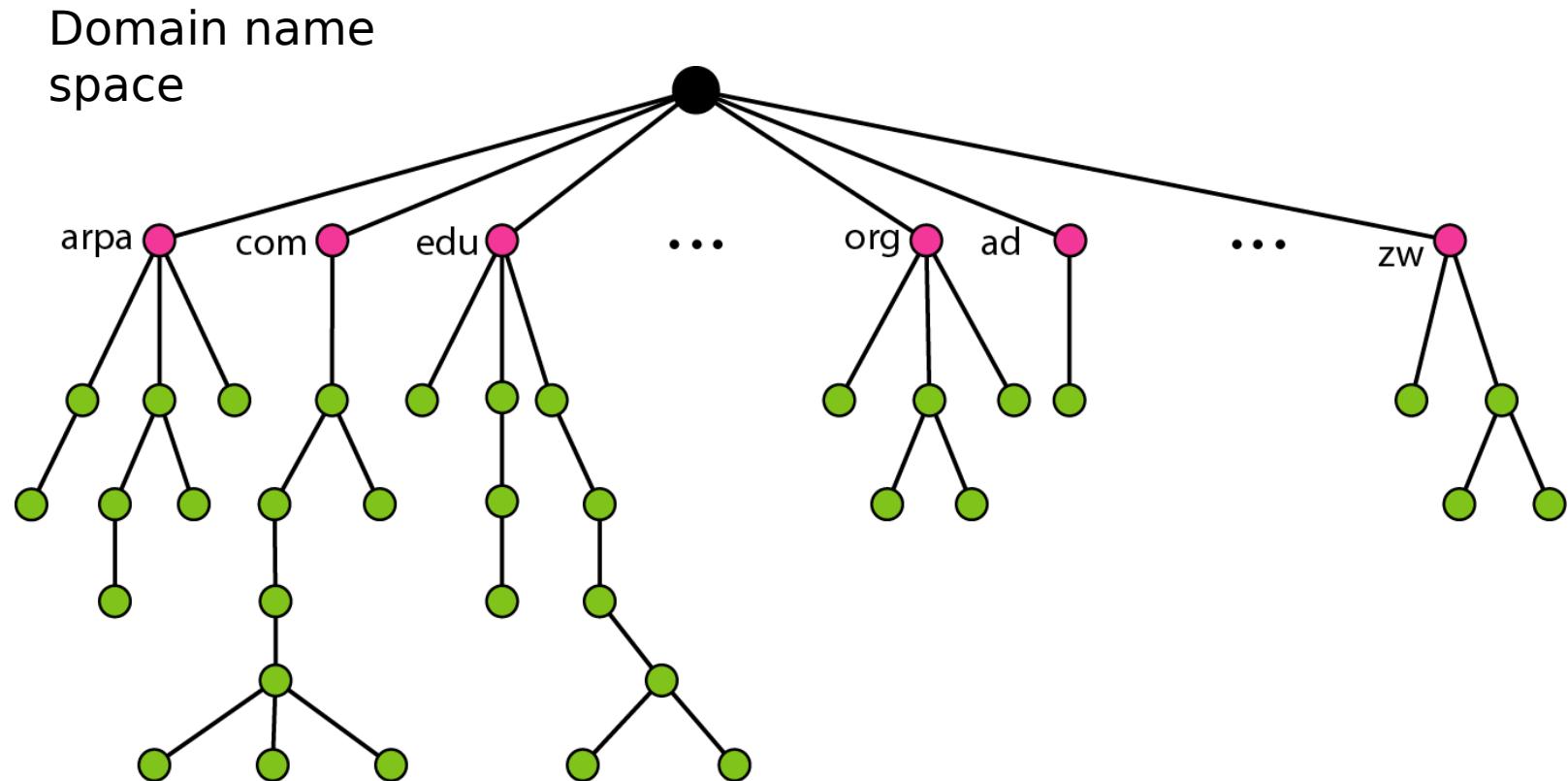
### Hierarchical Name Space

In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on.

Exa:      *challenger.jhda.edu*,      *challenger.berkeley.edu*,      and  
*challenger.smart.com*

## DOMAIN NAME SPACE

To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.



## Label

Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

## Domain Name

Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing. Below Figure shows some domain names

-

FQDN

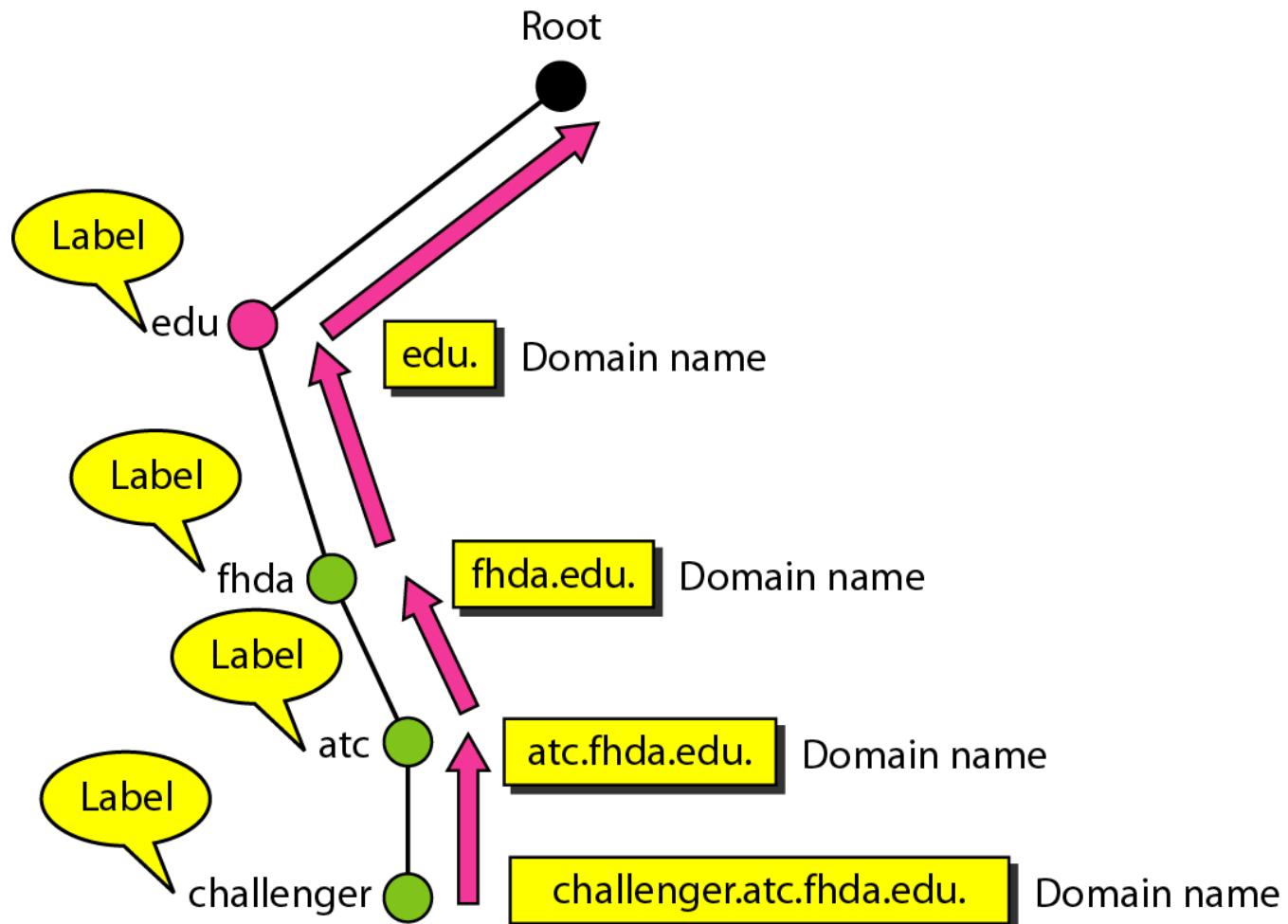
PQDN

challenger.atc.fhda.edu.  
cs.hmme.com.  
www.funny.int.

challenger.atc.fhda.edu  
cs.hmme  
www

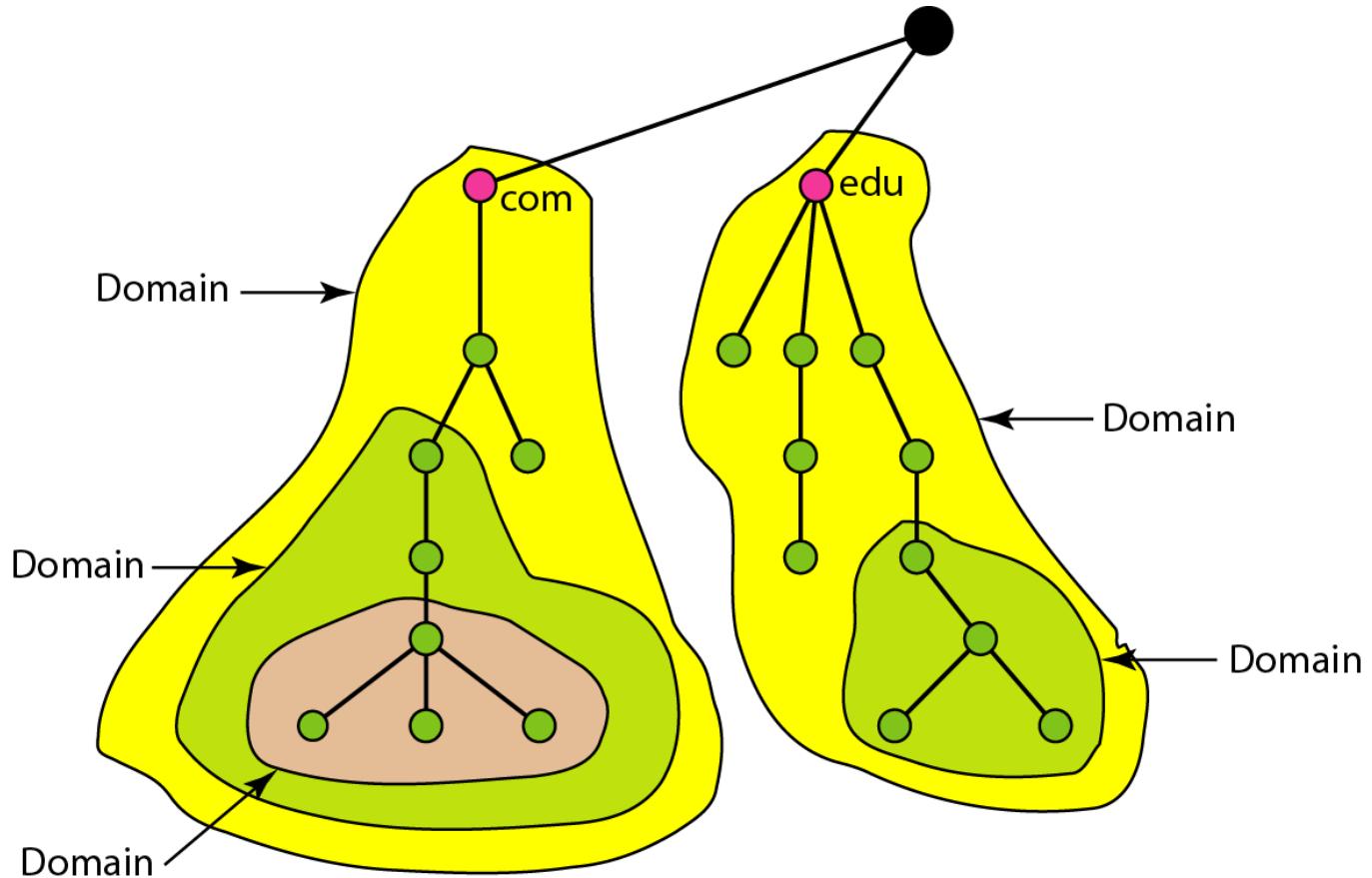
## *Domain names and labels*

---



## Domain

A domain is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree.

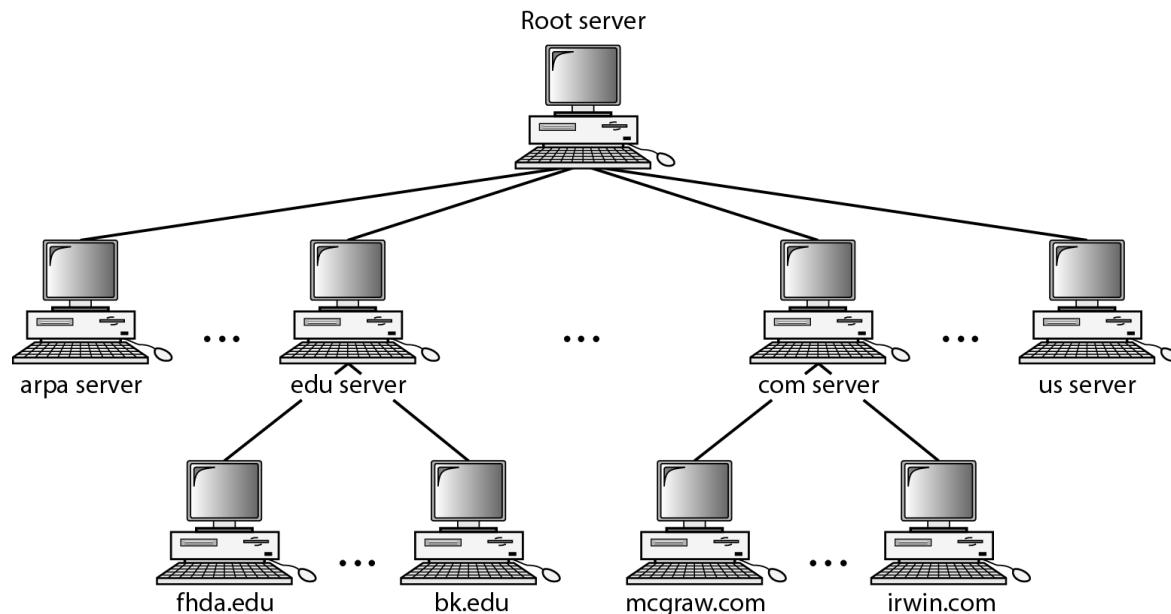


## DISTRIBUTION OF NAME SPACE:

The information contained in the domain name space must be stored. However, it is very inefficient and also unreliable to have just one computer store such a huge amount of information. In this section, we discuss the distribution of the domain name space

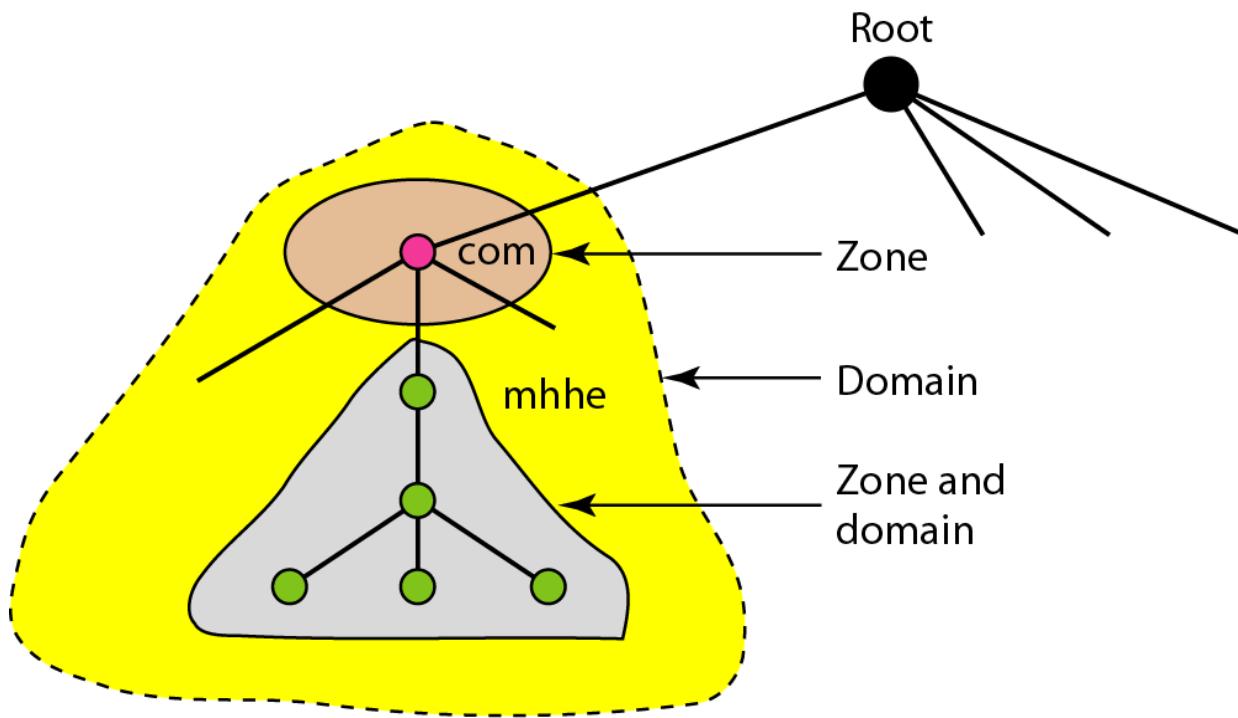
### 1 Hierarchy of Name Servers

distribute the information among many computers called DNS servers. we let the root stand alone and create as many domains (subtrees) as there are first-level nodes



## 2 Zone

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a zone. We can define a zone as a contiguous part of the entire tree



### 3 Root Server

A root server is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers. There are several root servers, each covering the whole domain name space. The servers are distributed all around the world.

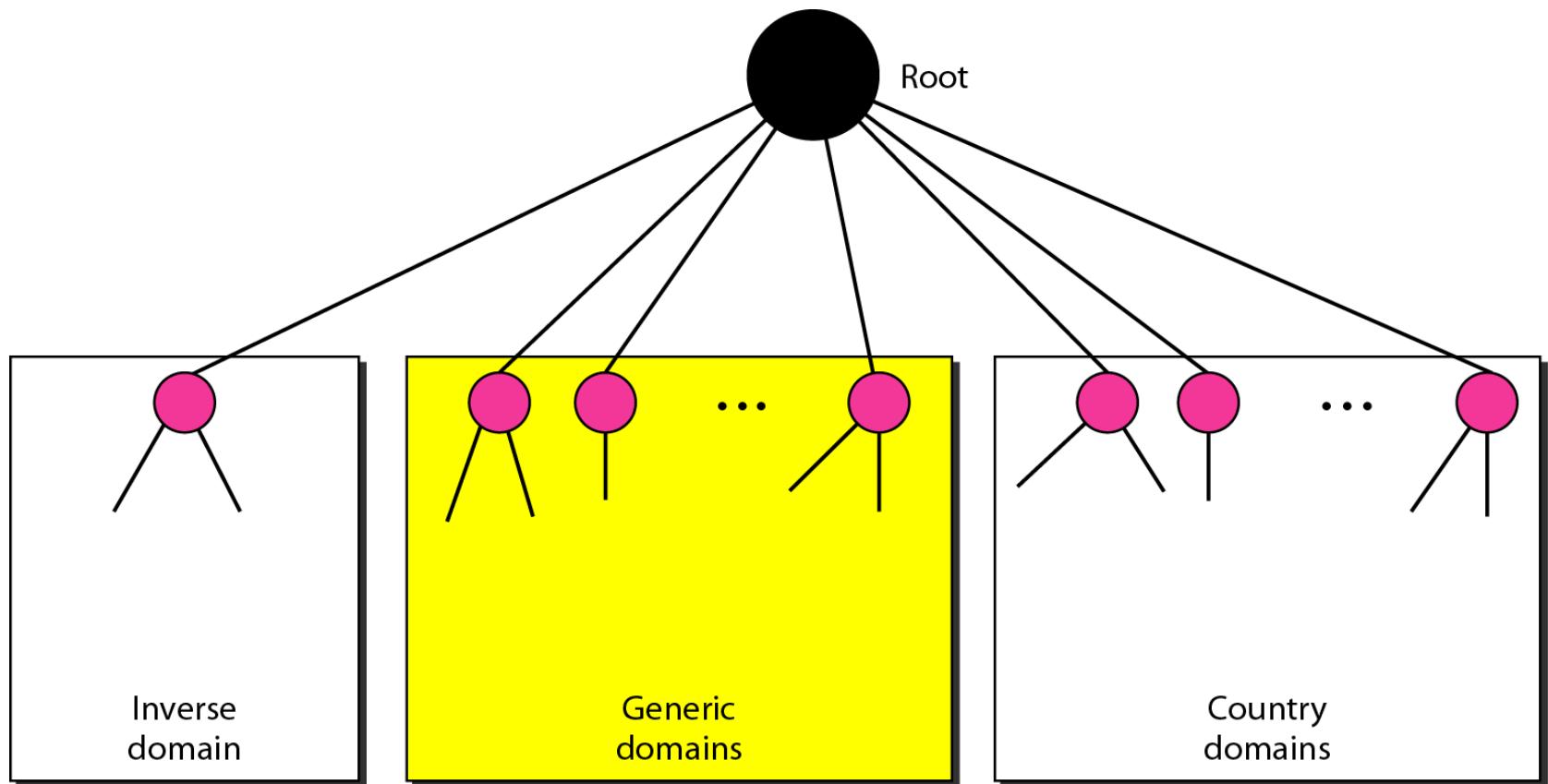
### 4 Primary and Secondary Servers

A primary server is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk

A secondary server is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files

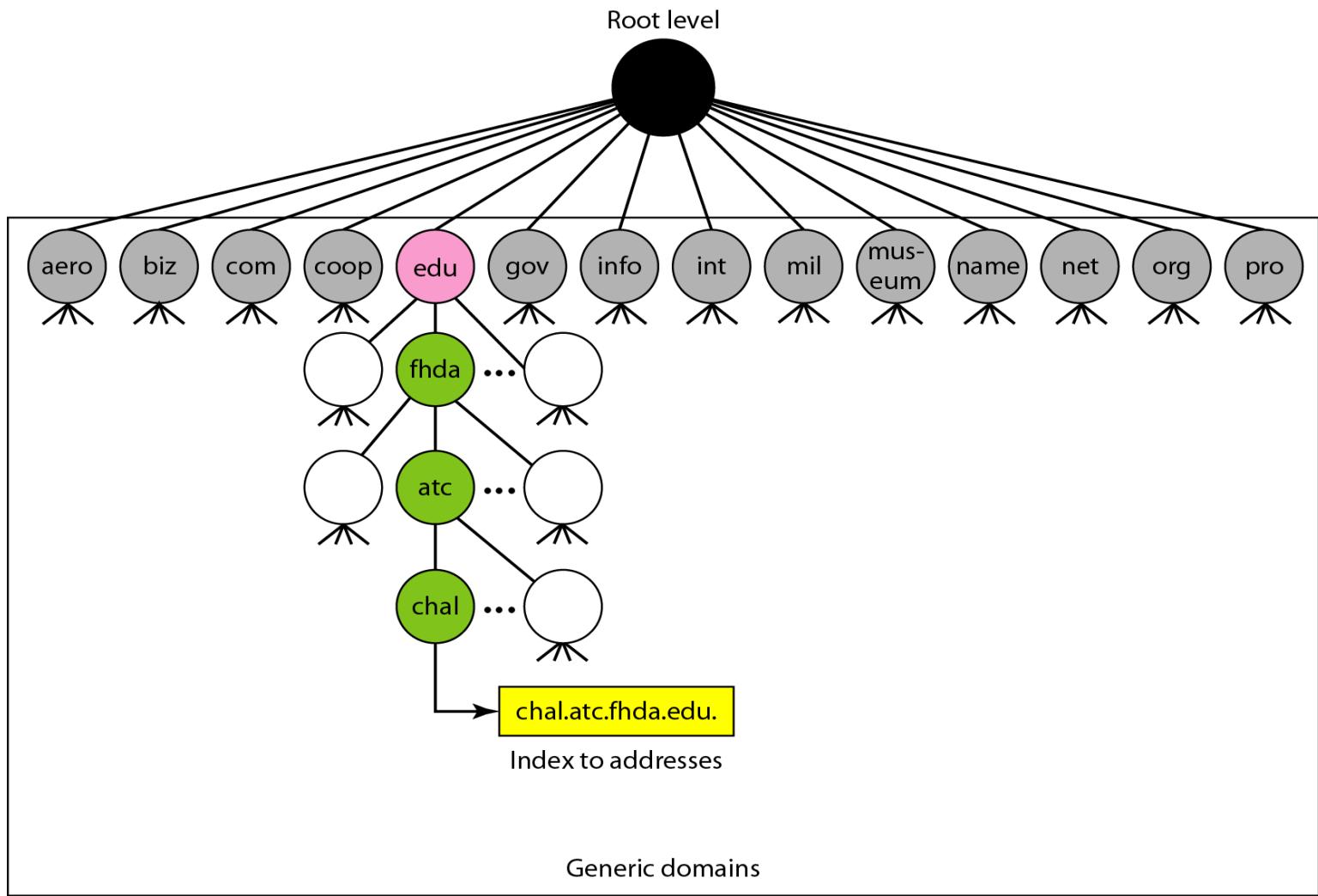
## DNS IN THE INTERNET

DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) is divided into three different sections: generic domains, country domains, and the inverse domain



## 1 Generic Domains

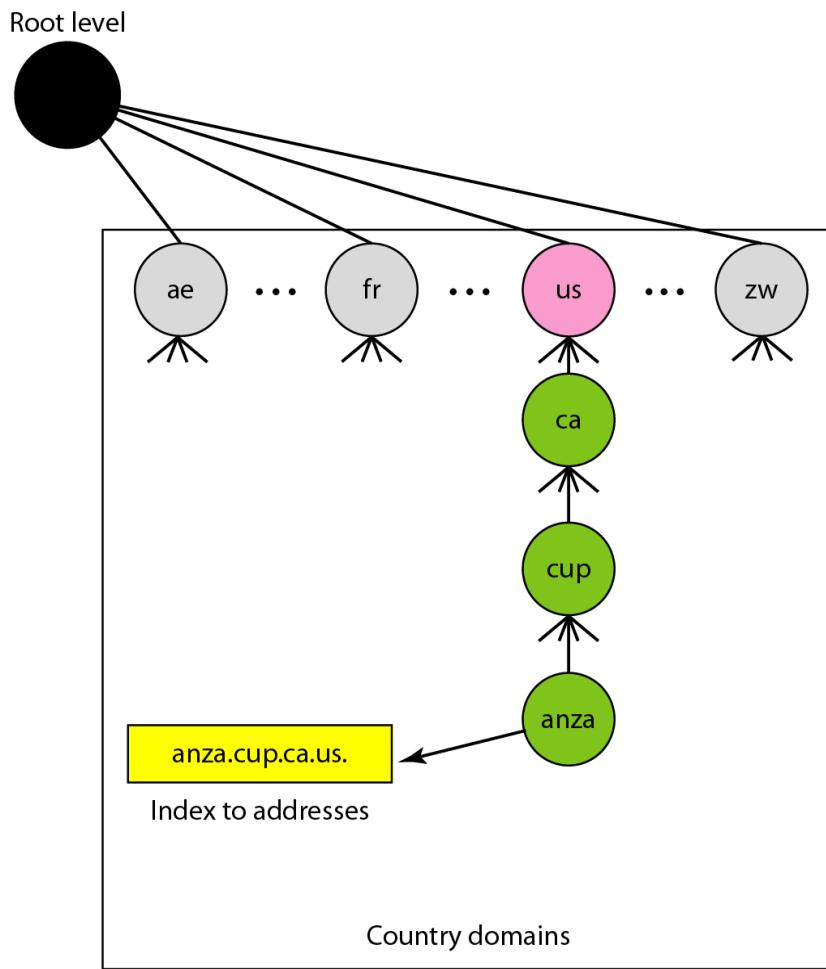
The generic domains define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database



<i>Label</i>	<i>Description</i>
<b>aero</b>	Airlines and aerospace companies
<b>biz</b>	Businesses or firms (similar to “com”)
<b>com</b>	Commercial organizations
<b>coop</b>	Cooperative business organizations
<b>edu</b>	Educational institutions
<b>gov</b>	Government institutions
<b>info</b>	Information service providers
<b>int</b>	International organizations
<b>mil</b>	Military groups
<b>museum</b>	Museums and other nonprofit organizations
<b>name</b>	Personal names (individuals)
<b>net</b>	Network support centers
<b>org</b>	Nonprofit organizations
<b>pro</b>	Professional individual organizations

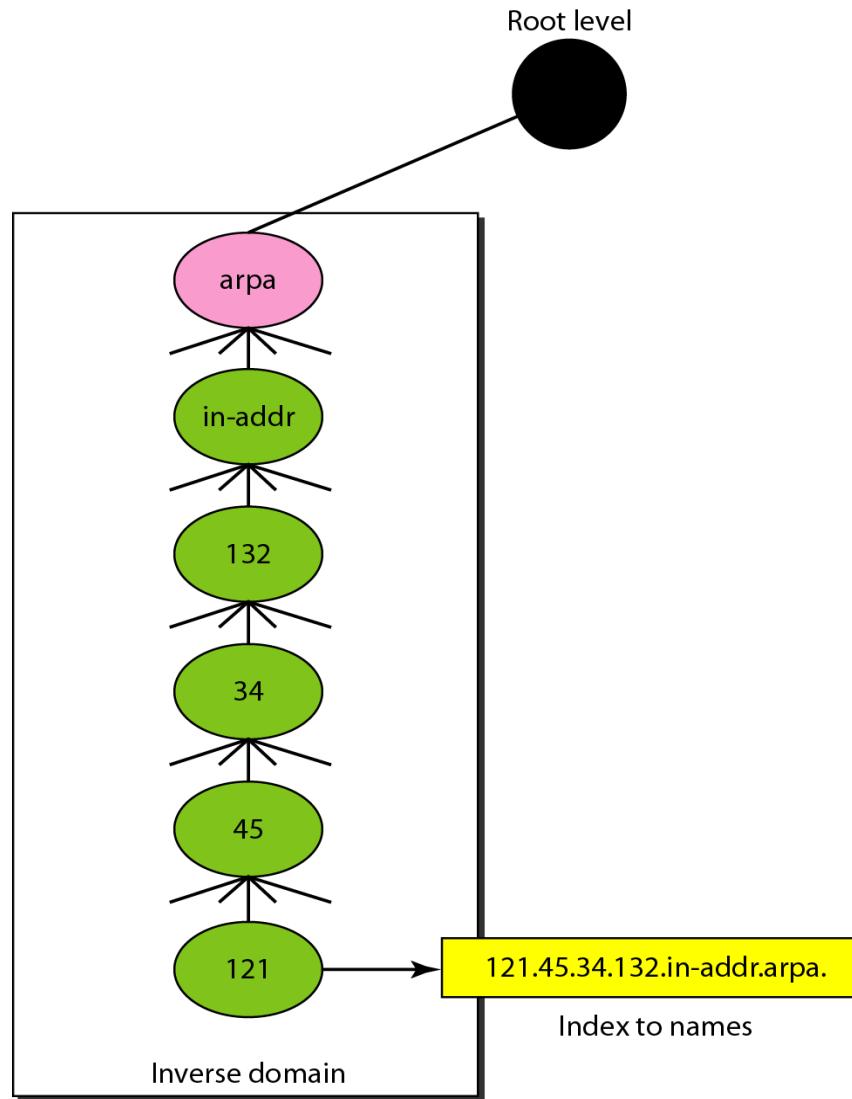
## 2 Country Domains

The country domains section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).



### 3 Inverse Domain

The inverse domain is used to map an address to a name.



## RESOLUTION

Mapping a name to an address or an address to a name is called name address resolution

### **1 Resolver**

DNS is designed as a client/server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information.

### **2 Mapping Names to Addresses**

In this case, the server checks the generic domains or the country domains to find the mapping.

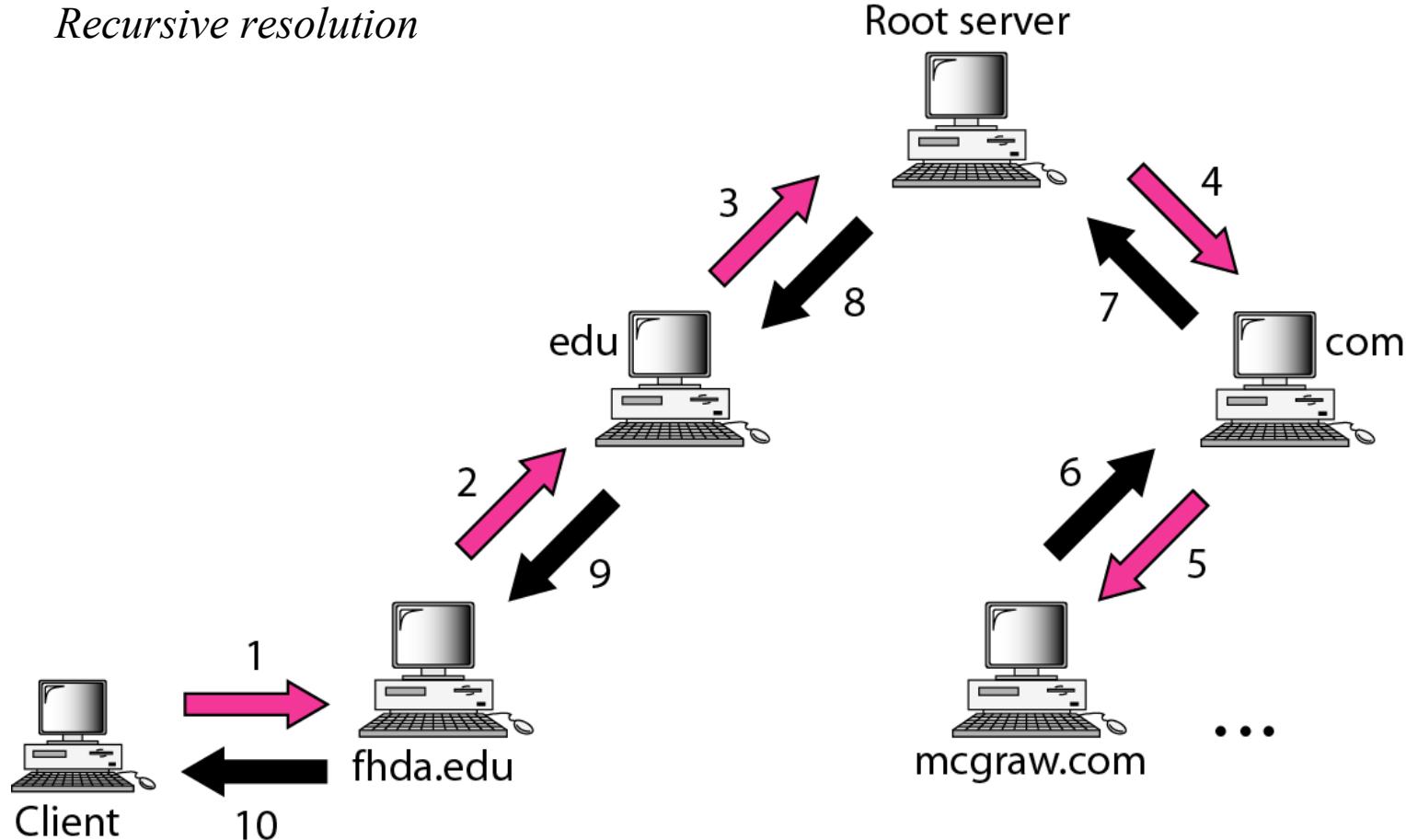
### **3 Mapping Addresses to Names**

To answer queries of this kind, DNS uses the inverse domain

### **4 Recursive Resolution**

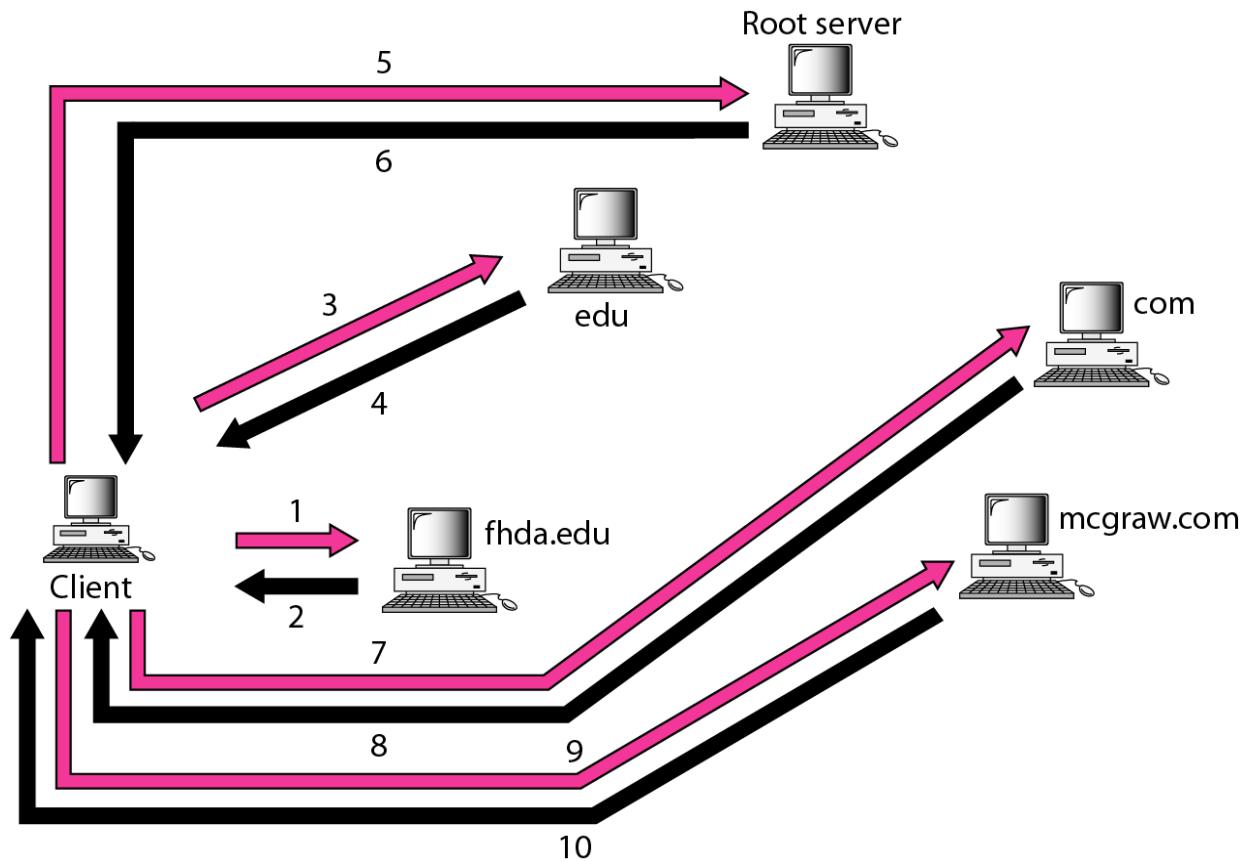
The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer.. When the query is finally resolved, the response travels back until it finally reaches the requesting client. This is called recursive resolution and is shown in FIG

*Recursive resolution*



## 5 Iterative Resolution

If the client does not ask for a recursive answer, the mapping can be done iteratively. If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query



## 6 Caching

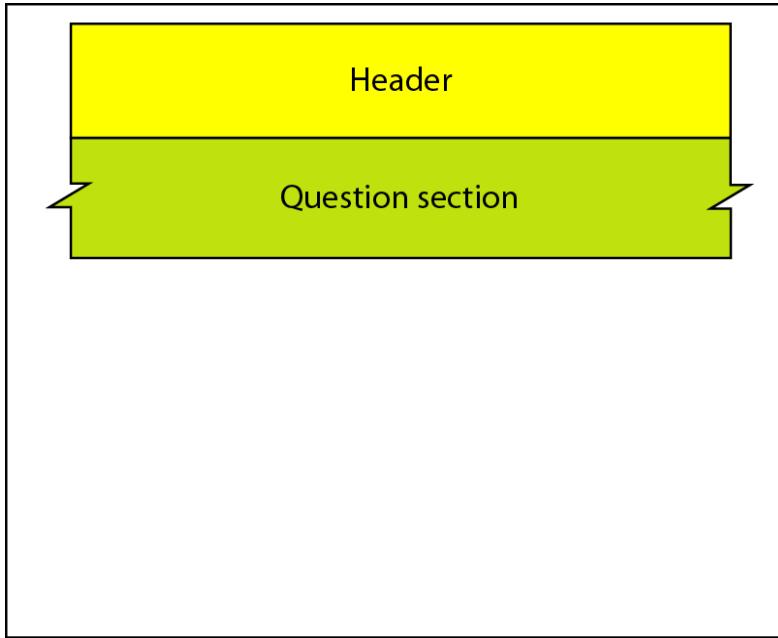
Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called caching.

## DNS MESSAGES

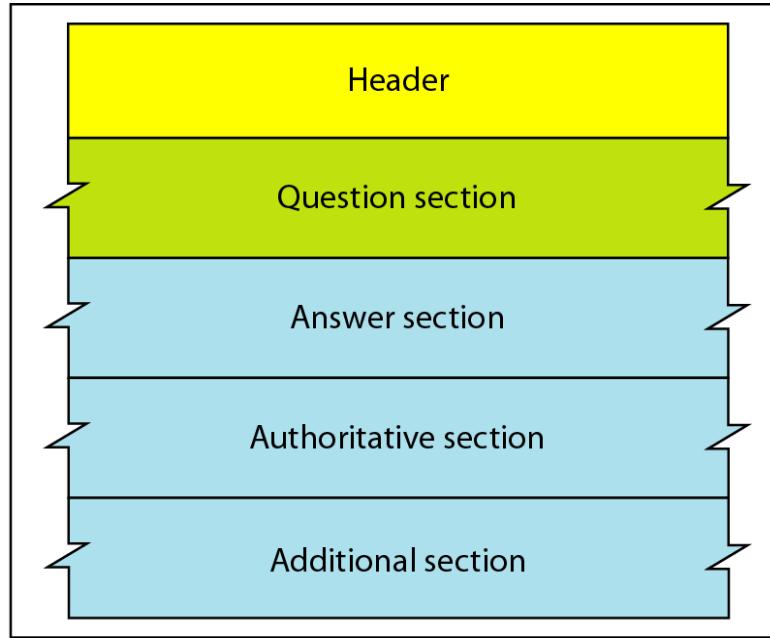
DNS has two types of messages: query and response. Both types have the same format.

The query message consists of a header,  
and question records;

the response message consists of a header,  
question records,  
answer records,  
authoritative records,  
and additional records



a. Query



b. Response

## Header

Both query and response messages have the same header format with some fields set to zero for the query messages. The header is 12 bytes,

Identification	Flags
Number of question records	Number of answer records (all 0s in query message)
Number of authoritative records (all 0s in query message)	Number of additional records (all 0s in query message)

## **TYPES OF RECORDS**

The question records are used in the question section of the query and response messages. The resource records are used in the answer, authoritative, and additional information sections of the response message.

### **Question Record**

A question record is used by the client to get information from a server..

### **Resource Record**

Each domain name (each node on the tree) is associated with a record called the resource record. The server database consists of resource records. Resource records are also what is returned by the server to the client.

## **REGISTRARS**

How are new domains added to DNS? This is done through a registrar, a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged. Today, there are many registrars; their names and addresses can be found at <http://www.intenic.net>

## **DYNAMIC DOMAIN NAME SYSTEM (DDNS)**

In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file. The size of today's Internet does not allow for this kind of manual operation.

The DNS master file must be updated dynamically. The Dynamic Domain Name System (DDNS) therefore was devised to respond to this need.

## **ENCAPSULATION**

DNS can use either UDP or TCP. In both cases the well-known port used by the server is port 53.