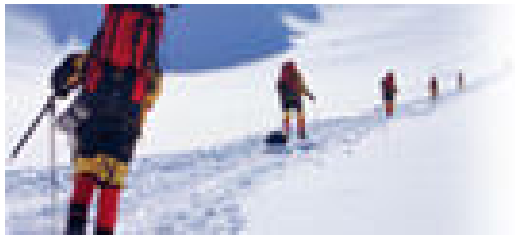


Data Communications and Networking Fourth Edition

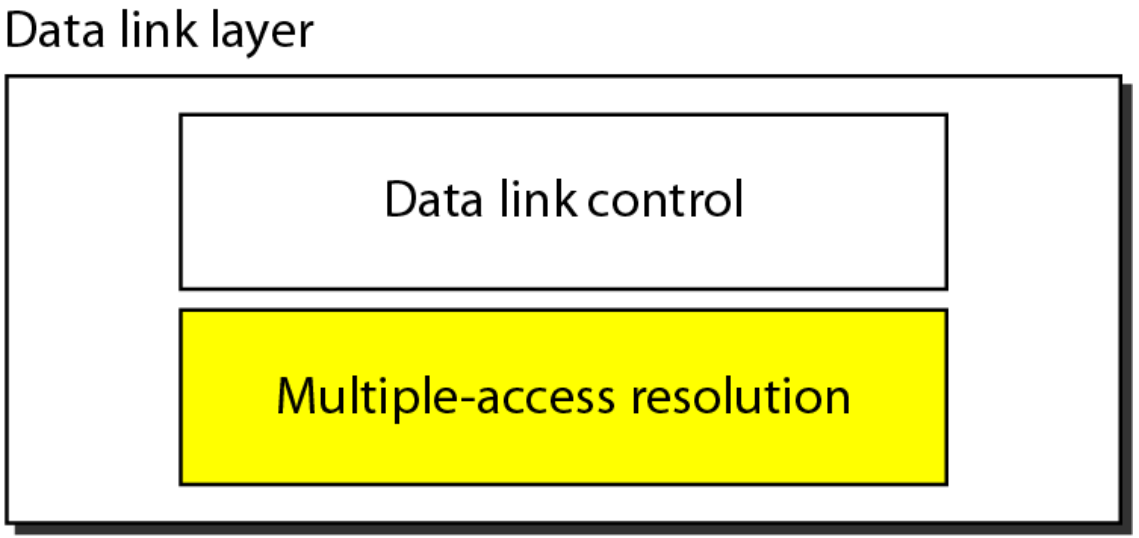
Forouzan

Chapter 12

Multiple Access

- Till now we assumed that there is an available dedicated link (or channel) between the sender and the receiver
- Ex: if we use our cellular phone to connect to another cellular phone, the channel (the band allocated to the vendor company) is not dedicated

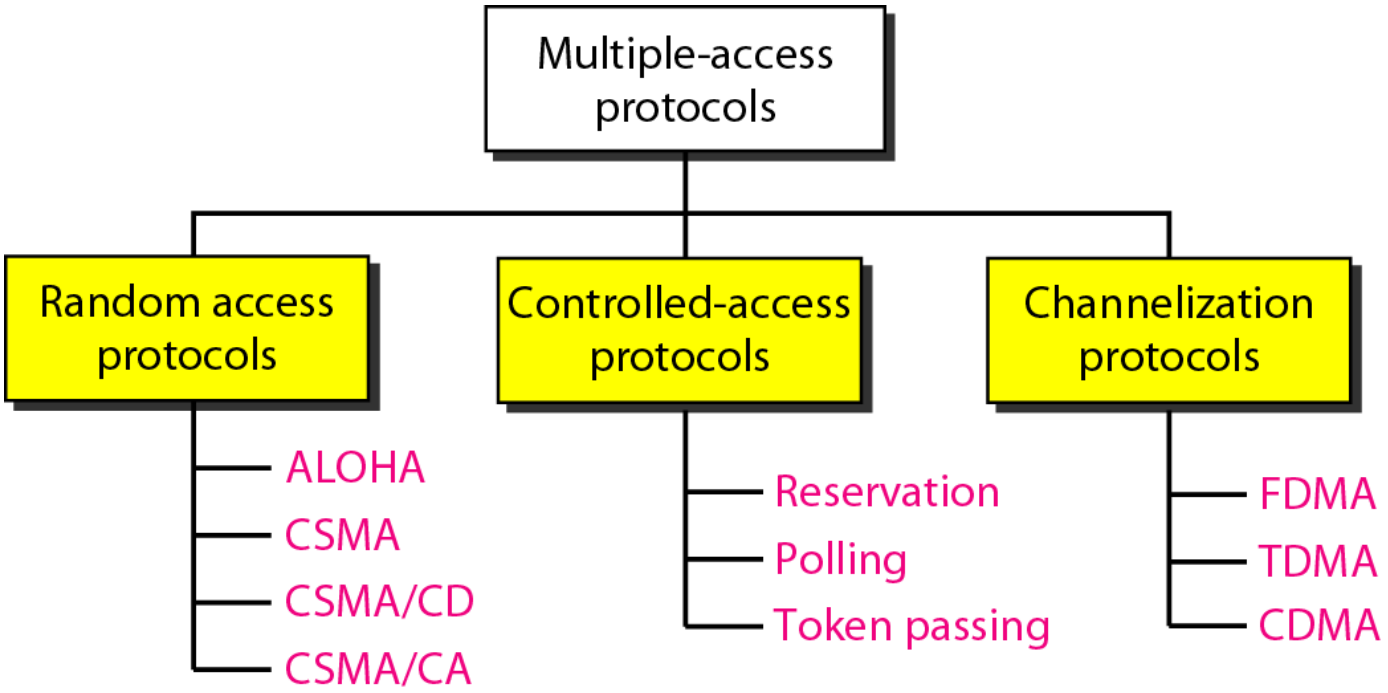
Figure 12.1 *Data link layer divided into two functionality-oriented sublayers*



- 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.
- The problem of controlling the access to the medium is similar to the rules of GD

- right to speak is upheld
- two people do not speak at the same time
- do not interrupt each other
- do not monopolize the discussion

Figure 12.2 *Taxonomy of multiple-access protocols discussed in this chapter*



12-1 RANDOM ACCESS

In random access or contention methods, no station is superior to another station and none is assigned the control over another. No station permits, or does not permit, another station to send. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send.

Topics discussed in this section:

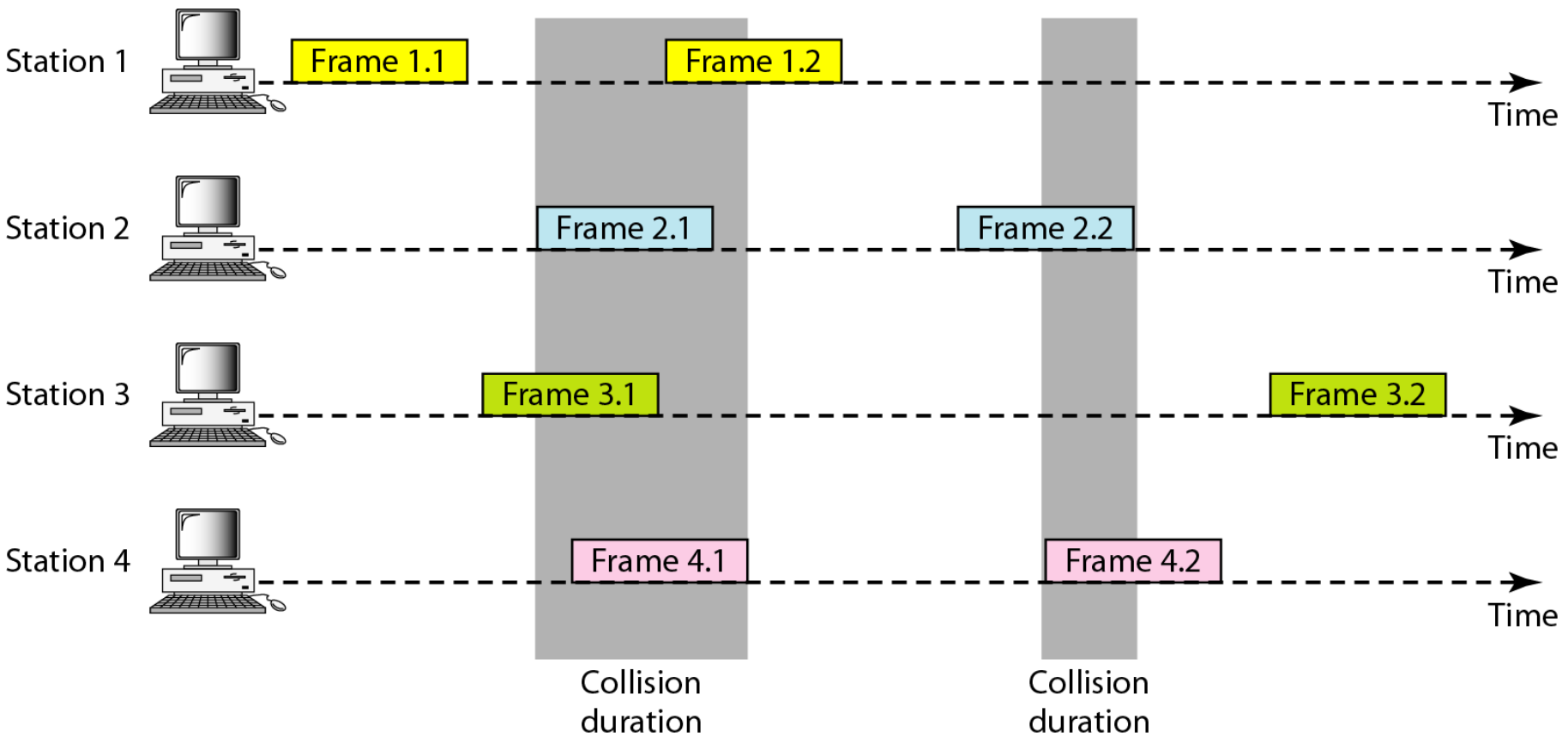
ALOHA
Carrier Sense Multiple Access
Carrier Sense Multiple Access with Collision Detection
Carrier Sense Multiple Access with Collision Avoidance

In a random access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict-collision!!!!!!

Pure ALOHA

- "Each station sends a frame whenever it has a frame to send"
- However, since there is only one channel to share, there is the possibility of collision between frames from different stations

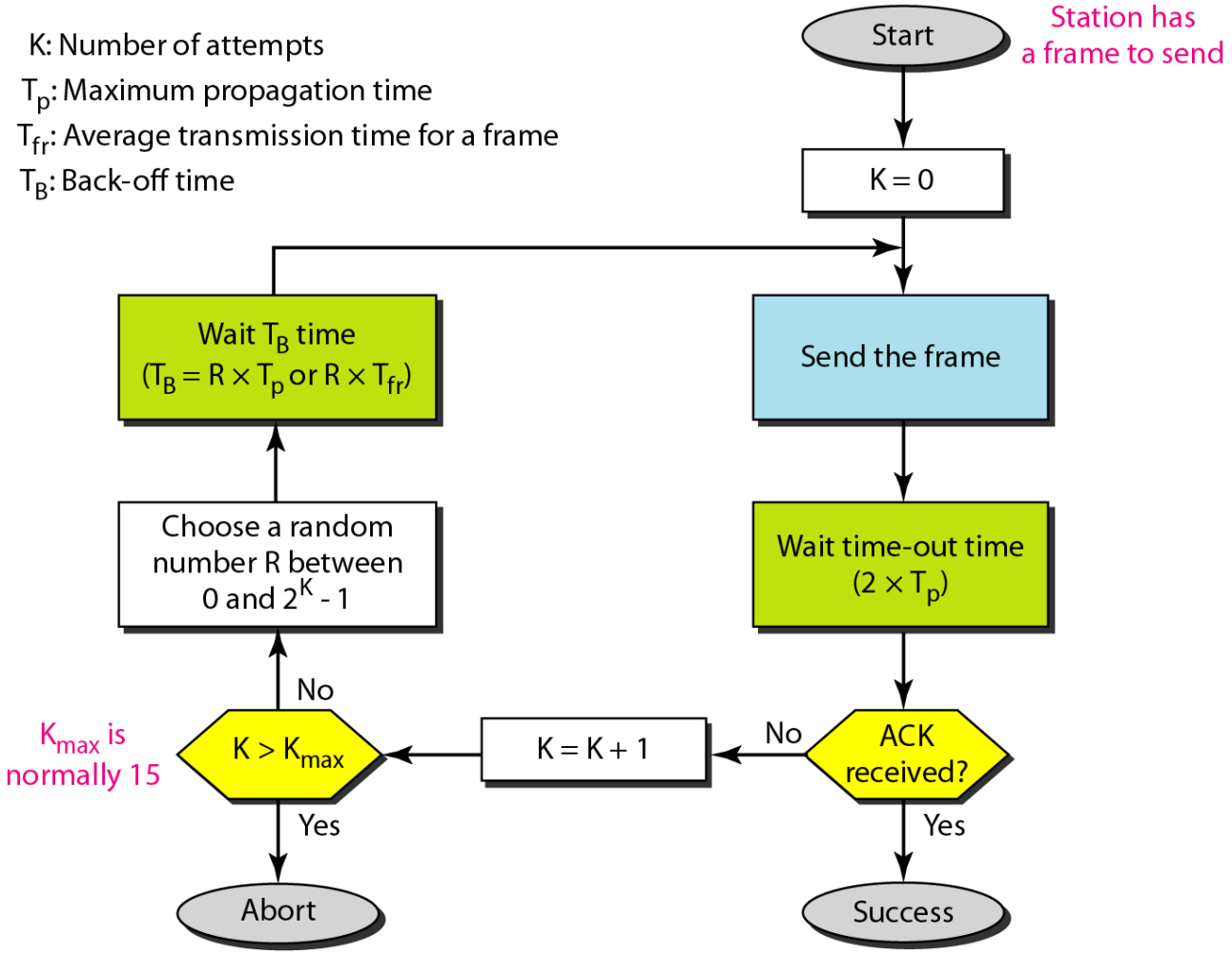
Figure 12.3 *Frames in a pure ALOHA network*



Mechanism

- How to decide resend?
  - If ACK doesn't arrive after time out, resend
- How to handle simultaneous resending of frames from more than on station?
  - Wait random amount of time before resending (back off time)
- How to handle congestion caused b retransmission?
  - After max number of attempts station gives up

Figure 12.4 *Procedure for pure ALOHA protocol*



Example 12.1

The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at 3×10^8 m/s, we find

$$T_p = (600 \times 10^5) / (3 \times 10^8) = 2 \text{ ms}.$$

Now we can find the value of T_B for different values of K =1,2,3

a. For K = 1, the range is {0, 1}. The station needs to generate a random number with a value of 0 or 1. This means that T_B is either 0 ms (0 × 2) or 2 ms (1 × 2), based on the outcome of the random variable.

Example 12.1 (continued)

b. *For K = 2, the range is {0, 1, 2, 3}. This means that T_B can be 0, 2, 4, or 6 ms, based on the outcome of the random variable.*

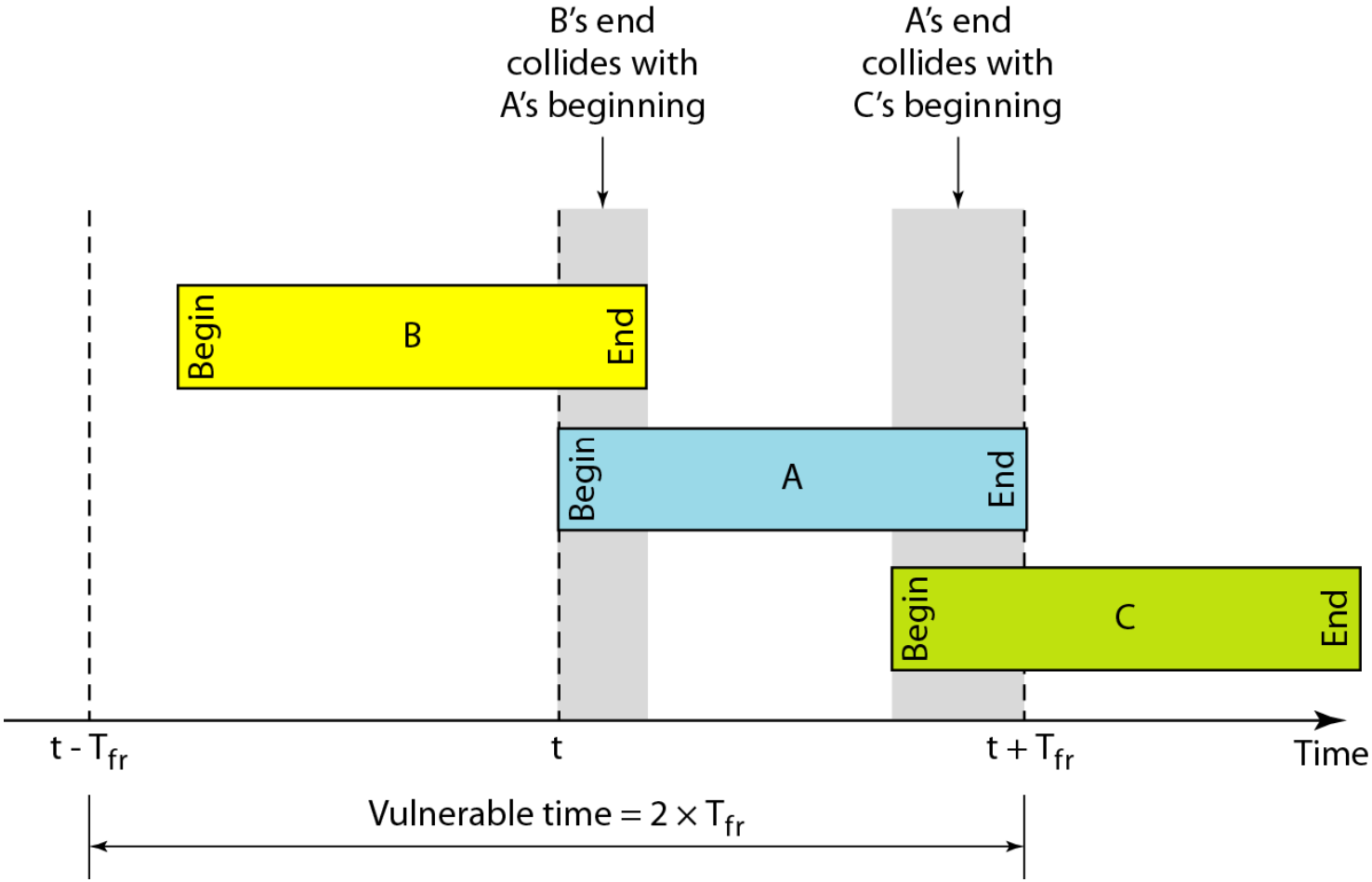
c. *For K = 3, the range is {0, 1, 2, 3, 4, 5, 6, 7}. This means that T_B can be 0, 2, 4, . . . , 14 ms, based on the outcome of the random variable.*

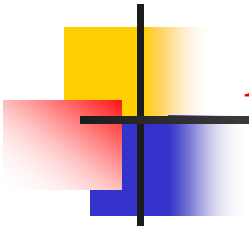
d. *We need to mention that if K > 10, it is normally set to 10.*

Vulnerable time

- 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 *Tfr* Second to send.

Figure 12.5 *Vulnerable time for pure ALOHA protocol*





Example 12.2

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

Average frame transmission time T_{fr} is 200 bits/200 kbps or 1 ms. The vulnerable time is $2 \times 1 \text{ ms} = 2 \text{ ms}$. This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the one 1-ms period that this station is sending.

Note

The throughput for pure ALOHA is $S = G \times e^{-2G}$.

The maximum throughput $S_{max} = 0.184$ when G= (1/2).

Example 12.3

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is 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

The frame transmission time is 200/200 kbps or 1 ms.

a. If the system creates 1000 frames per second, this is 1 frame per millisecond. The load is 1. In this case $S = G \times e^{-2G}$ *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.*

Example 12.3 (continued)

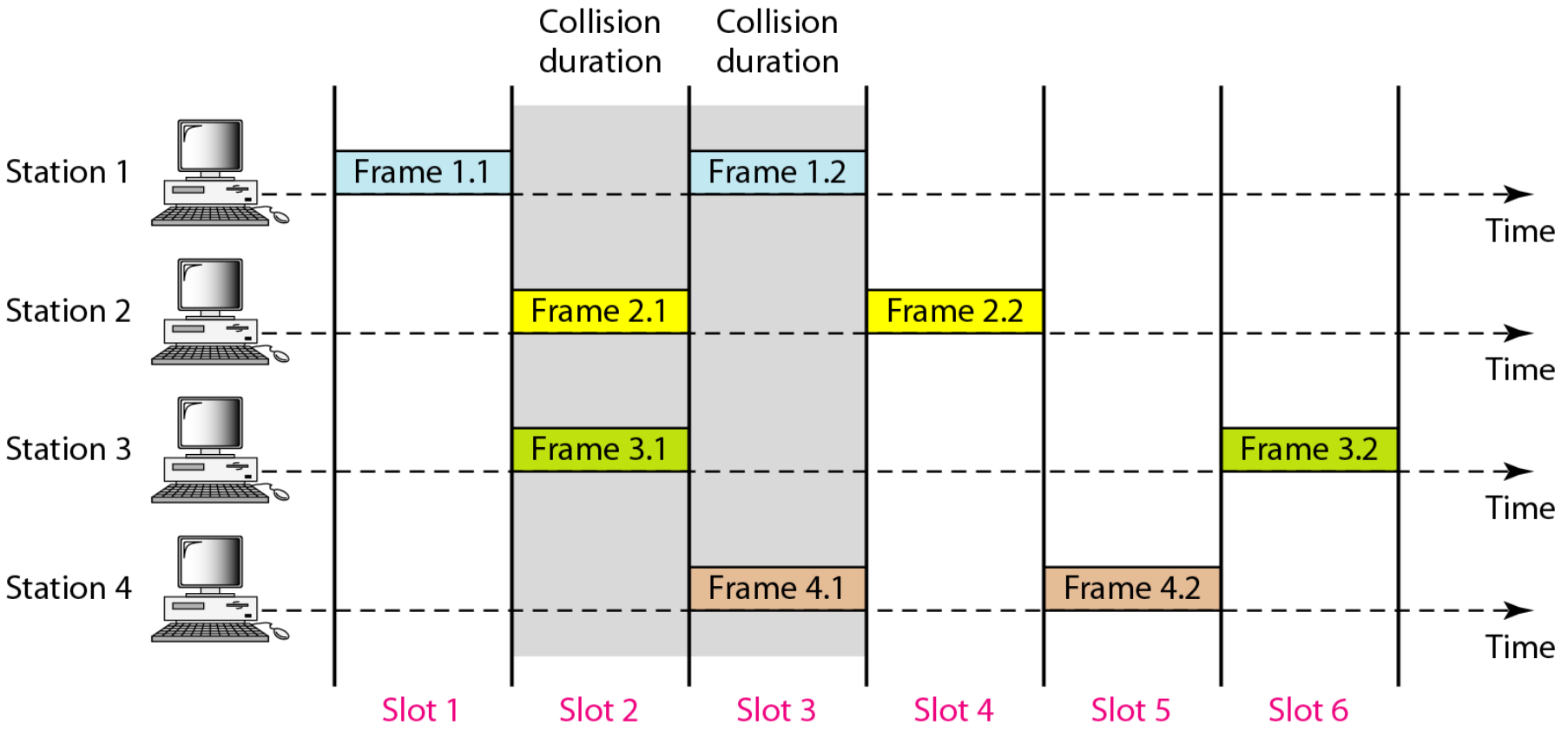
b. *If the system creates 500 frames per second, this is (1/2) frame per millisecond. The load is (1/2). In this case* $S = G \times e^{-2G}$ *or* $S = 0.184$ *(18.4 percent). This means that the throughput is* $500 \times 0.184 = 92$ *and that only 92 frames out of 500 will probably survive. Note that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

c. *If the system creates 250 frames per second, this is (1/4) frame per millisecond. The load is (1/4). In this case* $S = G \times e^{-2G}$ *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.*

Slotted ALOHA

- To improve the efficiency of pure ALOHA
- In slotted ALOHA we divide the time into slots of Tfr s and force the station to send only at the beginning of the time slot.

Figure 12.6 *Frames in a slotted ALOHA network*

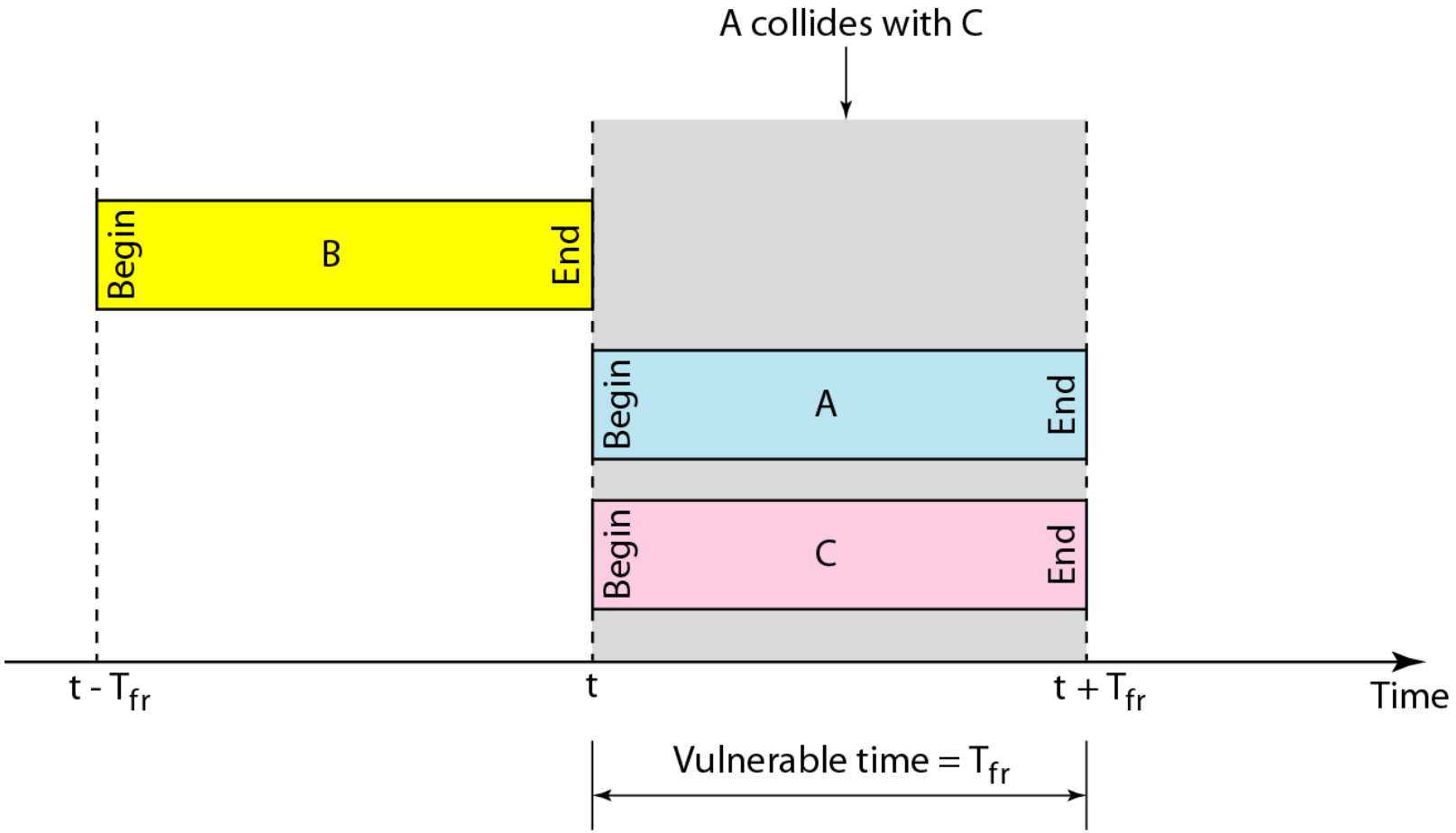


Note

The throughput for slotted ALOHA is $S = G \times e^{-G}$.

The maximum throughput $S_{max} = 0.368$ when G = 1.

Figure 12.7 *Vulnerable time for slotted ALOHA protocol*



Example 12.4

A slotted ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is 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

The frame transmission time is 200/200 kbps or 1 ms.

a. If the system creates 1000 frames per second, this is 1 frame per millisecond. The load is 1. In this case $S = G \times e^{-G}$ *or* $S = 0.368$ *(36.8 percent). This means that the throughput is* $1000 \times 0.0368 = 368$ *frames. Only 386 frames out of 1000 will probably survive.*

Example 12.4 (continued)

b. *If the system creates 500 frames per second, this is (1/2) frame per millisecond. The load is (1/2). In this case* $S = G \times e^{-G}$ *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. *If the system creates 250 frames per second, this is (1/4) frame per millisecond. The load is (1/4). In this case* $S = G \times e^{-G}$ *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.*

CSMA

- Sense before transmit
- Reduces possibility of collision
- The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time for the first bit to reach every station

Figure 12.8 *Space/time model of the collision in CSMA*

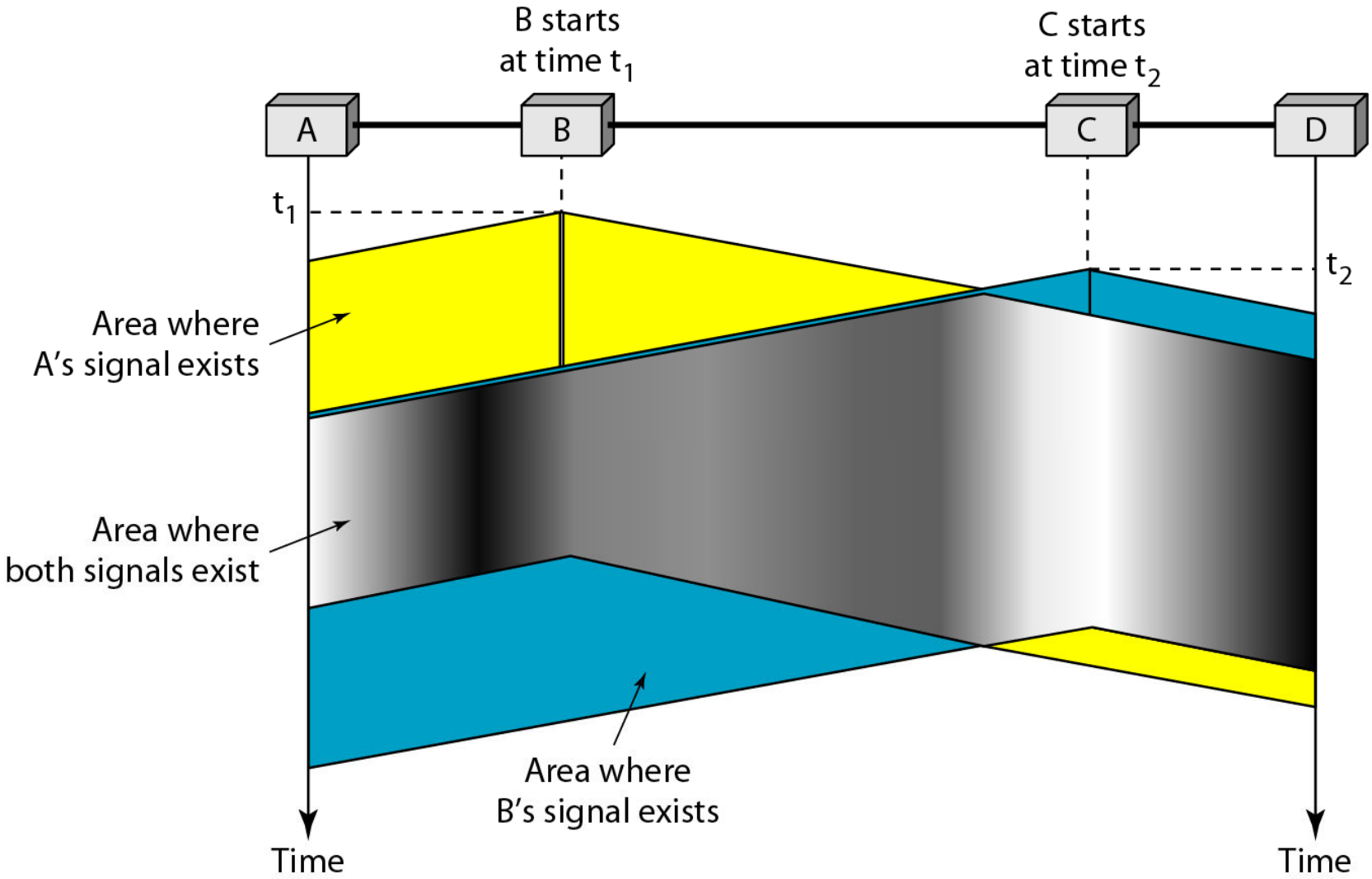
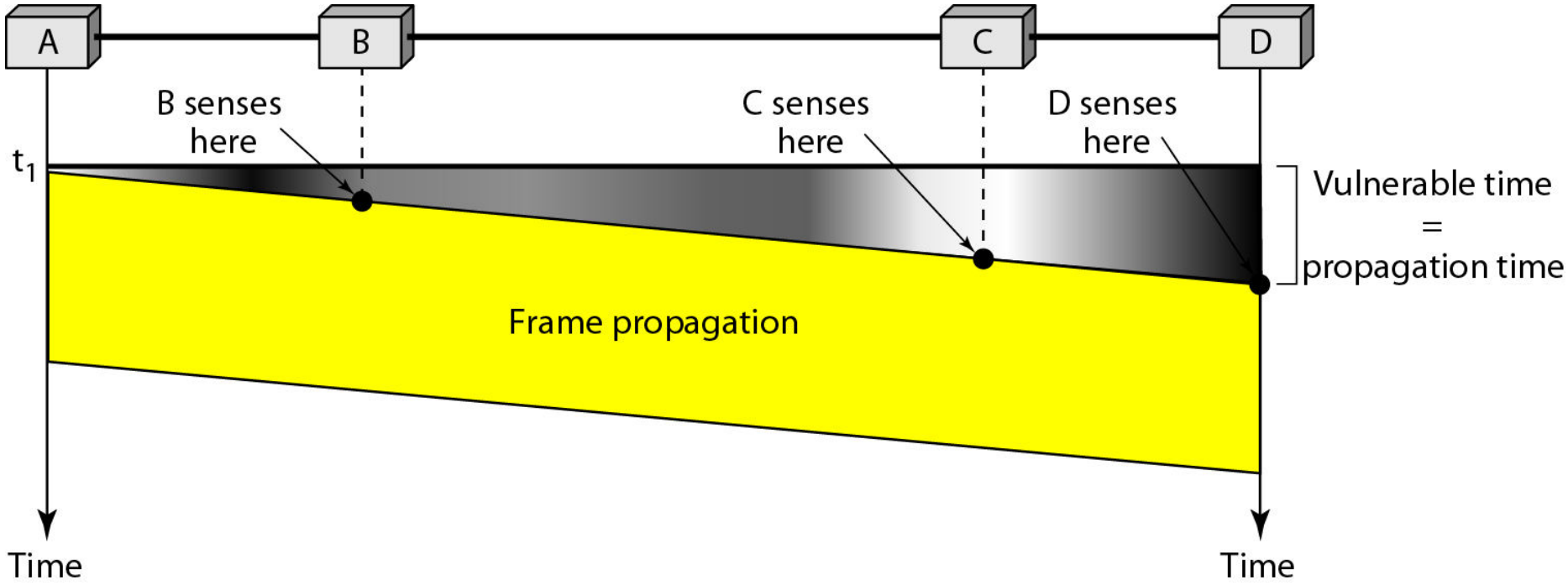


Figure 12.9 *Vulnerable time in CSMA*

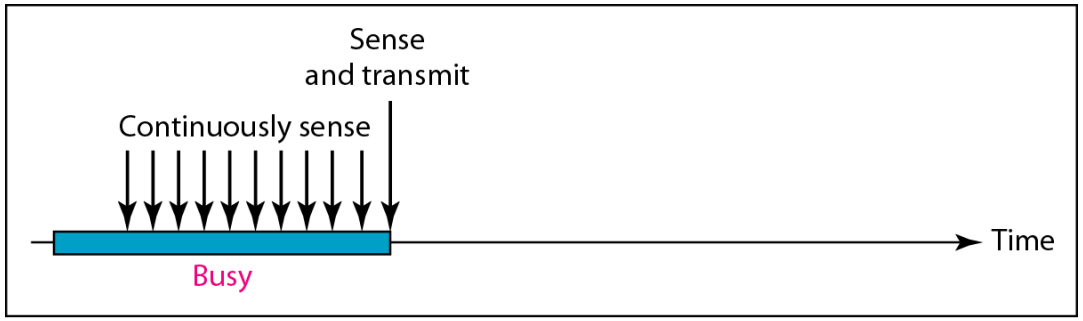


The vulnerable time for CSMA is the propagation time *Tp* . This is the time needed for a signal to propagate from one end of the medium to the other.

Persistent Method

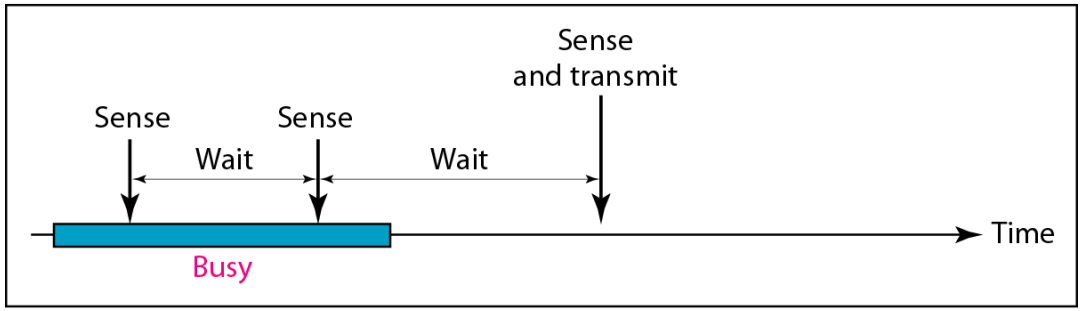
- What should a station do if
- Channel is busy?
- Channel is idle?

Figure 12.10 *Behavior of three persistence methods*



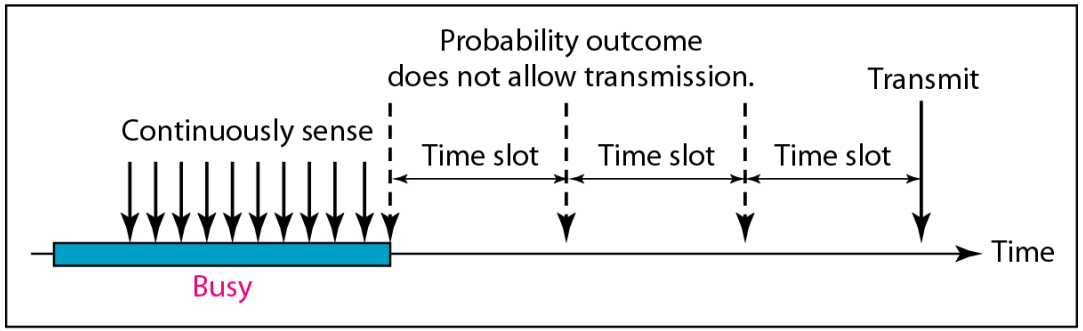
a. 1-persistent

Highest Collision



b. Nonpersistent

Less Collision But reduced efficiency



c. p-persistent

Reduced collision Improved efficiancy

Figure 12.11 *Flow diagram for three persistence methods*

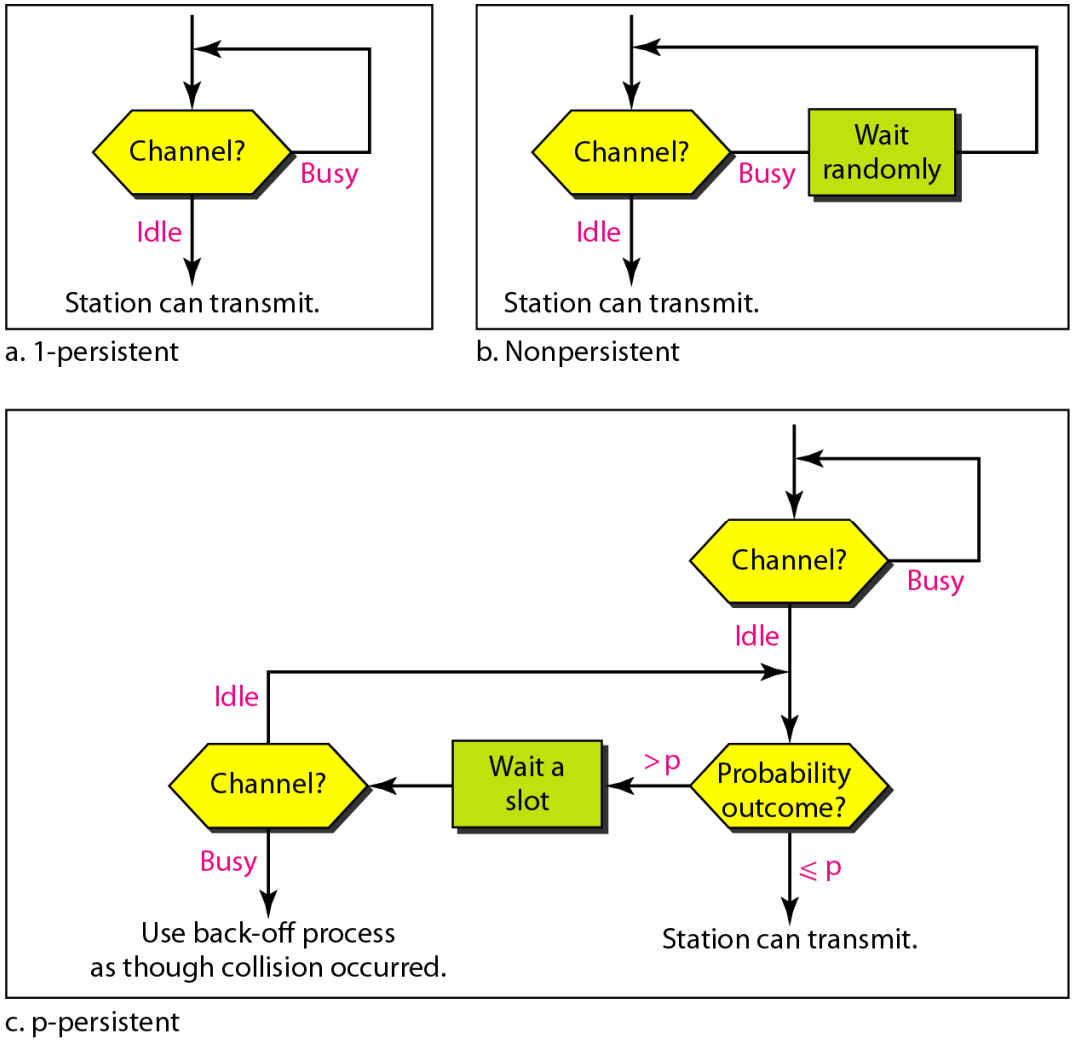


Figure 12.12 *Collision of the first bit in CSMA/CD*

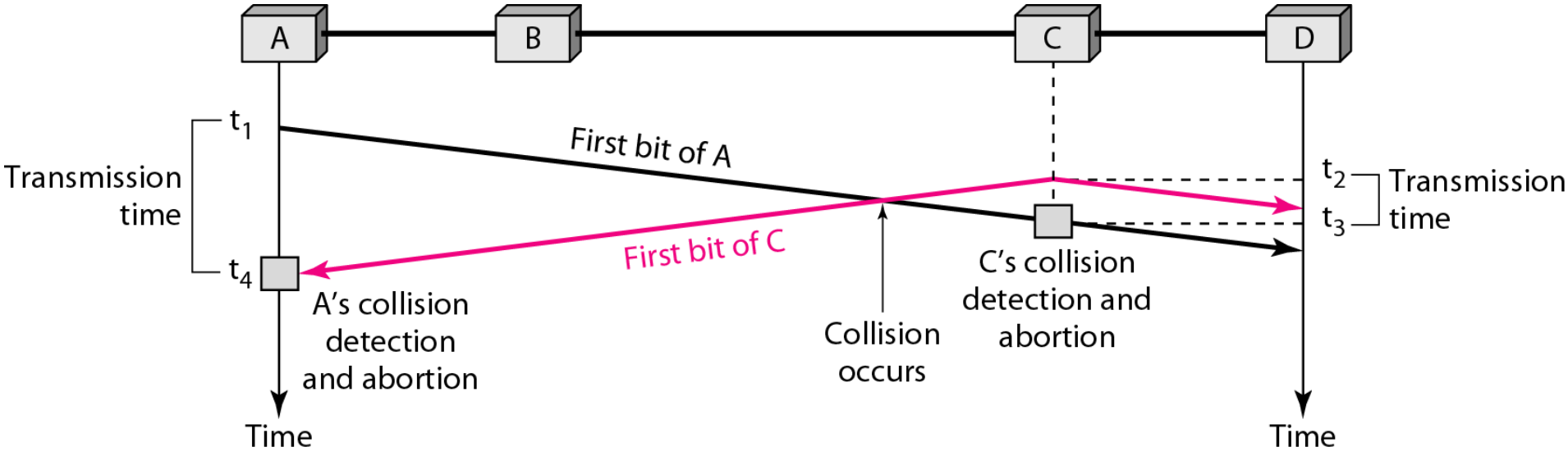
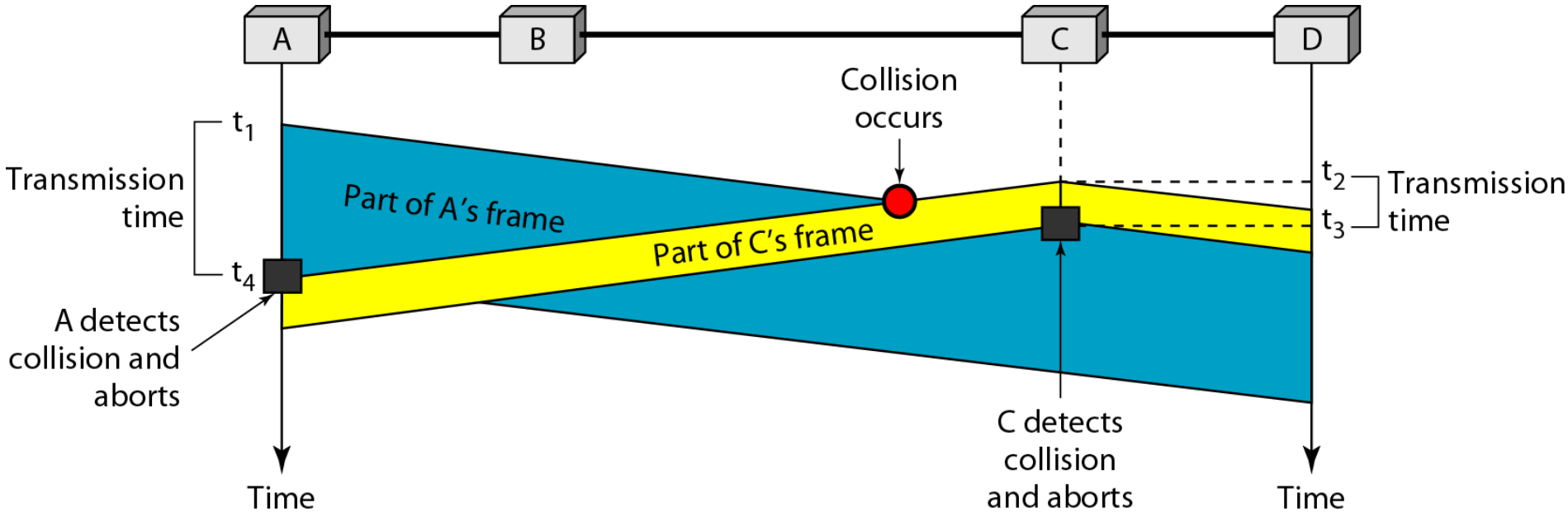


Figure 12.13 *Collision and abortion in CSMA/CD*



Example 12.5

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 μs, what is the minimum size of the frame?

Solution

The frame transmission time is $T_{fr} = 2 \times T_p = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet.

Figure 12.14 *Flow diagram for the CSMA/CD*

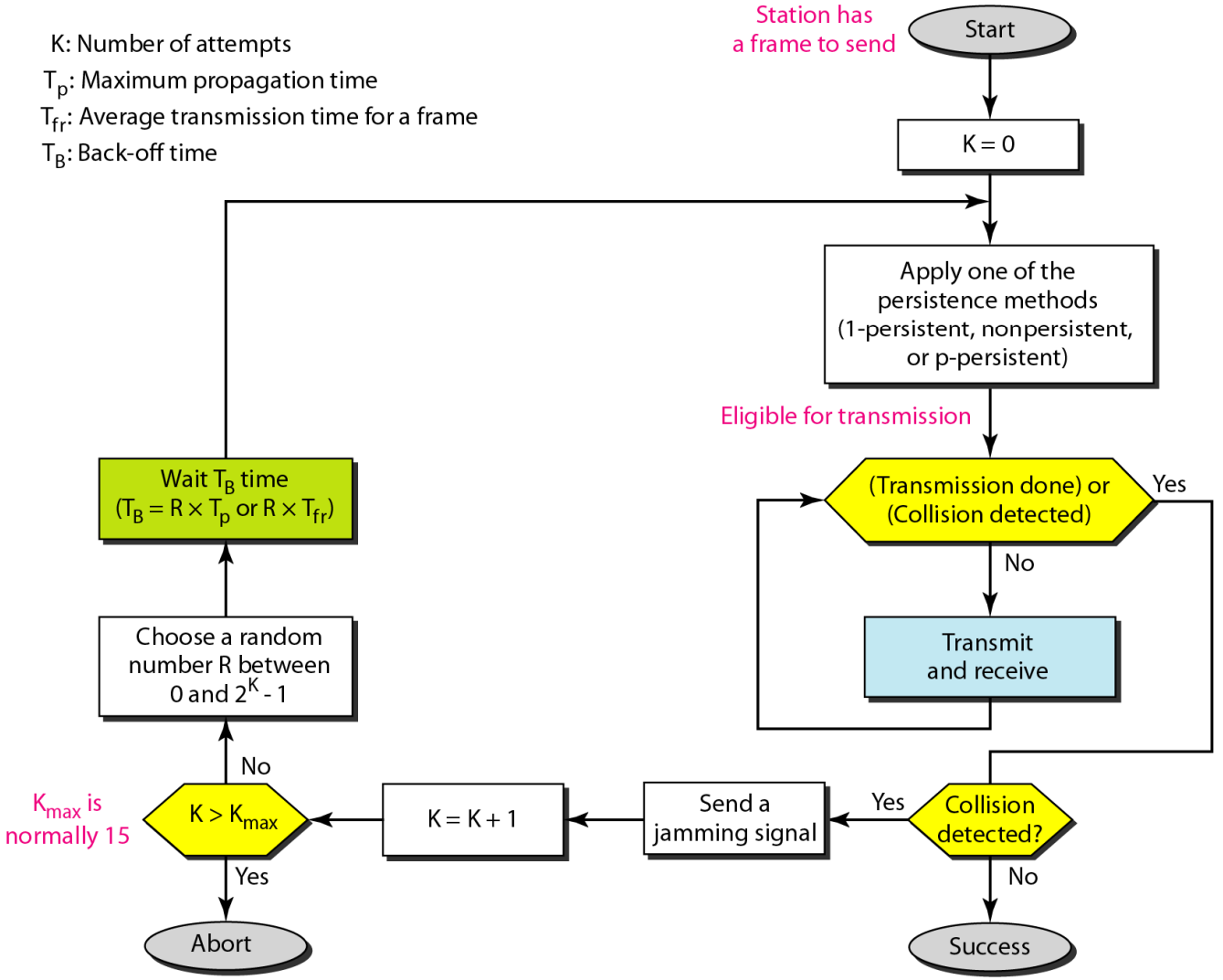


Figure 12.15 *Energy level during transmission, idleness, or collision*

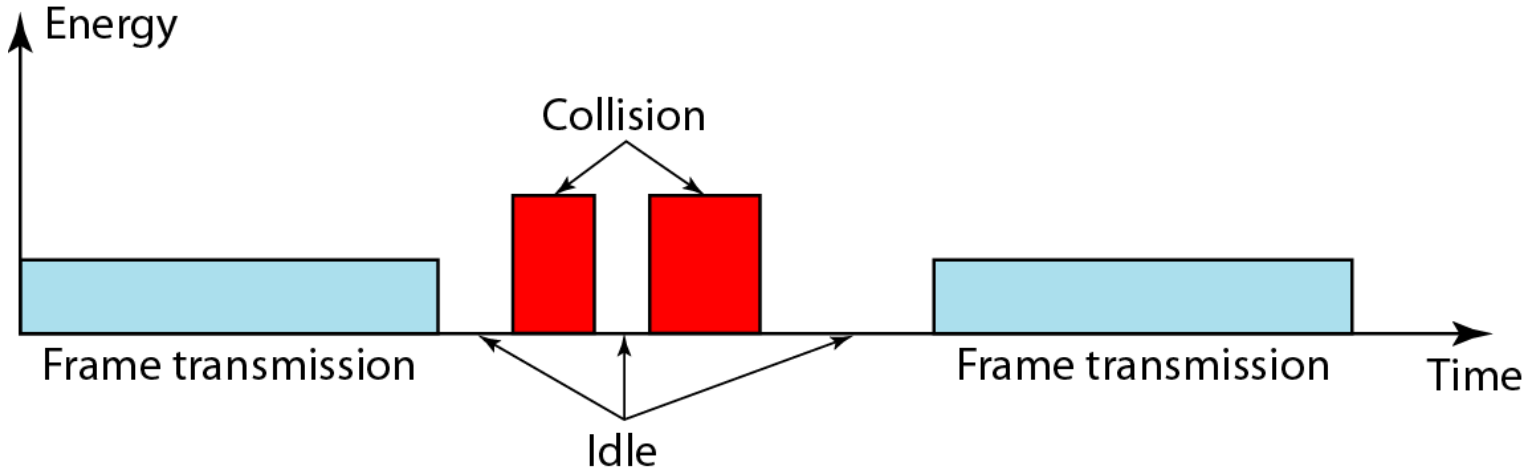
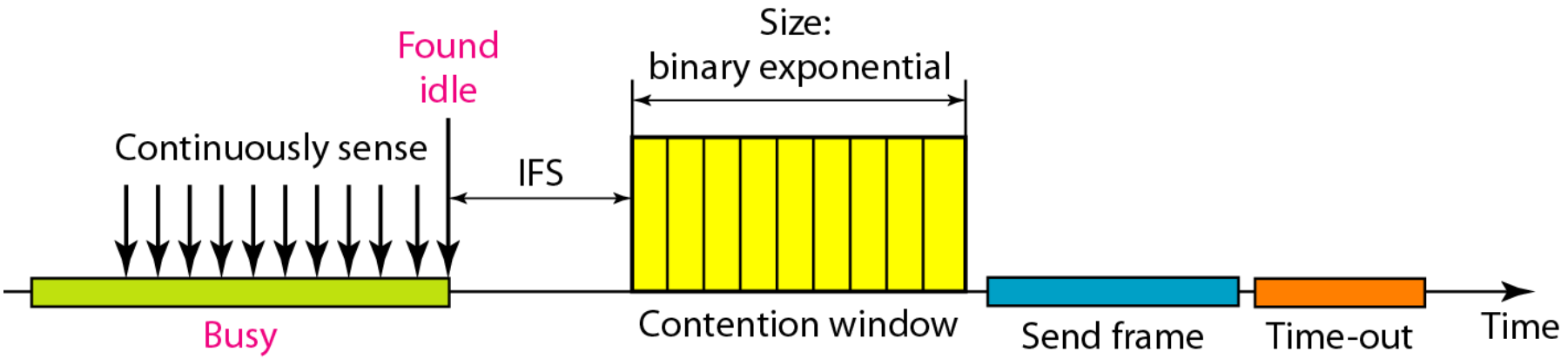


Figure 12.16 *Timing in CSMA/CA*



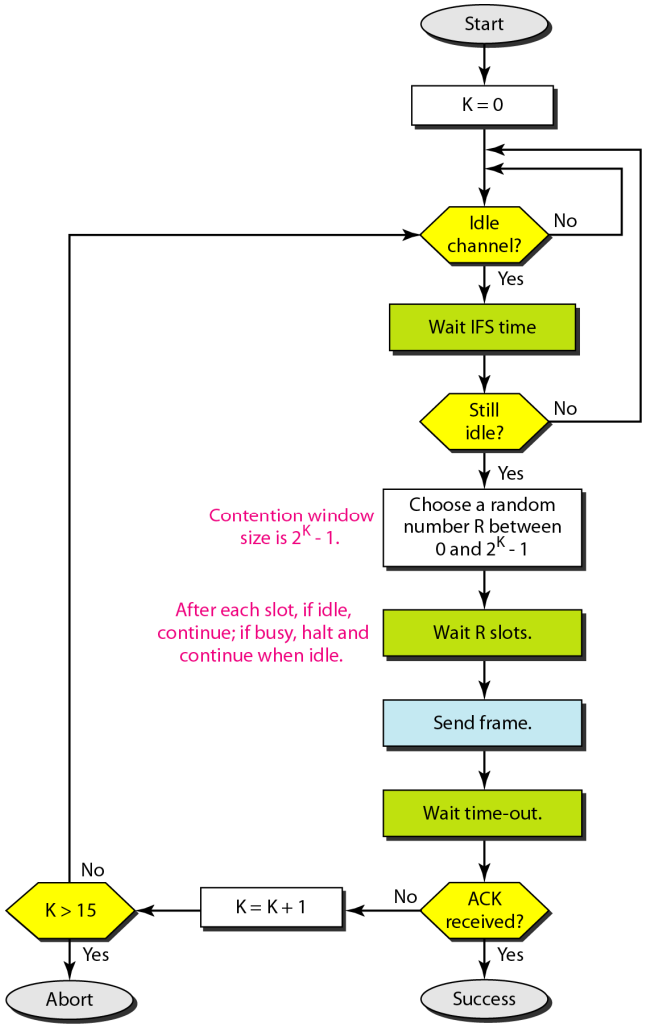
Note

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

Note

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.

Figure 12.17 *Flow diagram for CSMA/CA*



12-2 CONTROLLED ACCESS

In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss three popular controlled-access methods.

Topics discussed in this section:

Reservation
Polling
Token Passing

Figure 12.18 *Reservation access method*

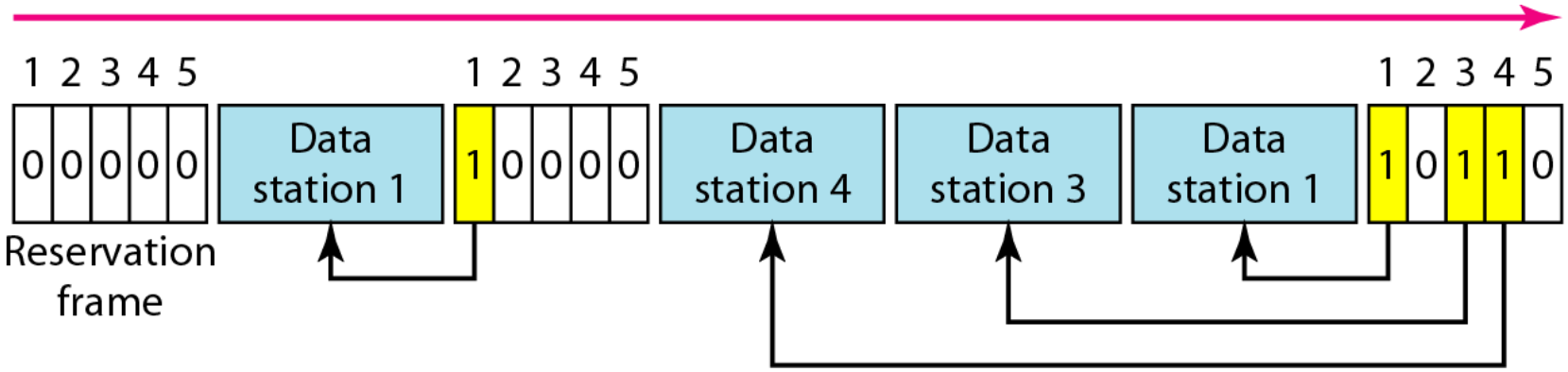


Figure 12.19 *Select and poll functions in polling access method*

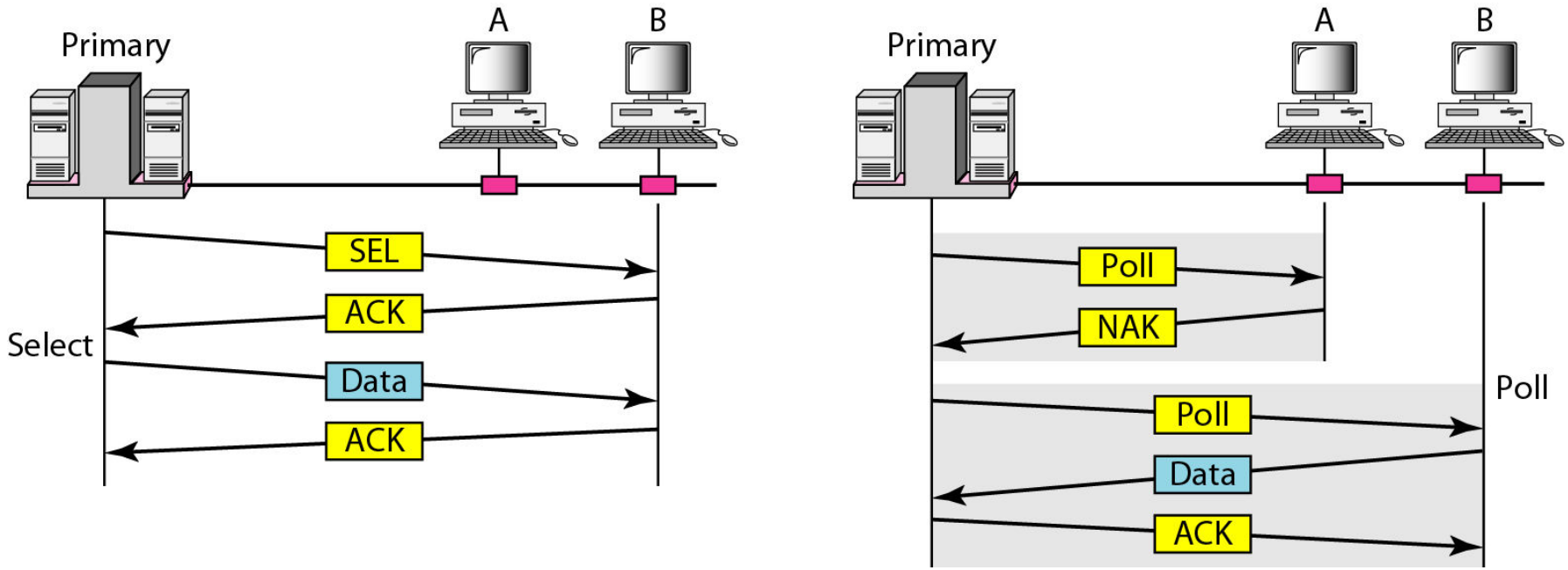
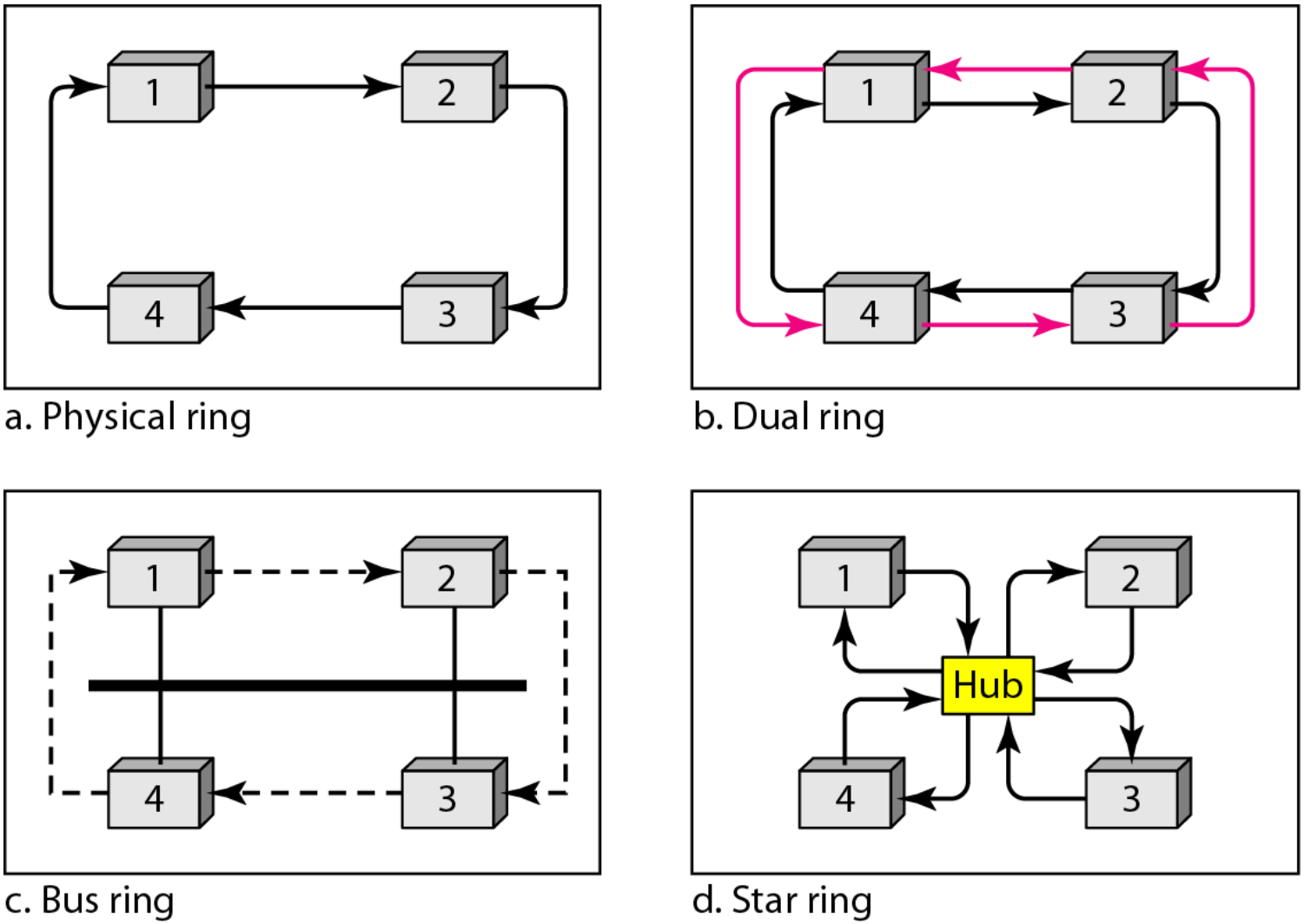
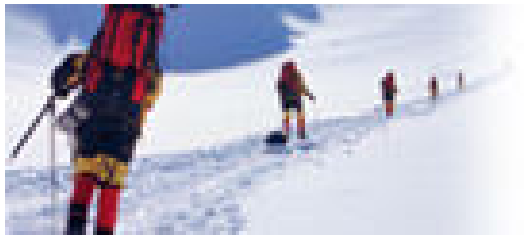


Figure 12.20 *Logical ring and physical topology in token-passing access method*





Data Communications and Networking Fourth Edition

Forouzan

Chapter 1

Introduction

1-1 DATA COMMUNICATIONS

The term telecommunication means communication at a distance. The word data refers to information presented in whatever form is agreed upon by the parties creating and using the data. Data communications are the exchange of data between two devices via some form of transmission medium such as a wire cable.

Topics discussed in this section:

- **Components of a data communications system**
- **Data Flow**

Characteristic of Data Communication System

- 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
- Accuracy: The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable
- Timeliness: The system must deliver data in a timely manner. Data delivered late are useless
- Jitter: Jitter refers to the variation in the packet arrival time.

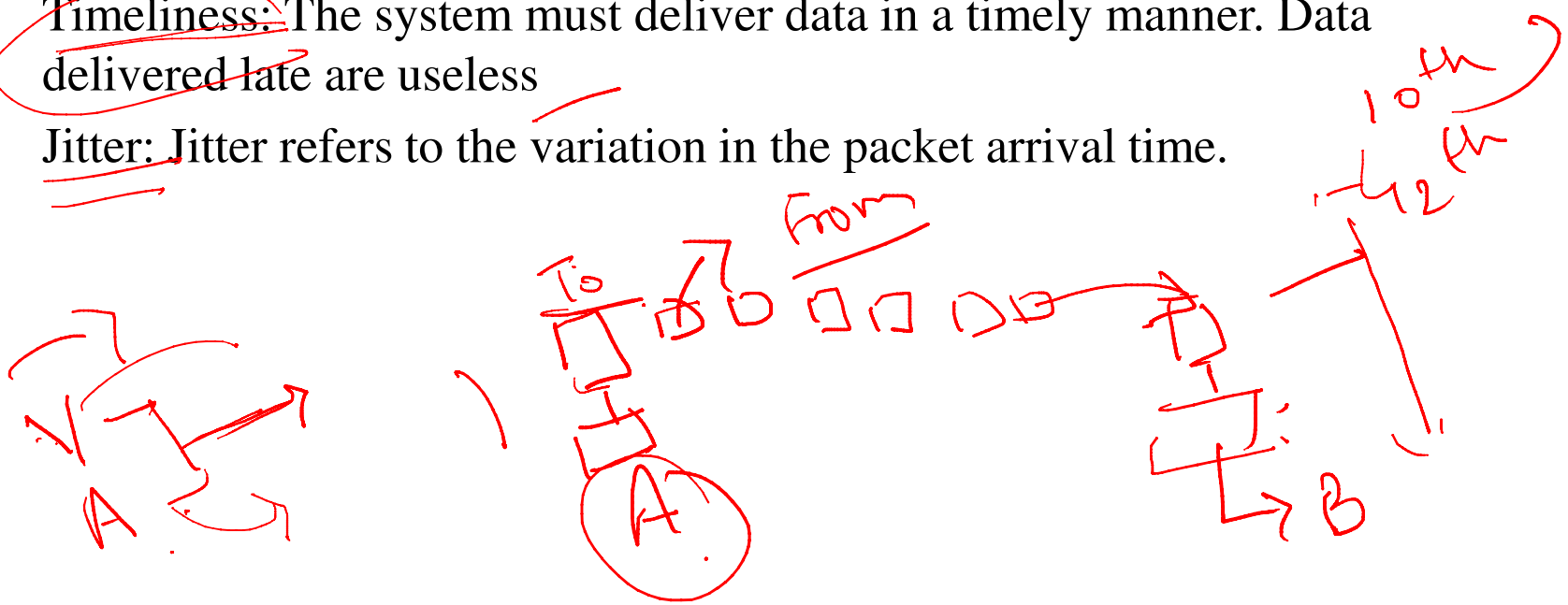


Figure 1.1 *Components of a data communication system*

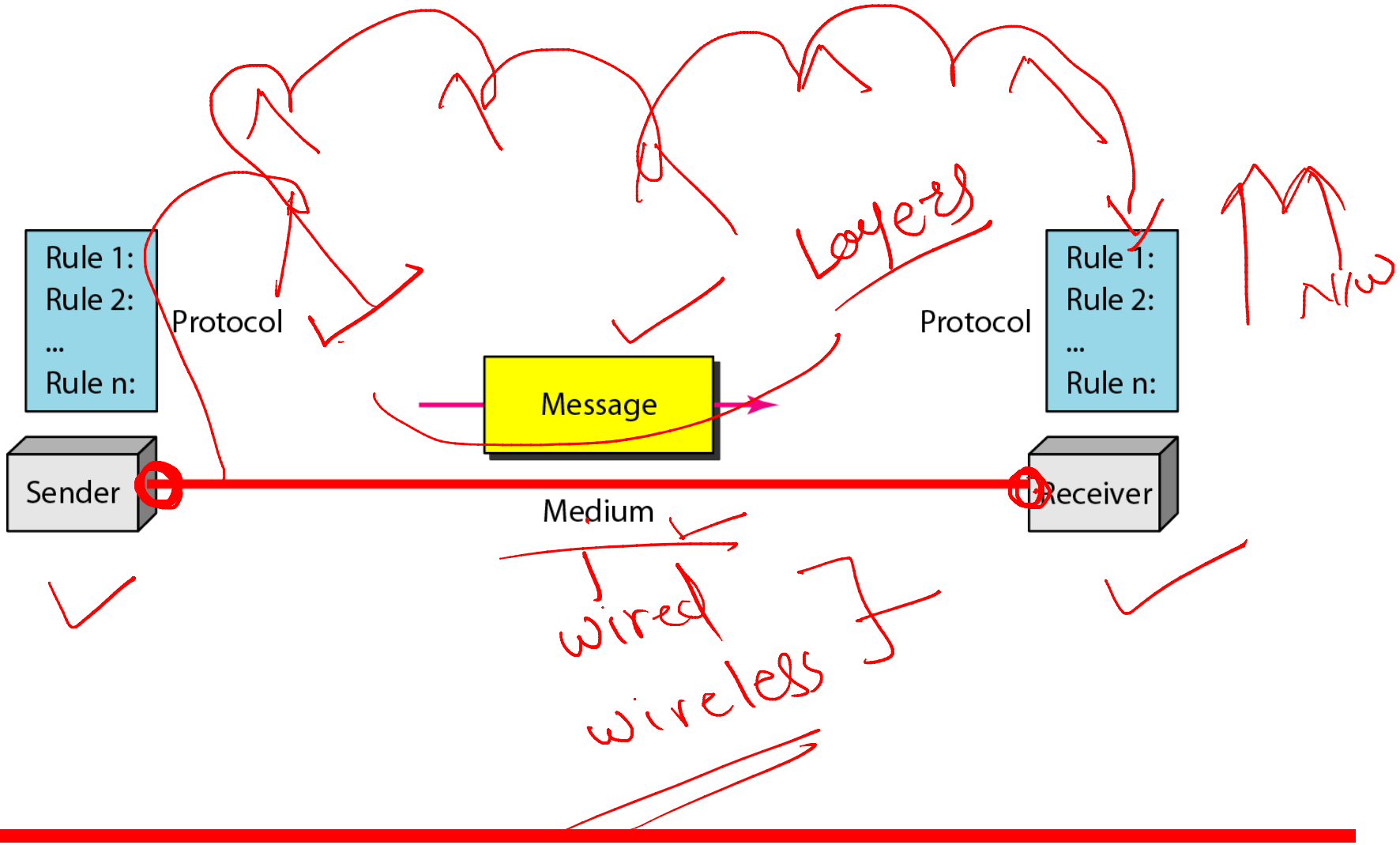
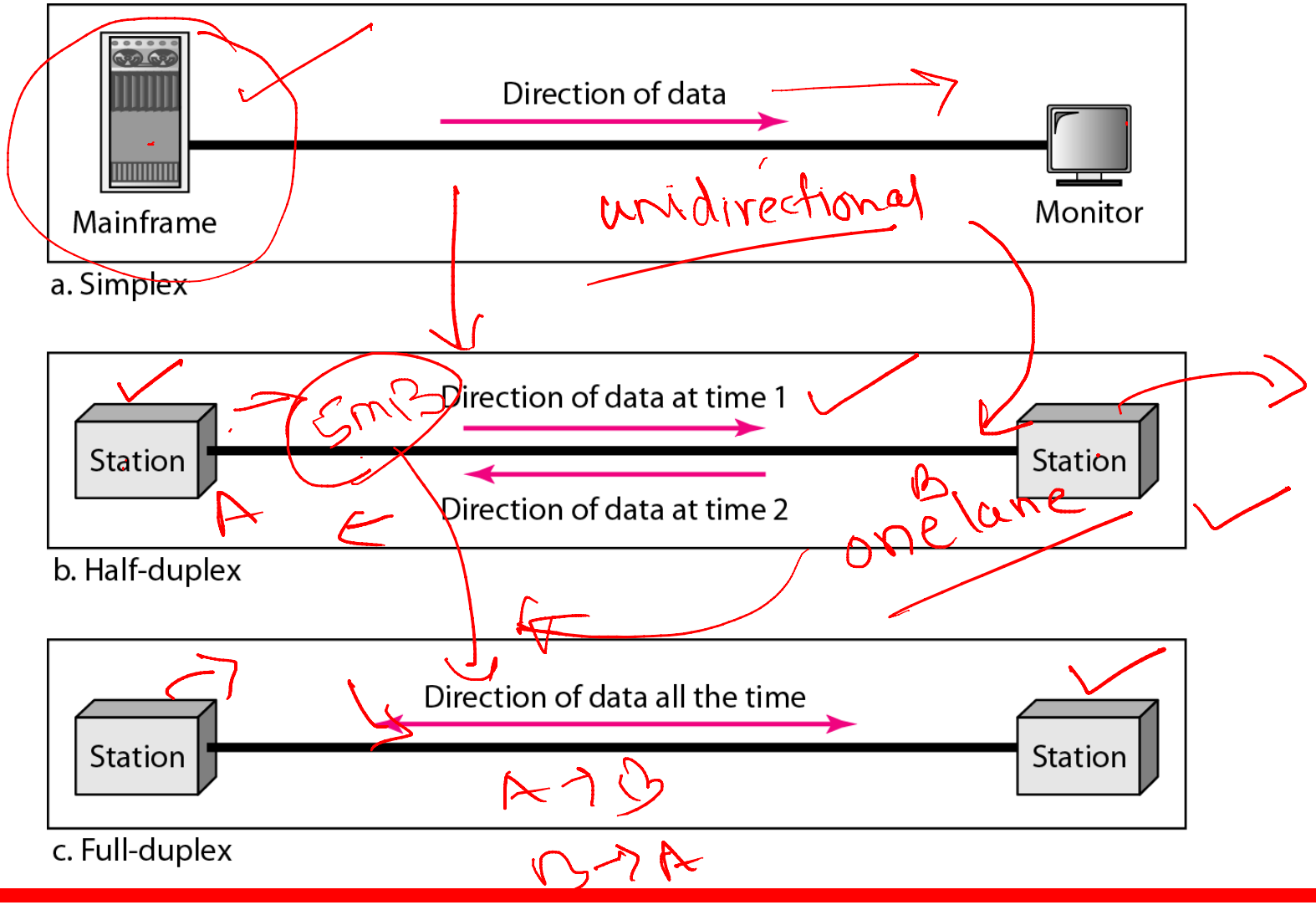


Figure 1.2 *Data flow (simplex, half-duplex, and full-duplex)*



1-2 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. A link can be a cable, air, optical fiber, or any medium which can transport a signal carrying information.

Topics discussed in this section:

- **Network Criteria**
- **Physical Structures**
- **Categories of Networks**

Network Criteria

- Performance
  - Depends on Network Elements
  - Measured in terms of Delay and Throughput
- Reliability
  - Failure rate of network components
  - Measured in terms of availability/robustness
- Security
  - Data protection against corruption/loss of data due to:
    - Errors
    - Malicious users

Physical Structures

- **Type of Connection**
  - Point to Point - single transmitter and receiver
  - Multipoint - multiple recipients of single transmission
- **Physical Topology**
  - Connection of devices
  - Type of transmission - unicast, mulitcast, broadcast

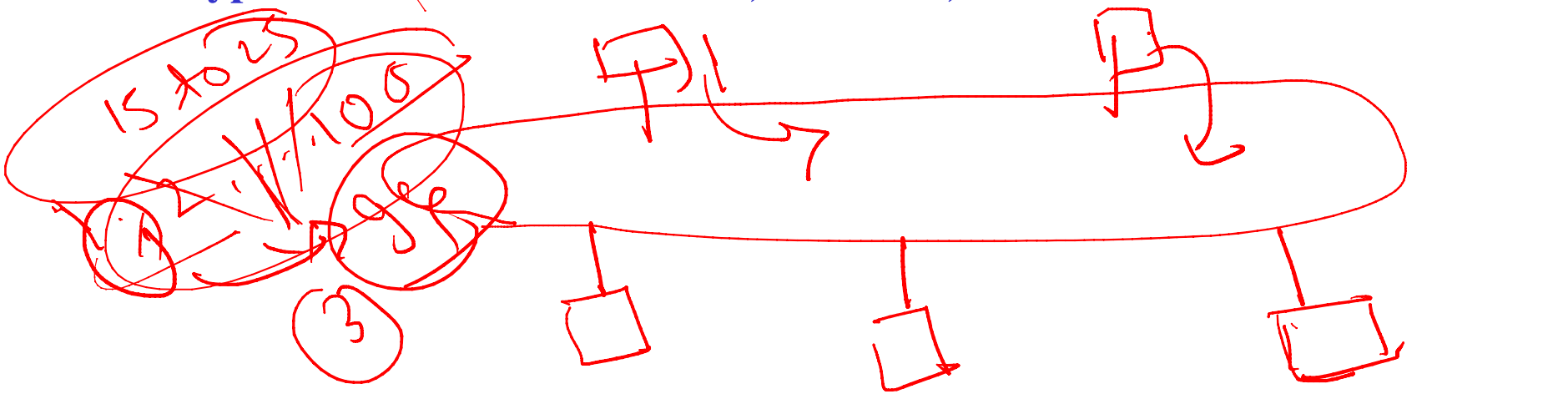
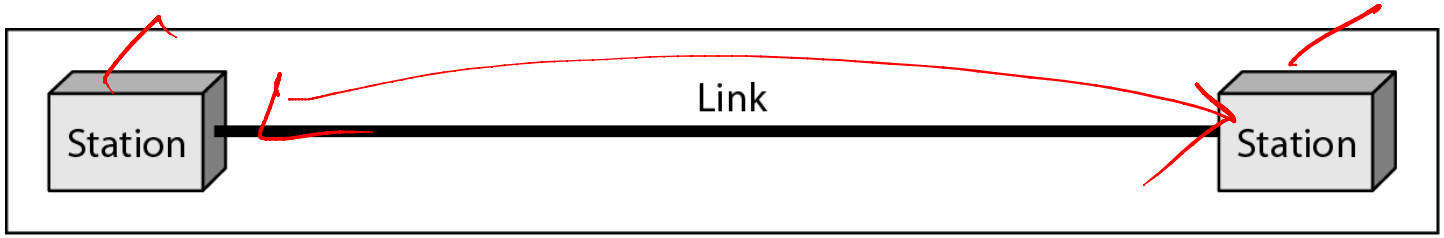
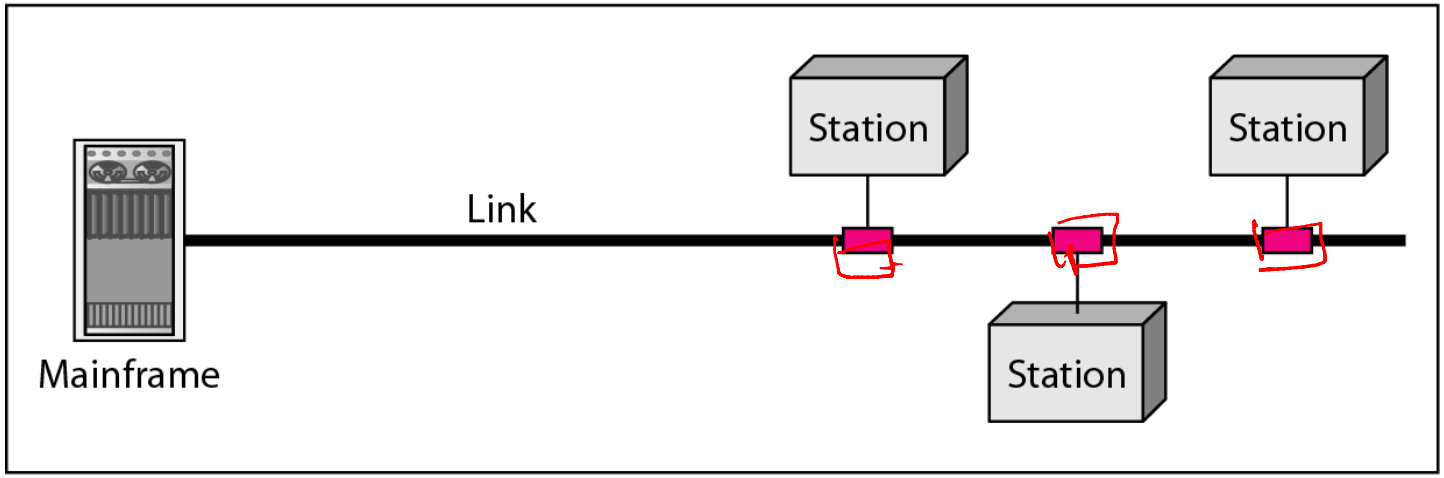


Figure 1.3 *Types of connections: point-to-point and multipoint*



a. Point-to-point



b. Multipoint

Figure 1.4 *Categories of topology*

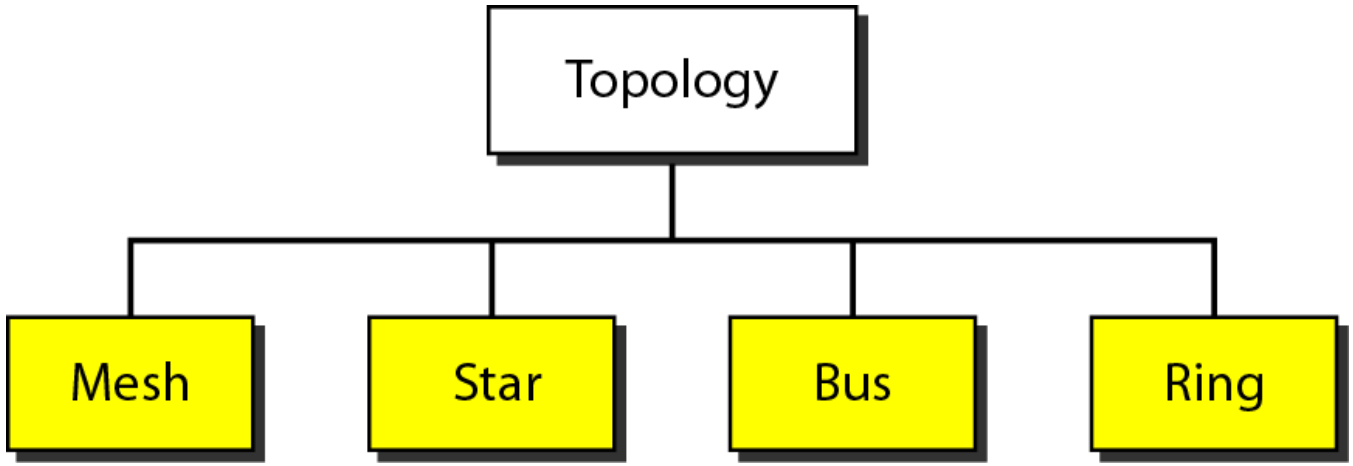


Figure 1.5 *A fully connected mesh topology (five devices)*

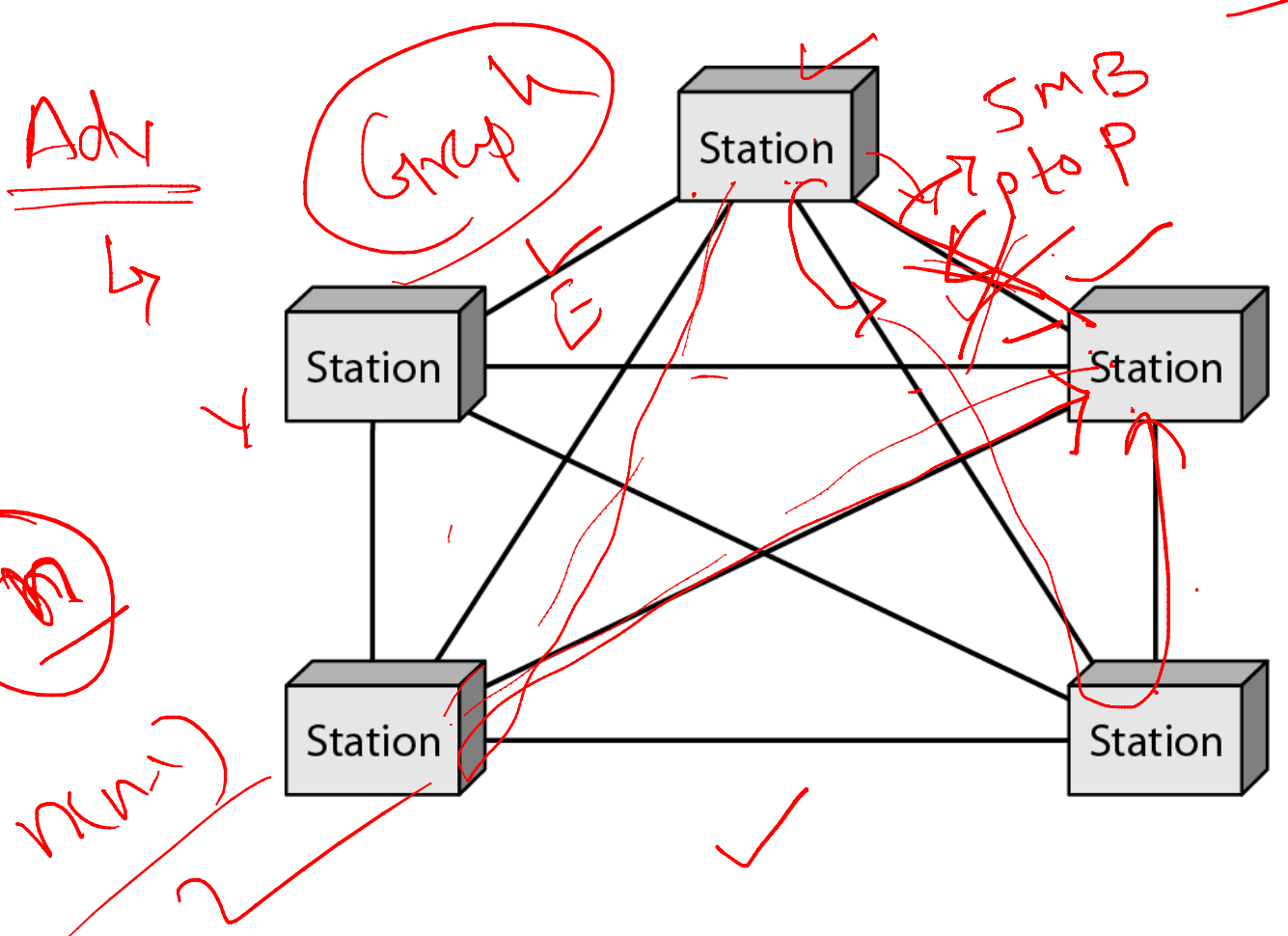


Figure 1.6 *A star topology connecting four stations*

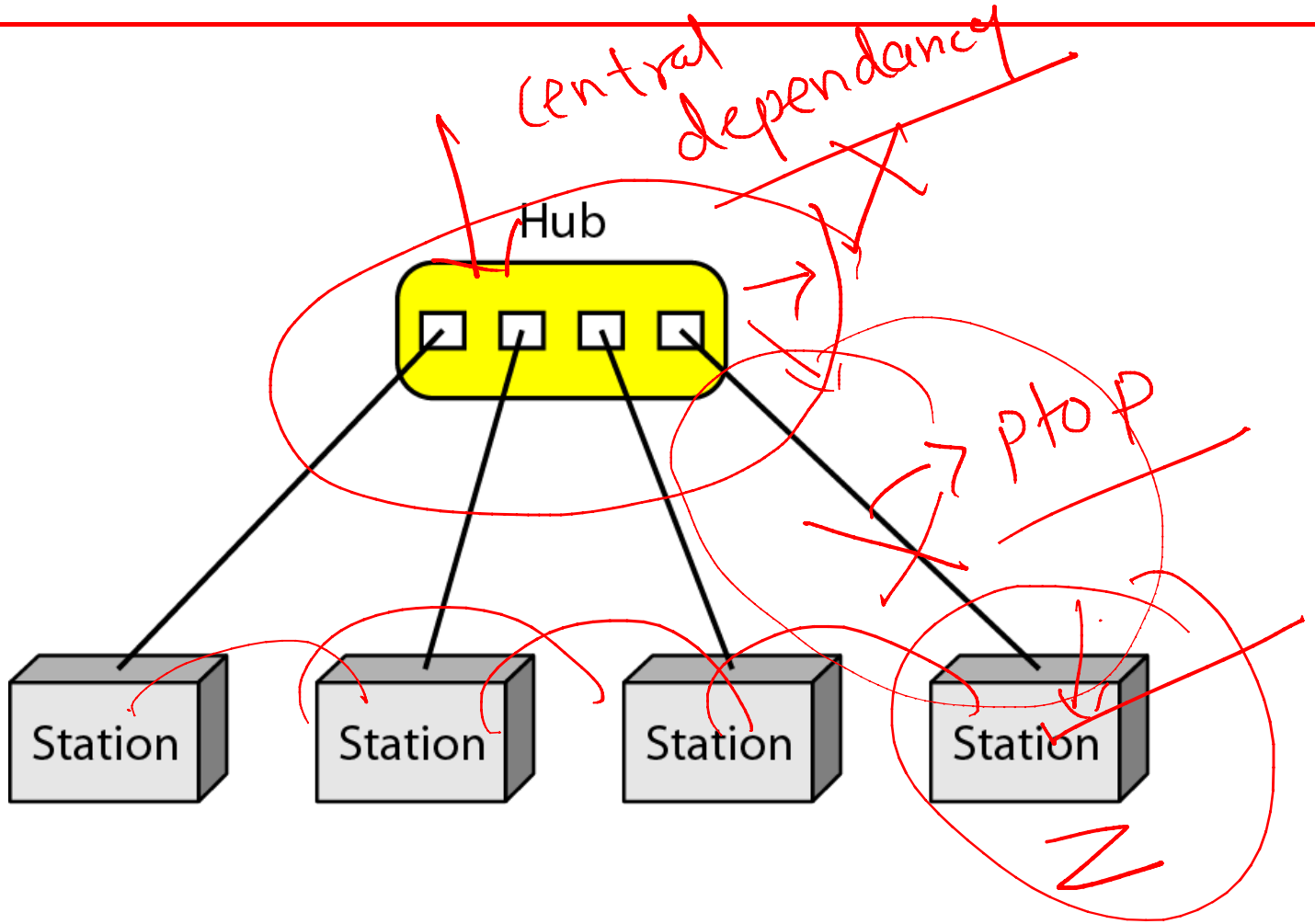


Figure 1.7 *A bus topology connecting three stations*

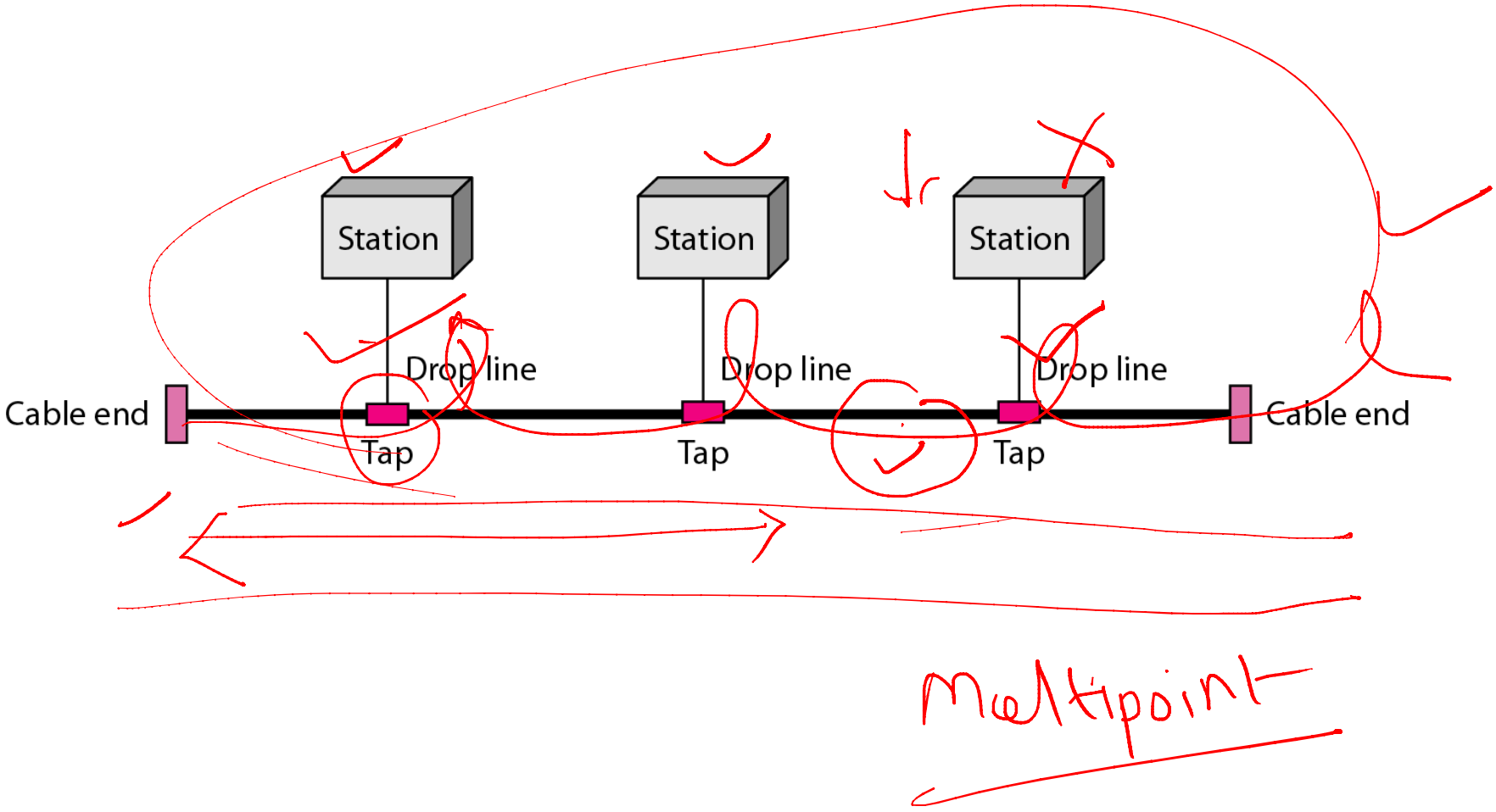


Figure 1.8 *A ring topology connecting six stations*

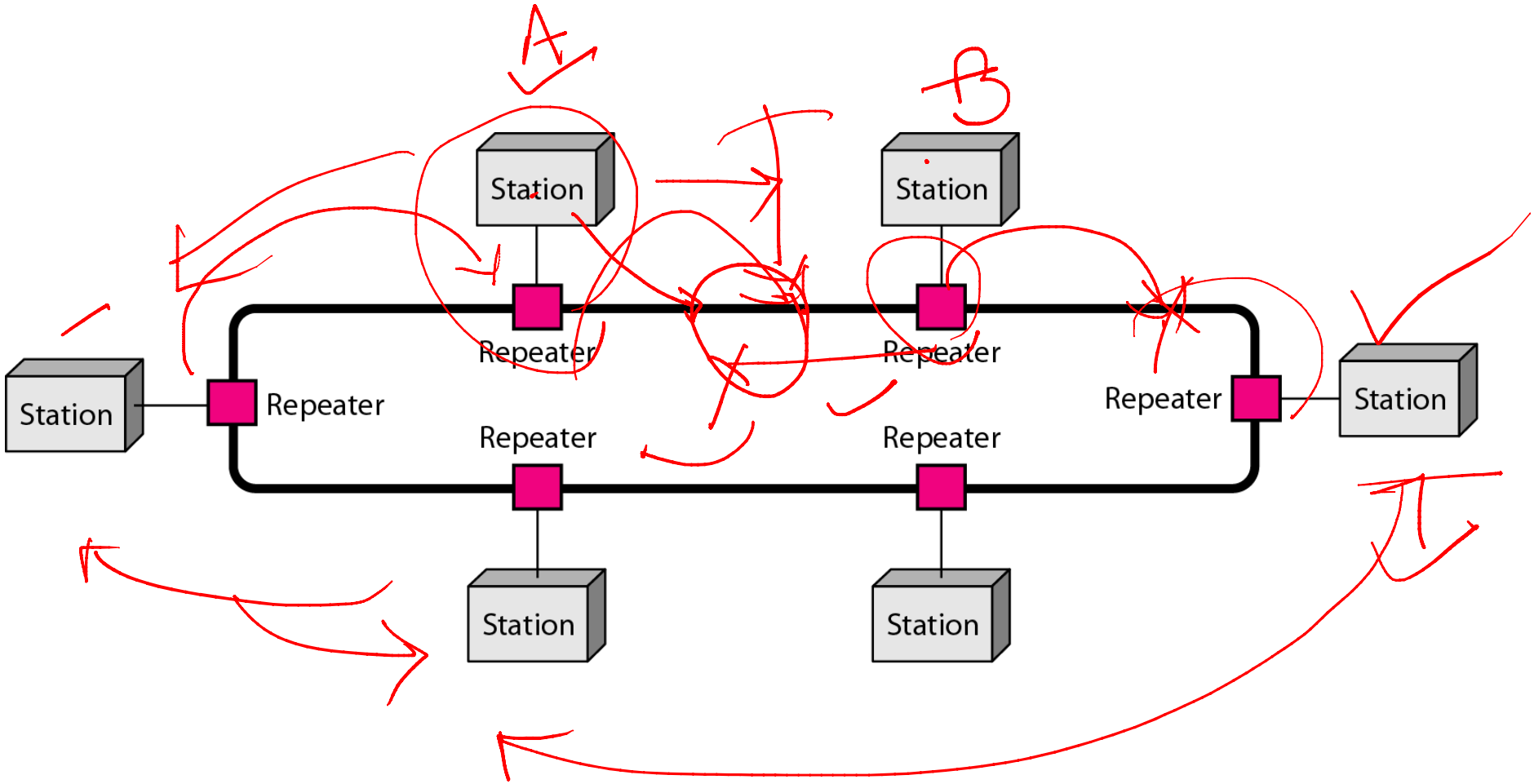
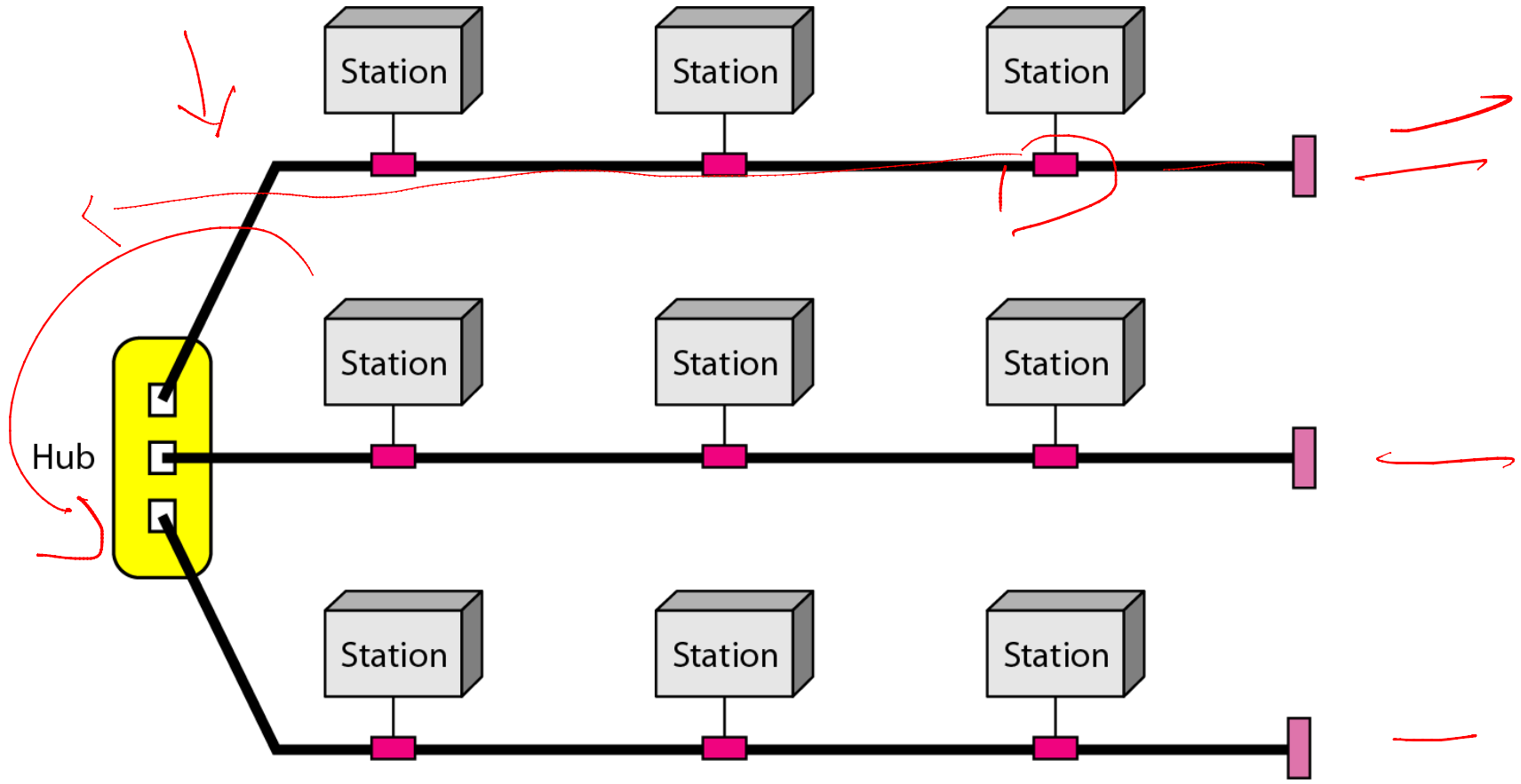


Figure 1.9 *A hybrid topology: a star backbone with three bus networks*



Categories of Networks

- **Local Area Networks (LANs)**
  - **Short distances**
  - **Designed to provide local interconnectivity**
- **Wide Area Networks (WANs)**
  - **Long distances**
  - **Provide connectivity over large areas**
- **Metropolitan Area Networks (MANs)**
  - **Provide connectivity over areas such as a city, a campus**

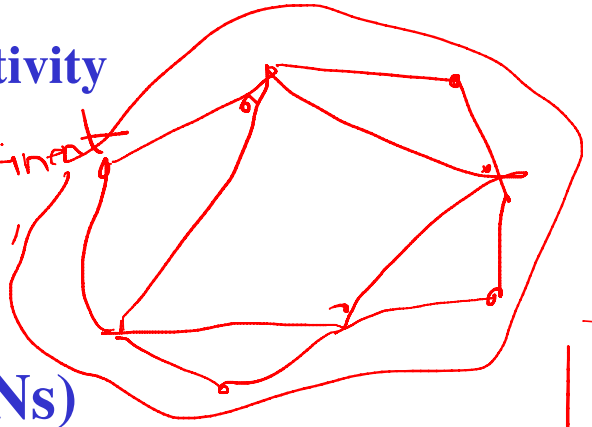


Figure 1.10 *An isolated LAN connecting 12 computers to a hub in a closet*

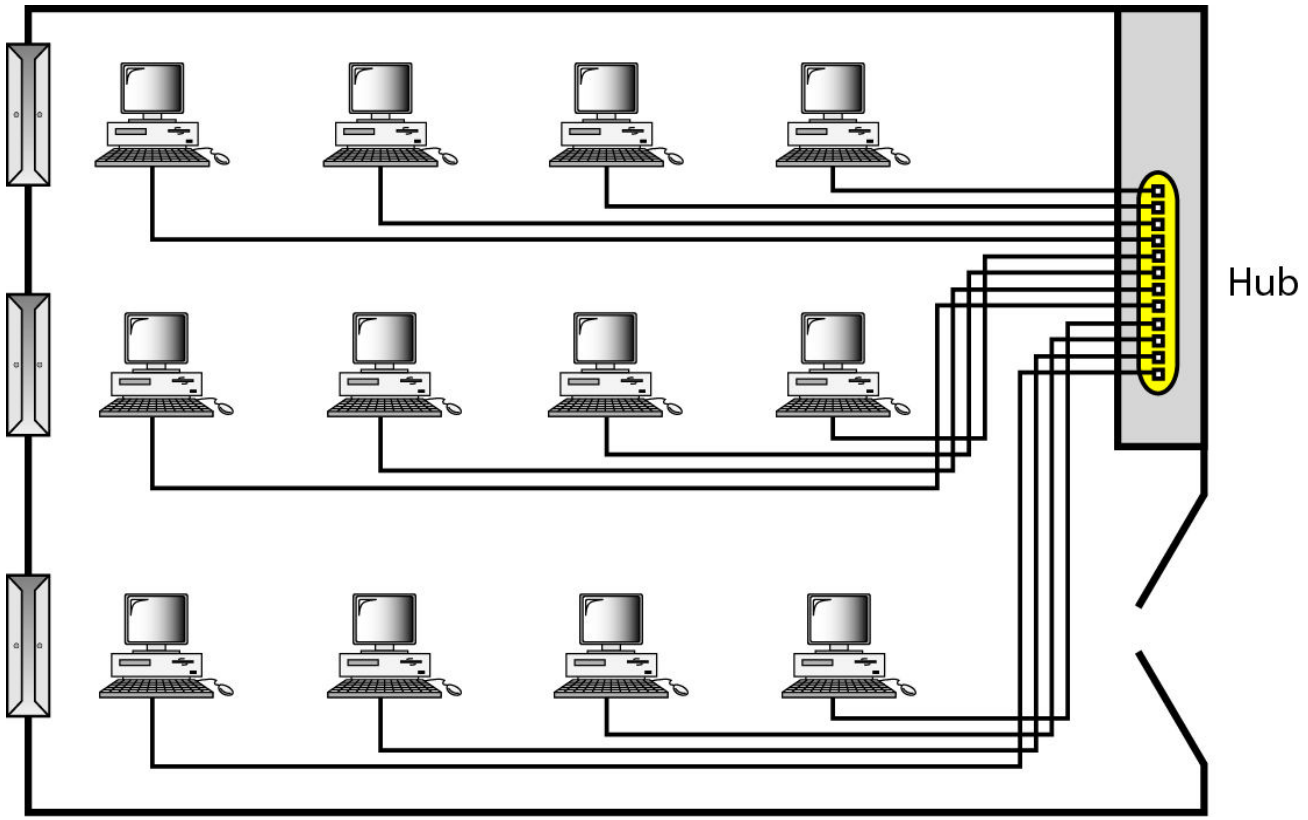


Figure 1.11 *WANs: a switched WAN and a point-to-point WAN*

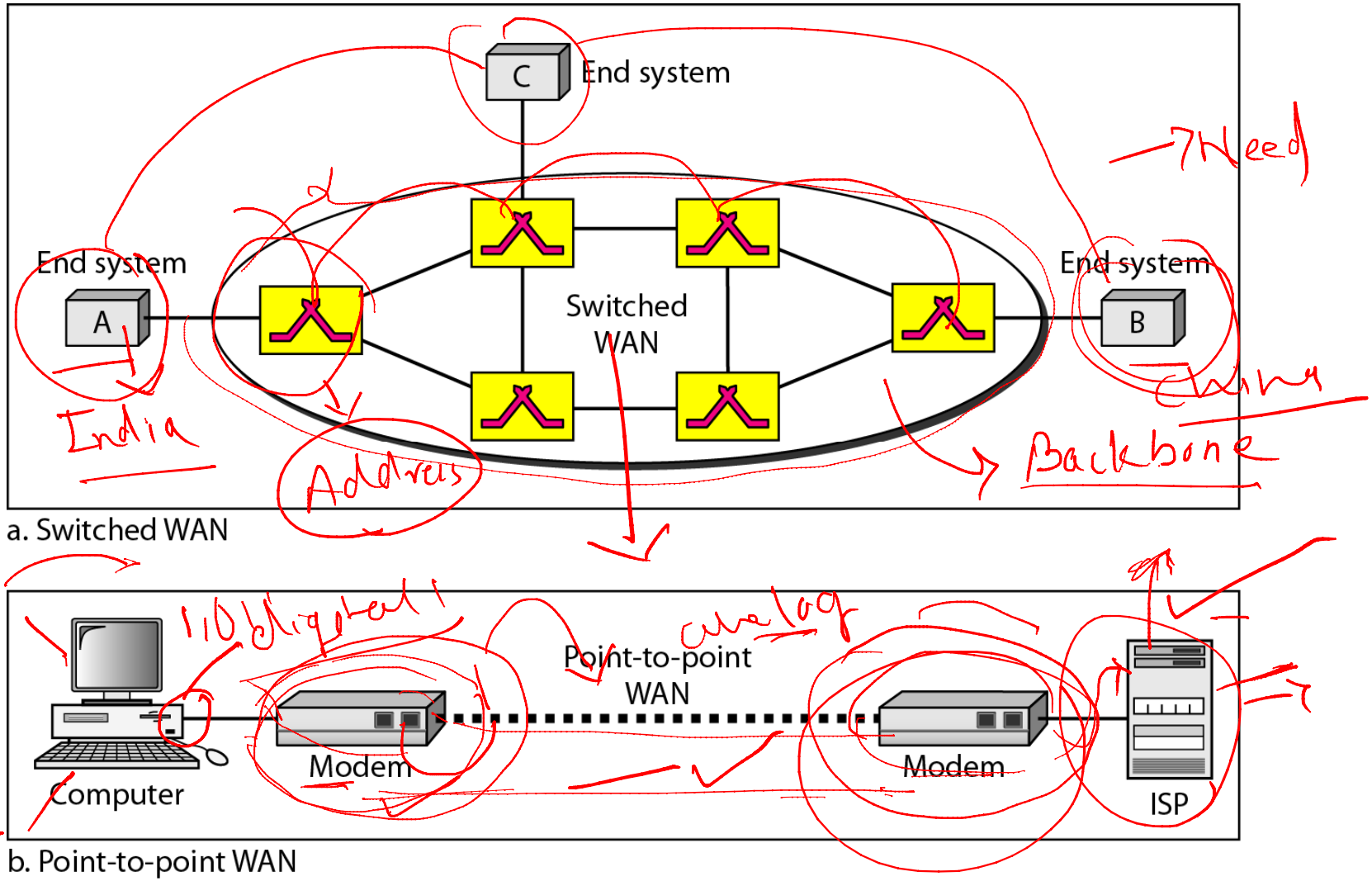
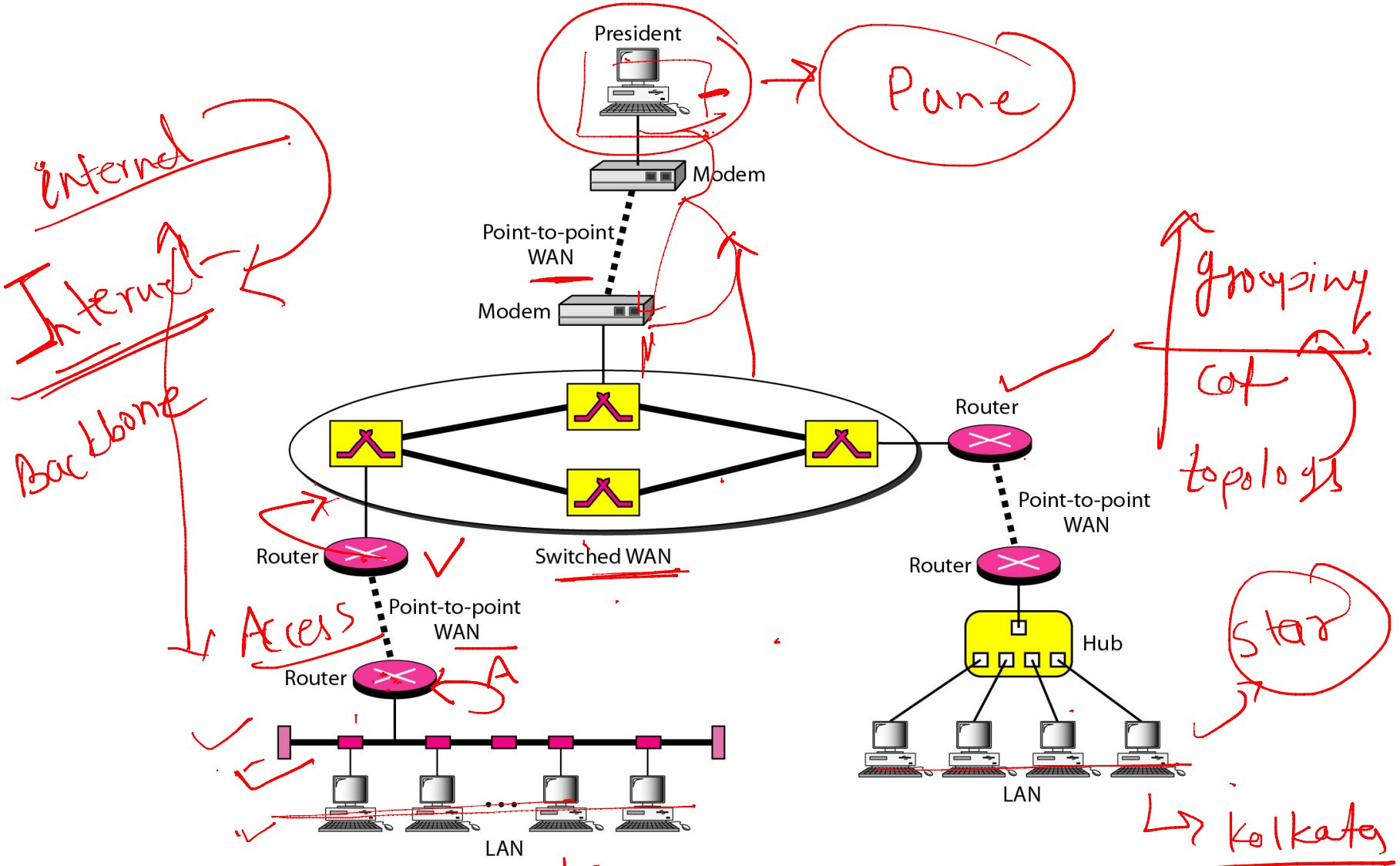
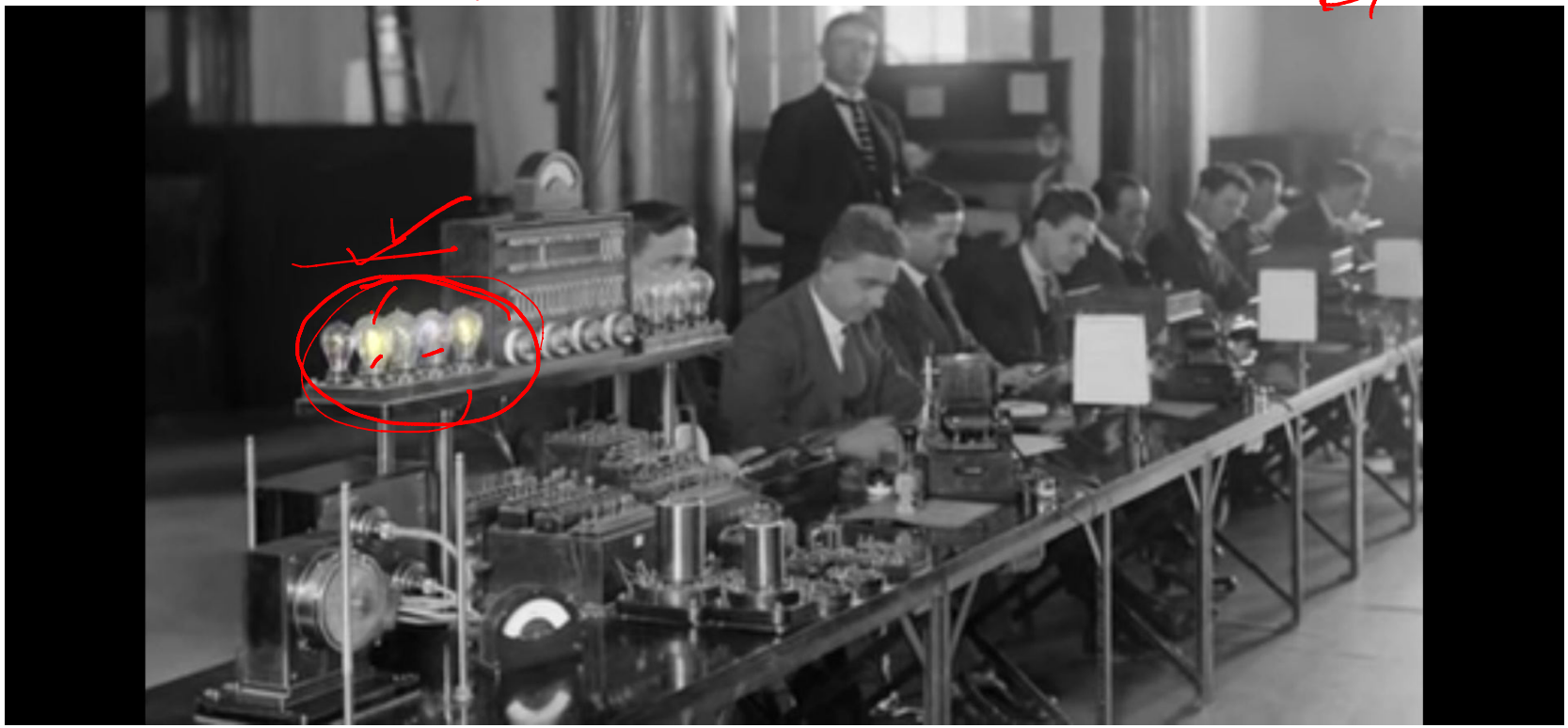


Figure 1.12 *A heterogeneous network made of four WANs and two LANs*

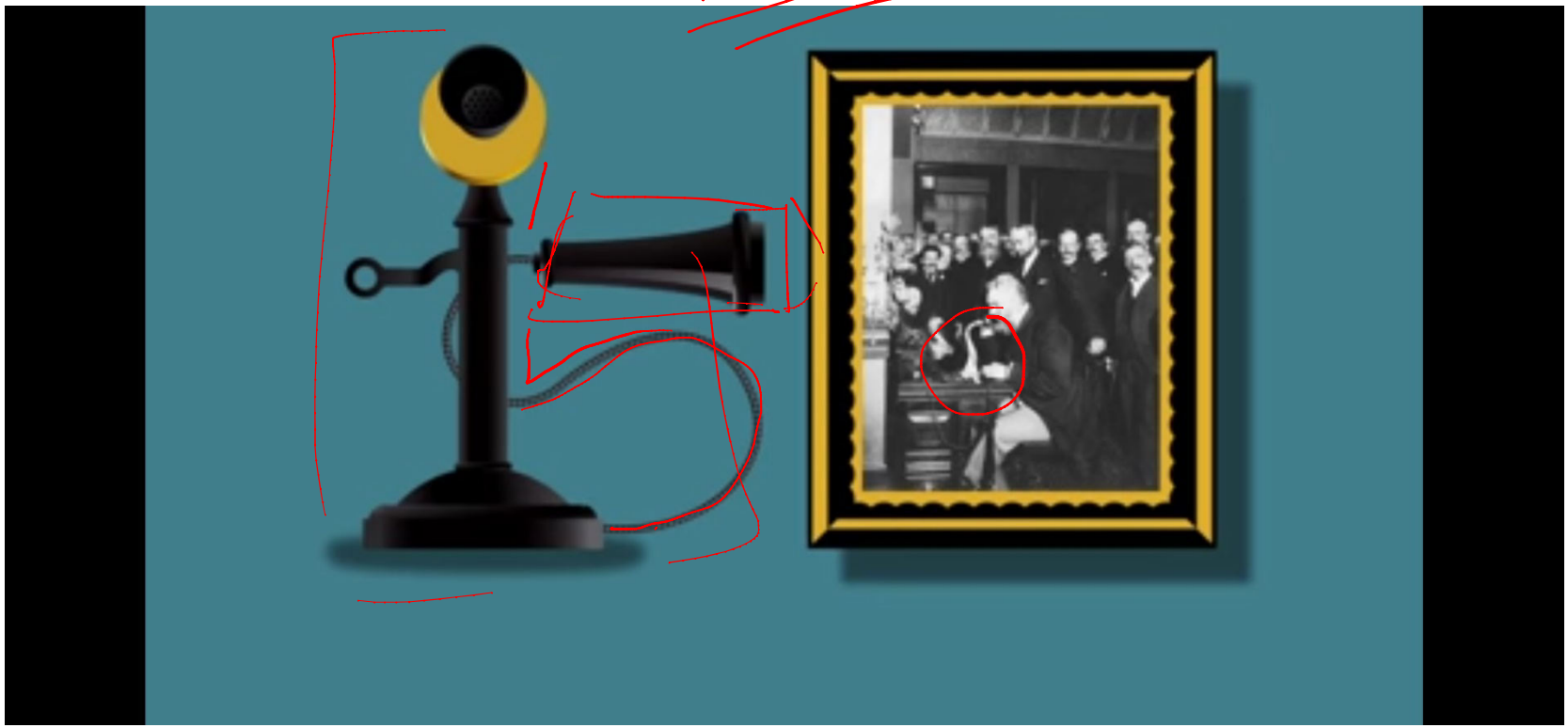


Evolution of network

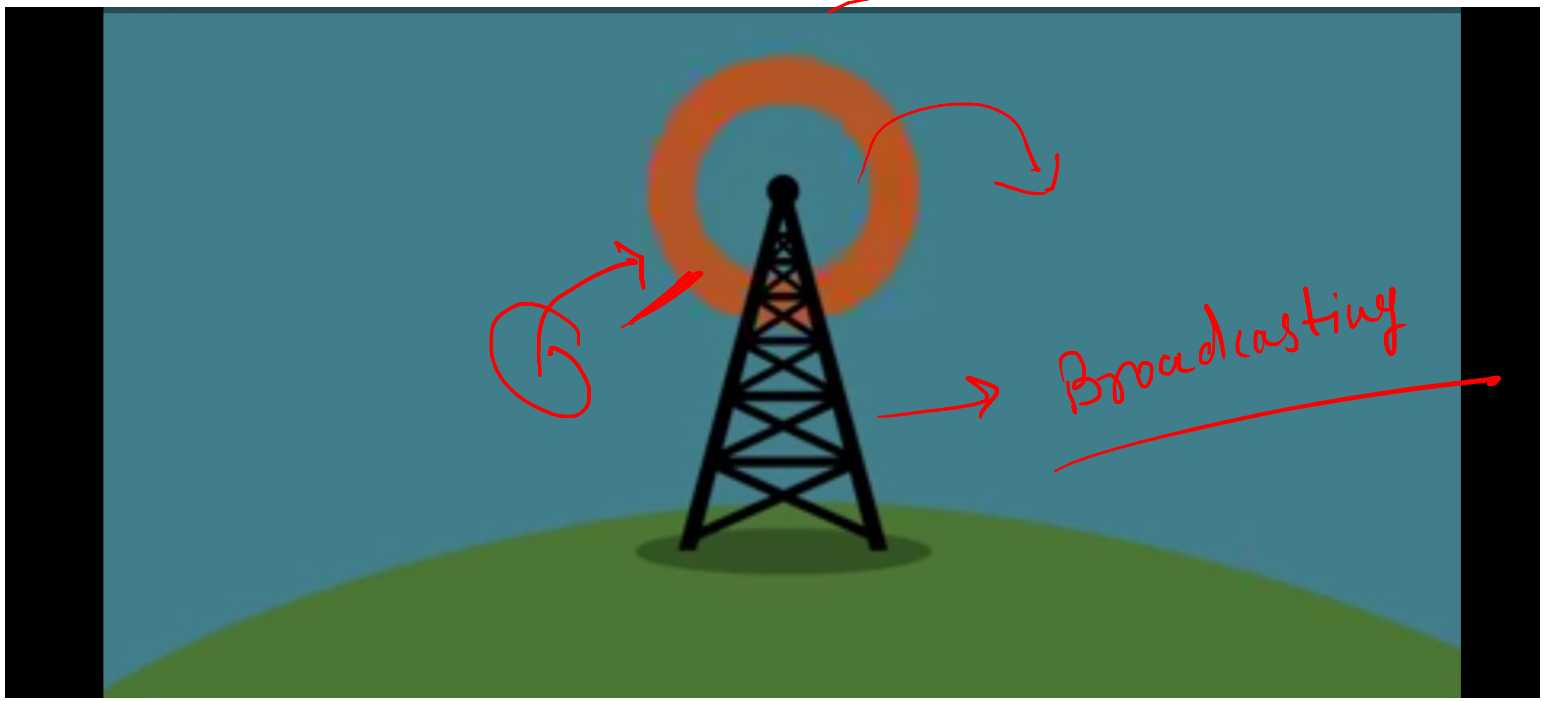


1.21

Audio



1.22


text → Audio → live
Stream
Radio
Radio
A+V
Broadcasting

1.24

IBM 650

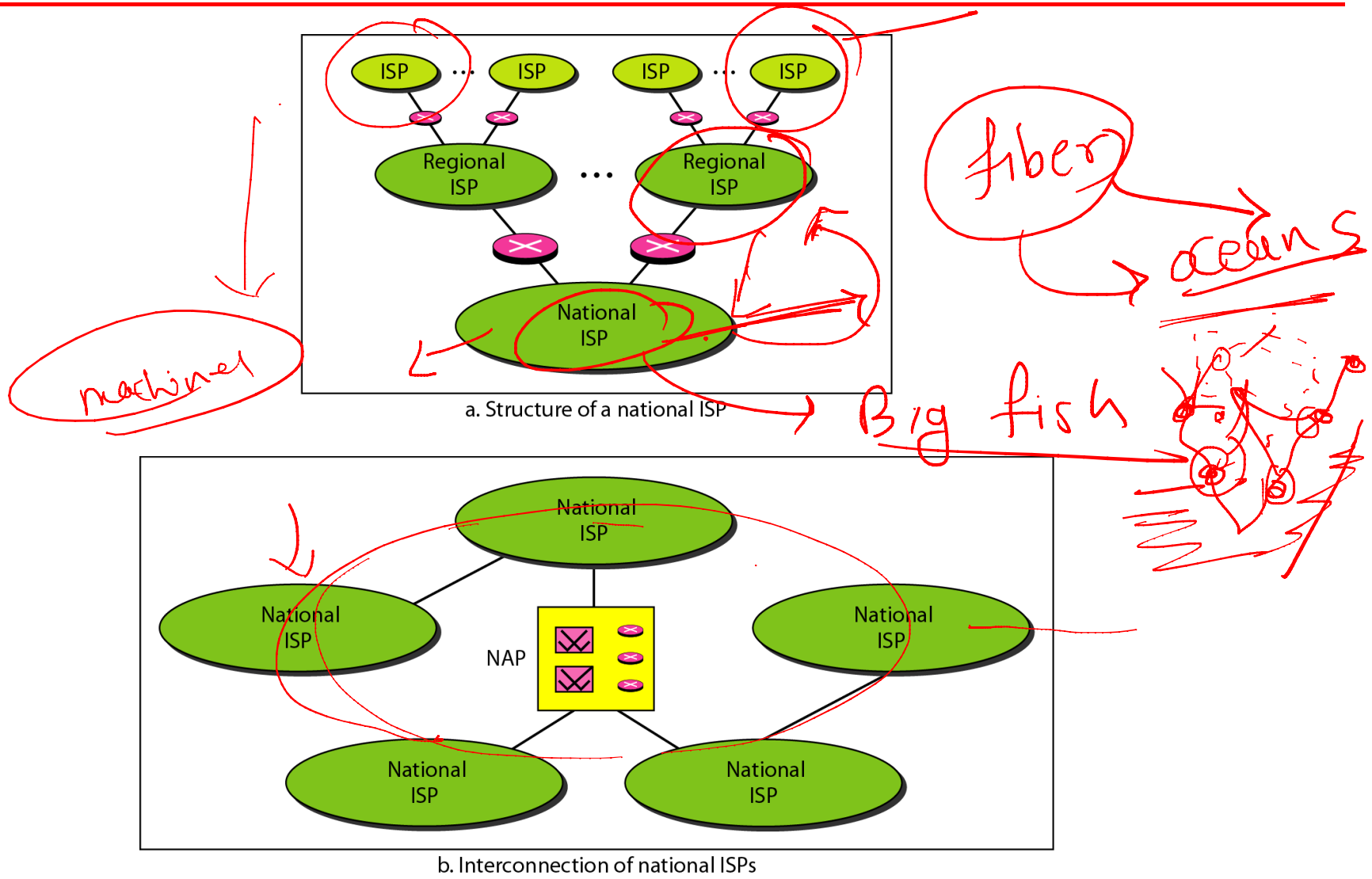
1-3 THE INTERNET

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. The Internet is a communication system that has brought a wealth of information to our fingertips and organized it for our use.

Topics discussed in this section:

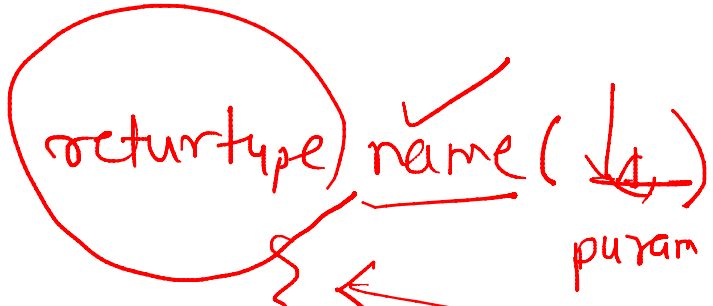
Organization of the Internet
Internet Service Providers (ISPs)

Figure 1.13 *Hierarchical organization of the Internet*



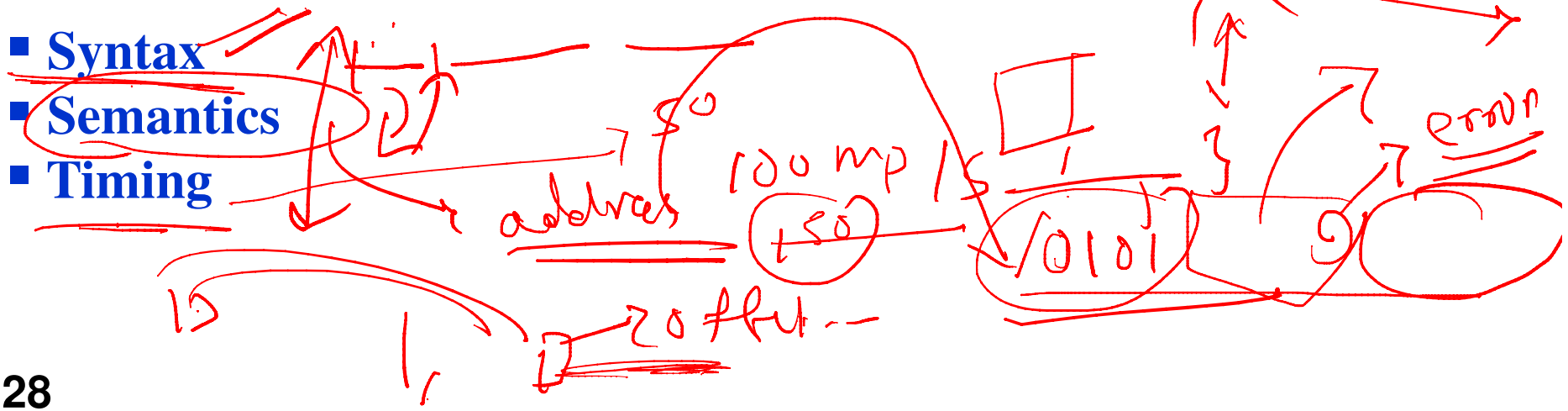
1-4 PROTOCOLS

A protocol is synonymous with rule. It consists of a set of rules that govern data communications. It determines what is communicated, how it is communicated and when it is communicated. The key elements of a protocol are syntax, semantics and timing



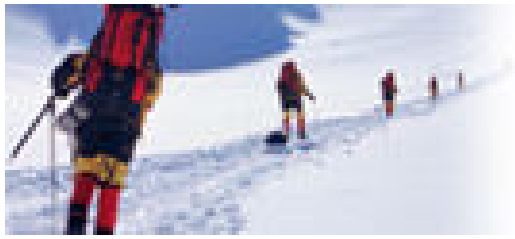
Topics discussed in this section:

- **Syntax**
- **Semantics**
- **Timing**



Elements of a Protocol

- Syntax
  - Structure or format of the data
  - Indicates how to read the bits - field delineation
- Semantics
  - Interprets the meaning of the bits
  - Knows which fields define what action
- Timing
  - When data should be sent and what
  - Speed at which data should be sent or speed at which it is being received.



Data Communications and Networking Fourth Edition

Forouzan

Chapter 23

Process-to-Process Delivery: UDP, TCP, and SCTP

23-1 PROCESS-TO-PROCESS DELIVERY

The transport layer is responsible for process-to-process delivery—the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship, as we will see later.

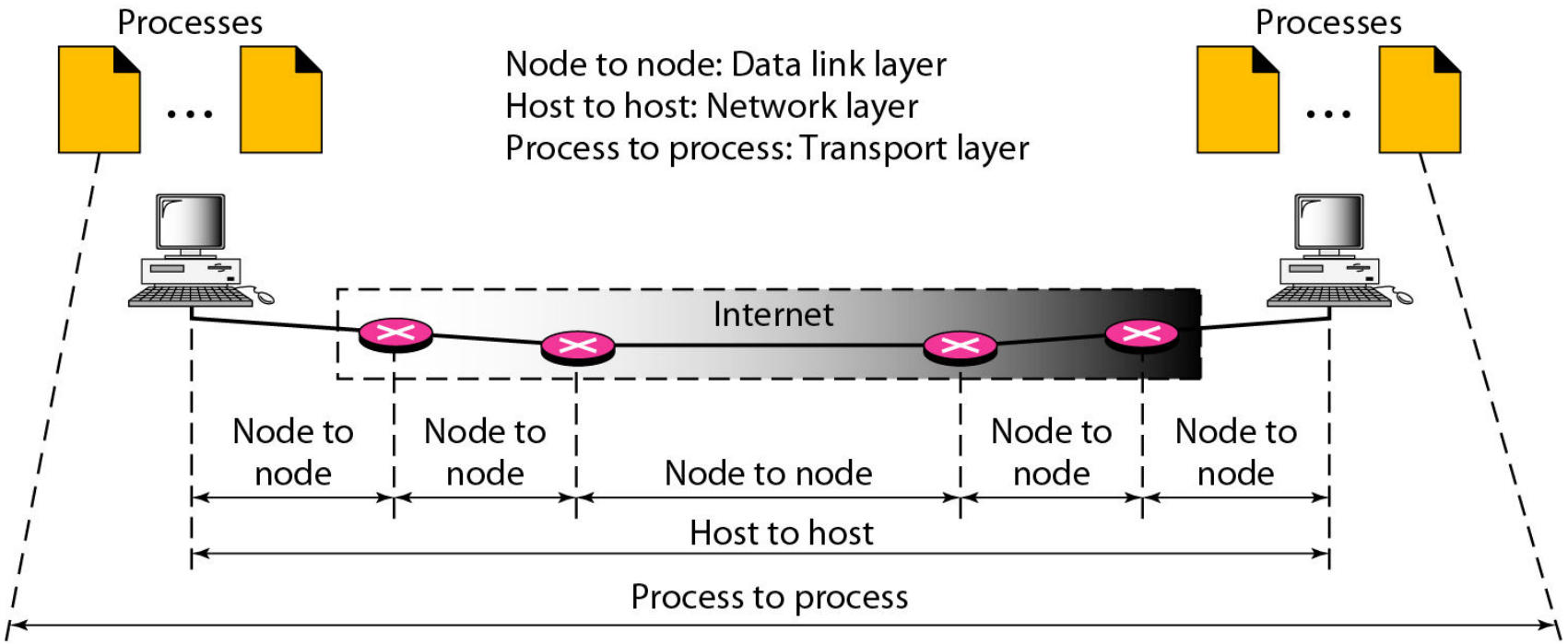
Topics discussed in this section:

Client/Server Paradigm
Multiplexing and Demultiplexing
Connectionless Versus Connection-Oriented Service
Reliable Versus Unreliable
Three Protocols

Note

The transport layer is responsible for process-to-process delivery.

Figure 23.1 *Types of data deliveries*



Client/Server Paradigm

- There are several ways to achieve process-to-process communication, Like client/server paradigm.
- A process on the local host, called a client, needs services from a process usually on the remote host, called a server.
- Both processes (client and server) have the same name

Addressing

- In the Internet model, the port numbers are 16-bit integers between 0 and 65,535.
- The client program defines itself with a port number
- How is it chosen?
  - randomly by the transport layer software running on the client host. This is the ephemeral port number.
- The server process must also define itself with a port number. This port number, **cannot** be chosen randomly

Figure 23.2 *Port numbers*

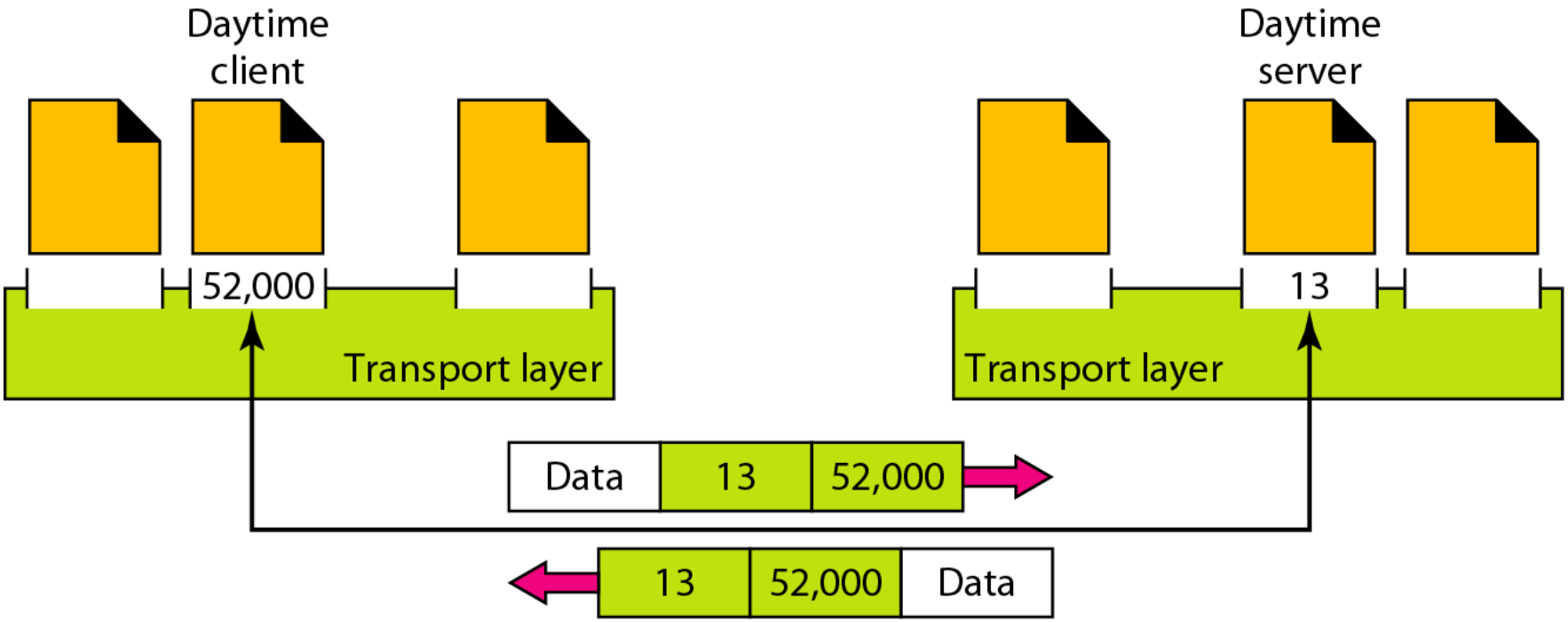
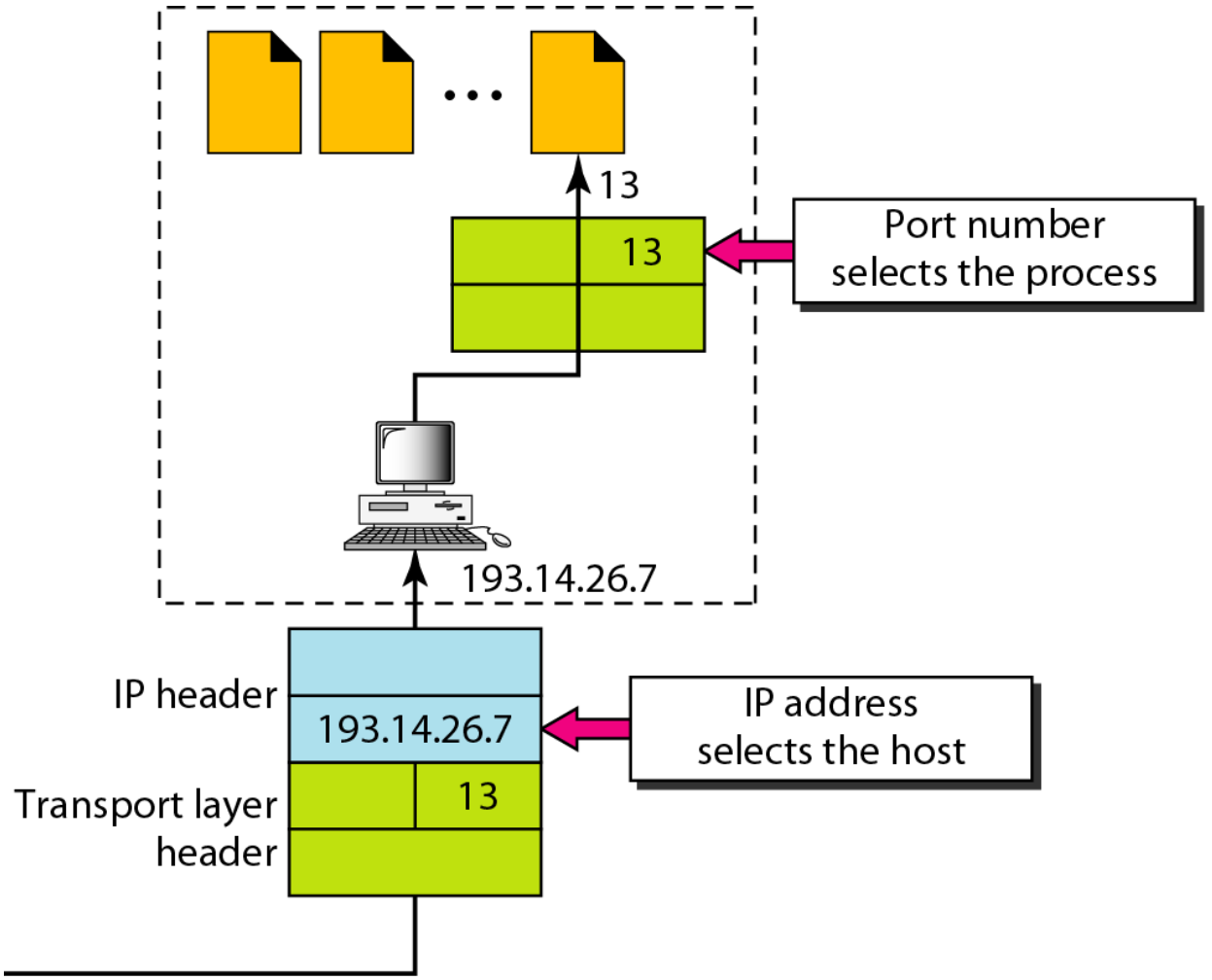


Figure 23.3 *IP addresses versus port numbers*



IANA Ranges

- Well-known ports. The ports ranging from 0 to 1023 are assigned and controlled by IANA. These are the well-known ports.
- Registered ports. The ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.
- Dynamic ports. The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process. These are the ephemeral ports.

Figure 23.4 *IANA ranges*

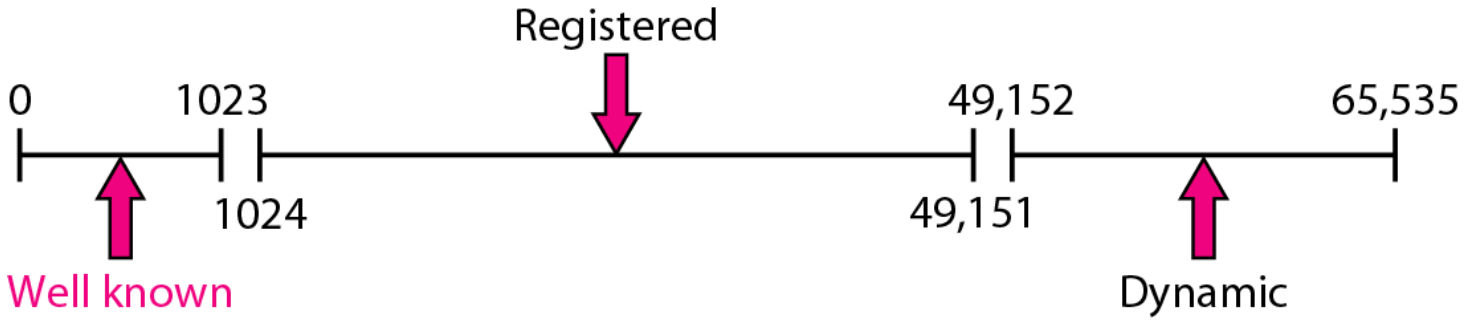


Figure 23.5 *Socket address*

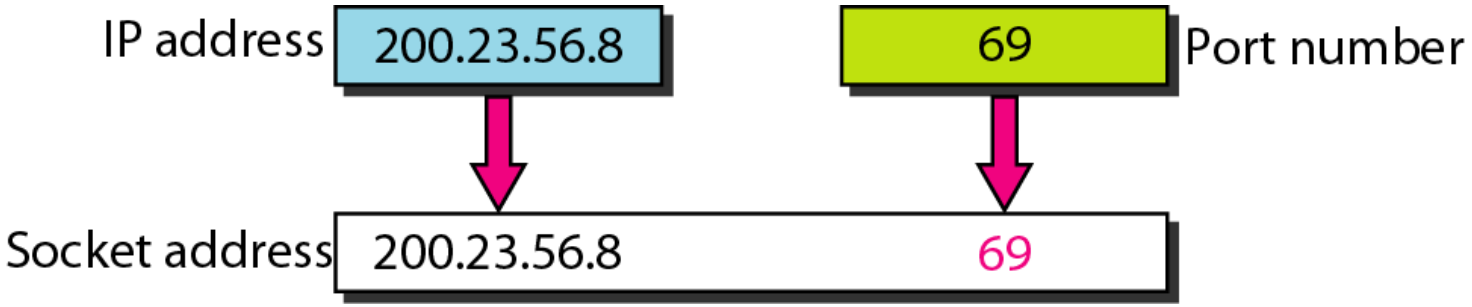
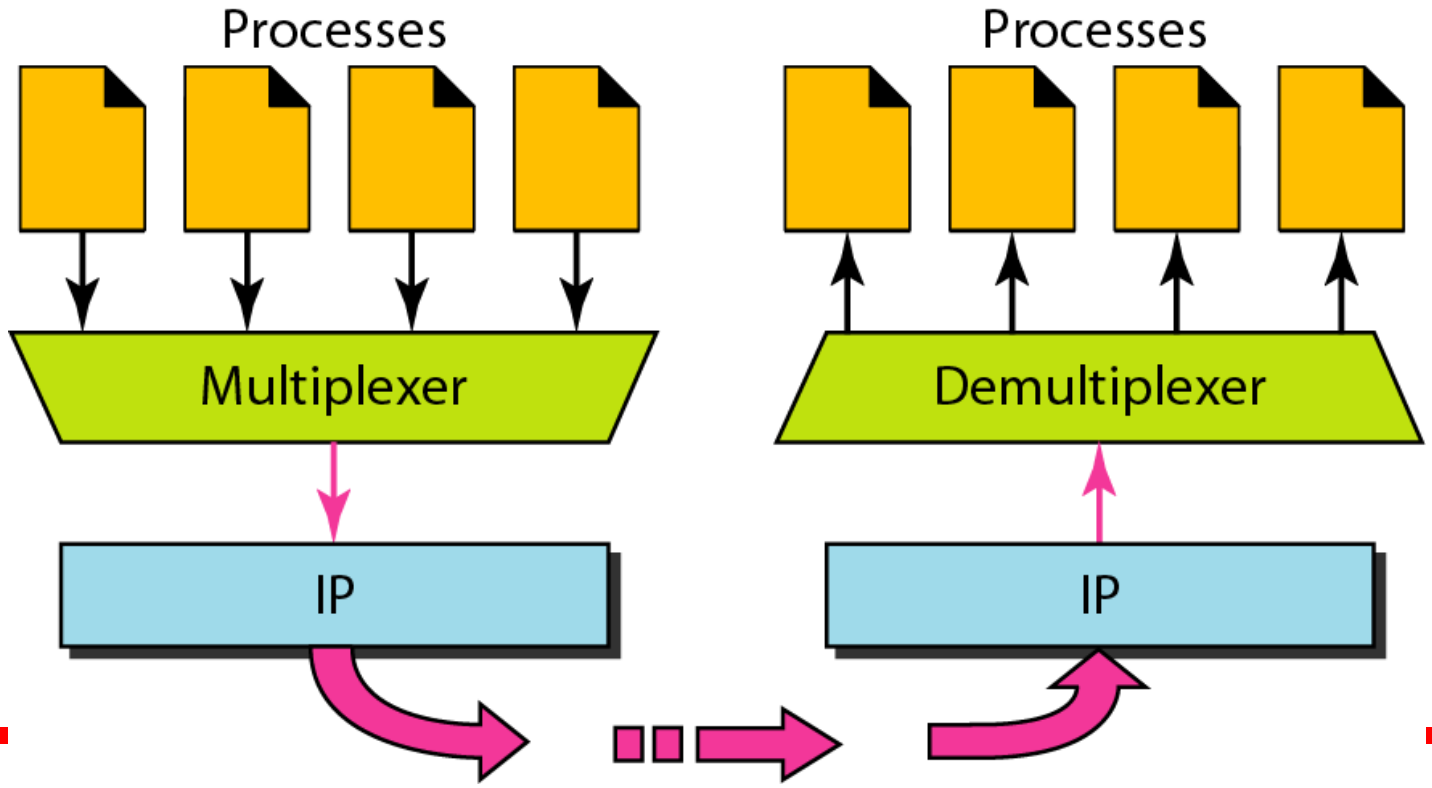


Figure 23.6 *Multiplexing and demultiplexing*

At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing.
At receiver: vice-versa



Connectionless Versus Connection-Oriented Service

- In a connectionless service:
  - no need for connection establishment or connection release.
  - The packets are not numbered
  - Delayed
  - lost
  - May arrive out of sequence.
  - no acknowledgment
  - Eg: UDP

- In a connection-oriented service:
  - a connection is first established
  - At the end the connection is released
  - TCP and SCTP

- If the data link layer is reliable and has flow and error control, why do we need this at the transport layer, too?
  - Reliability at the data link layer is between two nodes
  - we need reliability between two ends.
  - Because the network layer in the Internet is unreliable (best-effort delivery)
  - We need to implement reliability at the transport layer.

Figure 23.7 *Error control*

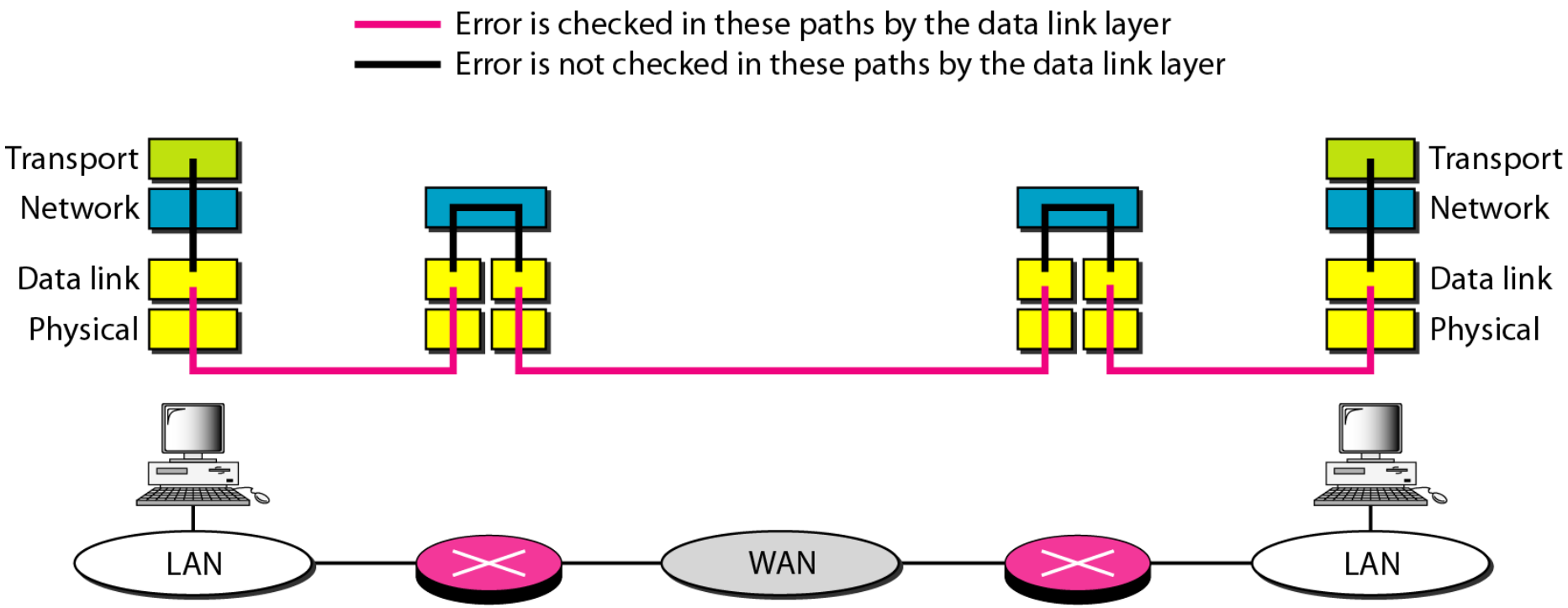
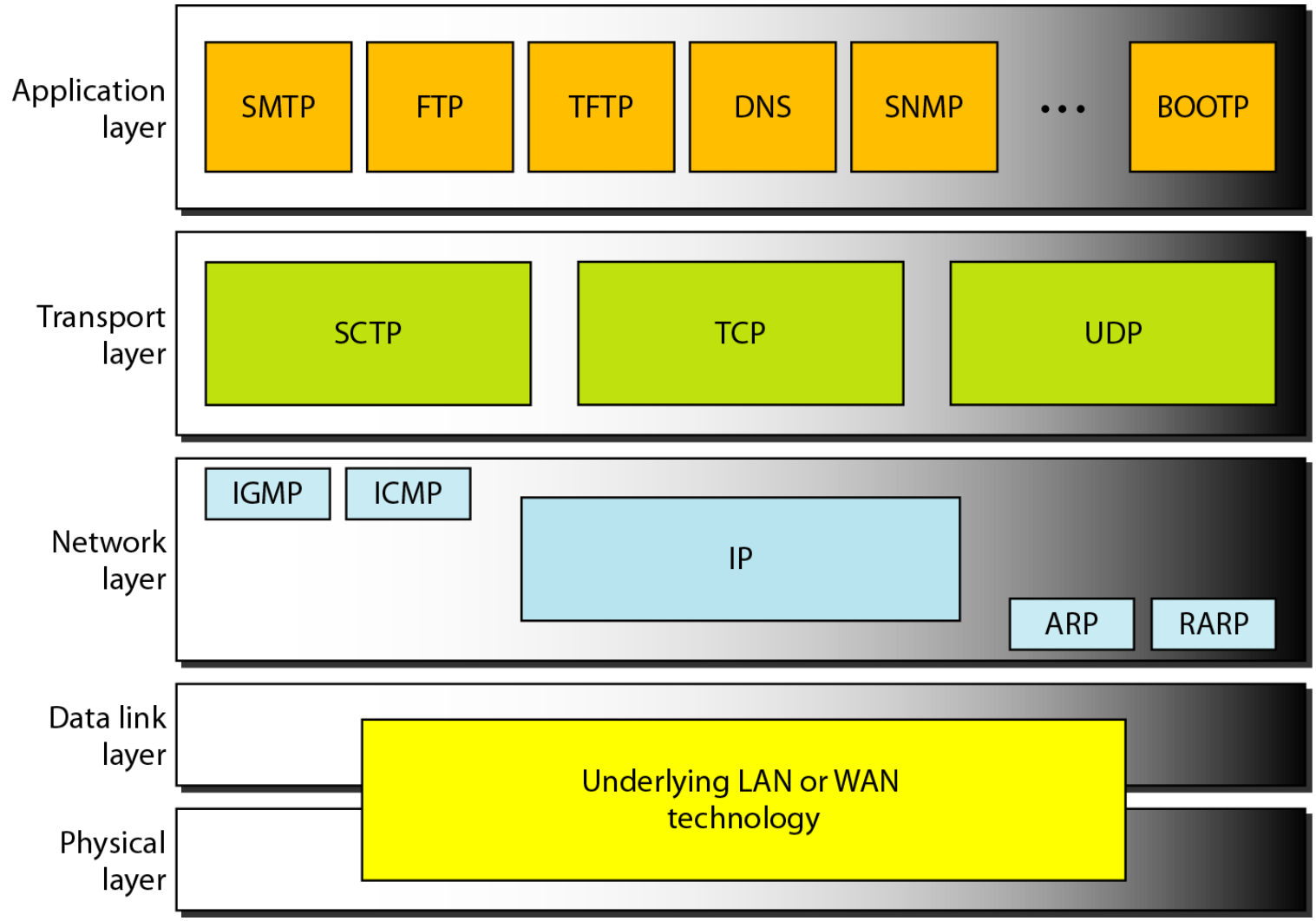


Figure 23.8 *Position of UDP, TCP, and SCTP in TCP/IP suite*



23-2 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

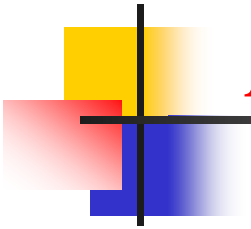
Table 23.1 *Well-known ports used with UDP*

Port	*Protocol*	*Description*
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)

Example 23.1

In UNIX, the well-known ports are stored in a file called /etc/services. Each line in this file gives the name of the server and the well-known port number. We can use the grep utility to extract the line corresponding to the desired application. The following shows the port for FTP. Note that FTP can use port 21 with either UDP or TCP.

```
$ grep    ftp    /etc/services
ftp          21/tcp
ftp          21/udp
```

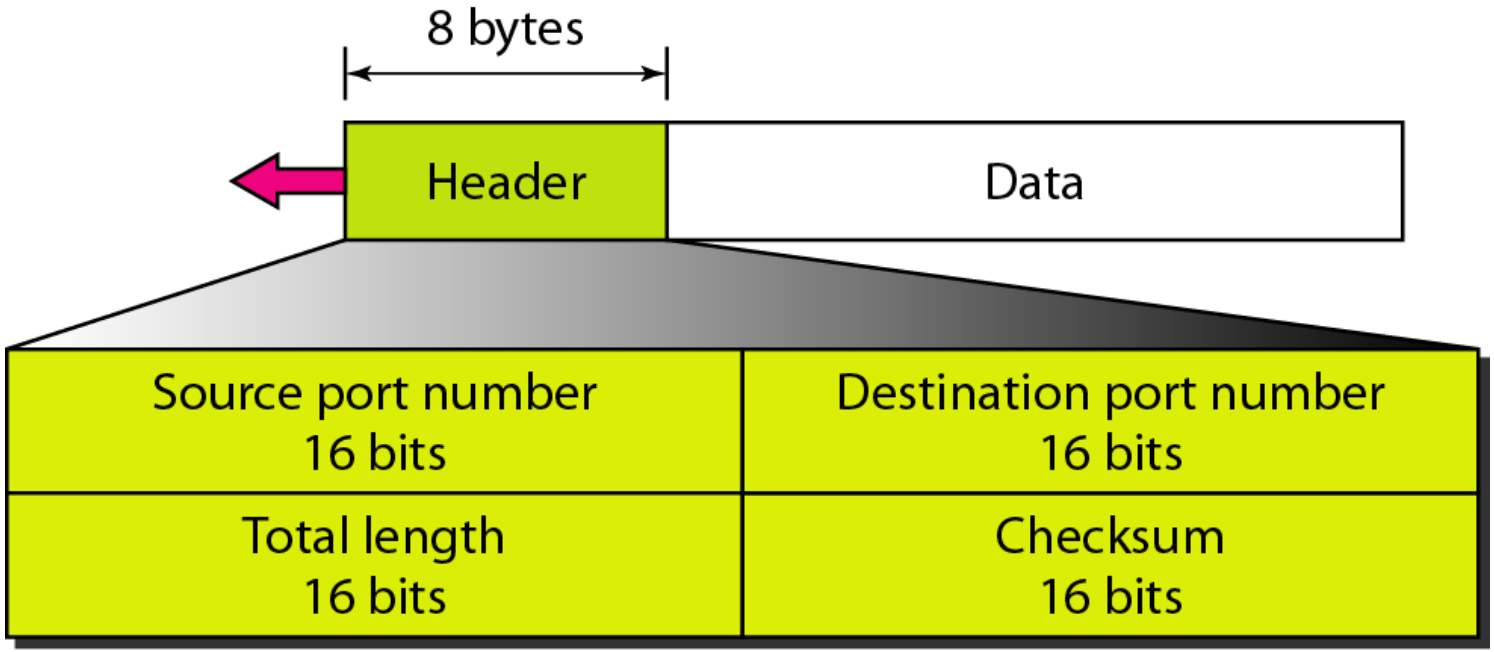


Example 23.1 (continued)

SNMP uses two port numbers (161 and 162), each for a different purpose, as we will see in Chapter 28.

```
$ grep      snmp /etc/services
snmp            161/tcp         #Simple Net  Mgmt Proto
snmp            161/udp         #Simple Net  Mgmt Proto
snmptrap        162/udp         #Traps for SNMP
```

Figure 23.9 *User datagram format*



Source port number.

- This is the port number used by the process running on the source host.
- It is 16 bits long, which means that the port number can range from 0 to 65,535.
- If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host.
- If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.

Destination port number.

- Used by the process running on the destination host.
- It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number.
- If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number.
- In this case, the server copies the ephemeral port number it has received in the request packet.

Length

- This is a 16-bit field that defines the total length of the user datagram :header plus data.
- The 16 bits can define a total length of 0 to 65,535 bytes
- But the total length needs to be much less , why?
  - because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes
- Is the length field really necessary?
- Why to add then?

Note

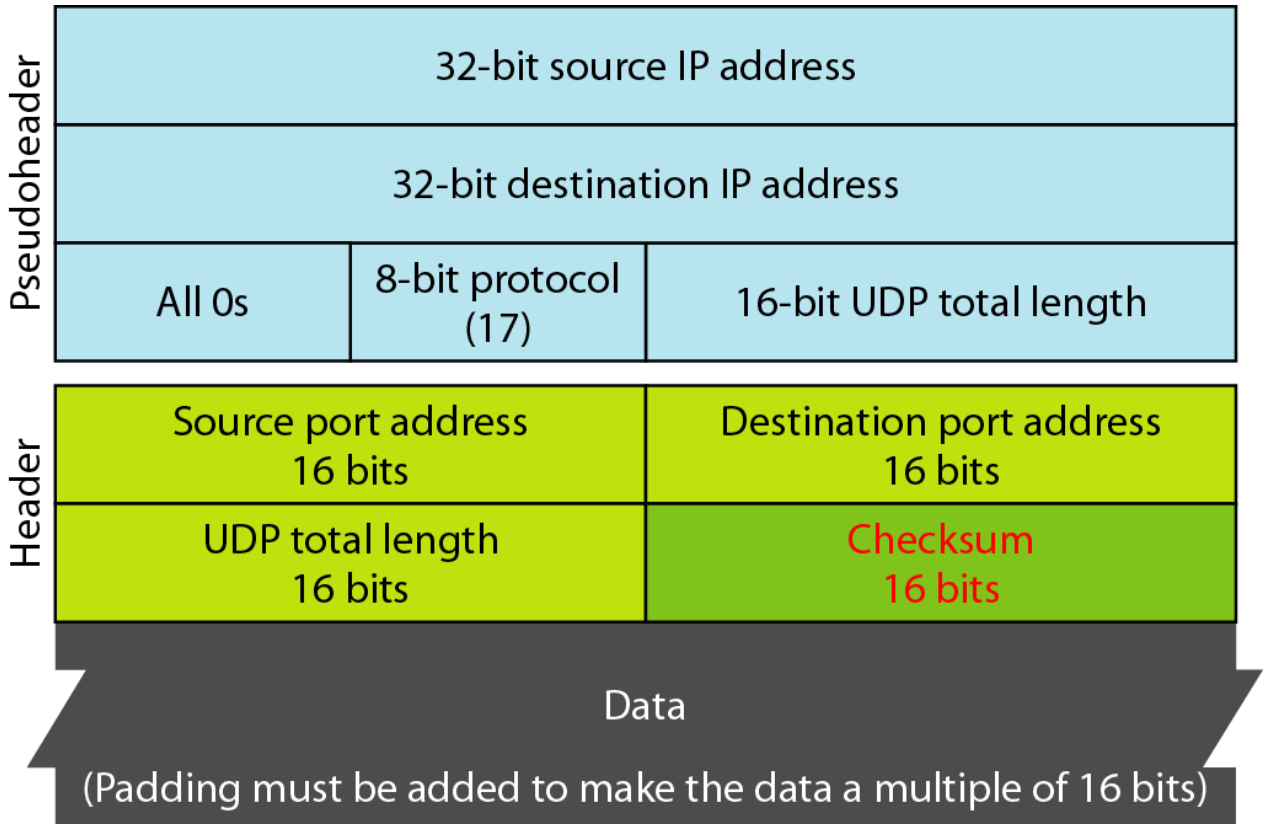
UDP length
= IP length – IP header's length

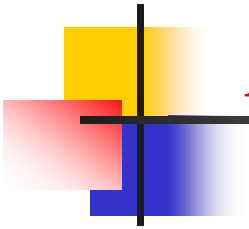
Checksum

- This field is used to detect errors over the entire user datagram (header plus data)
- Checksum includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer
- The pseudoheader is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s
- If the checksum does not include the pseudoheader, 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.
  - If a process can use either UDP or TCP, the destination port number can be the same.
  - The value of the protocol field for UDP is 17.
  - If this value is changed during transmission, the checksum calculation at the receiver will detect it and UDP drops the packet.
  - It is not delivered to the wrong protocol.
- The calculation of the checksum and its inclusion in a user datagram are optional. If the checksum is not calculated, the field is filled with 1s.

Figure 23.10 *Pseudoheader for checksum calculation*

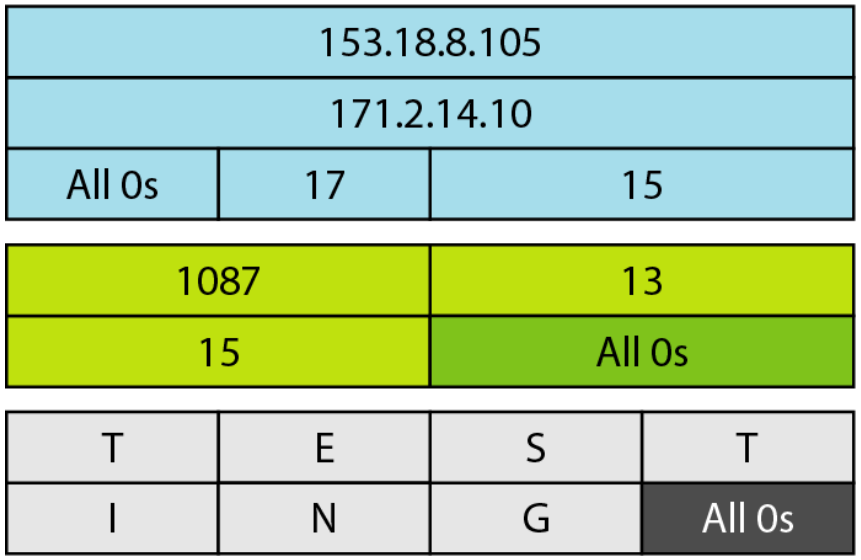




Example 23.2

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.

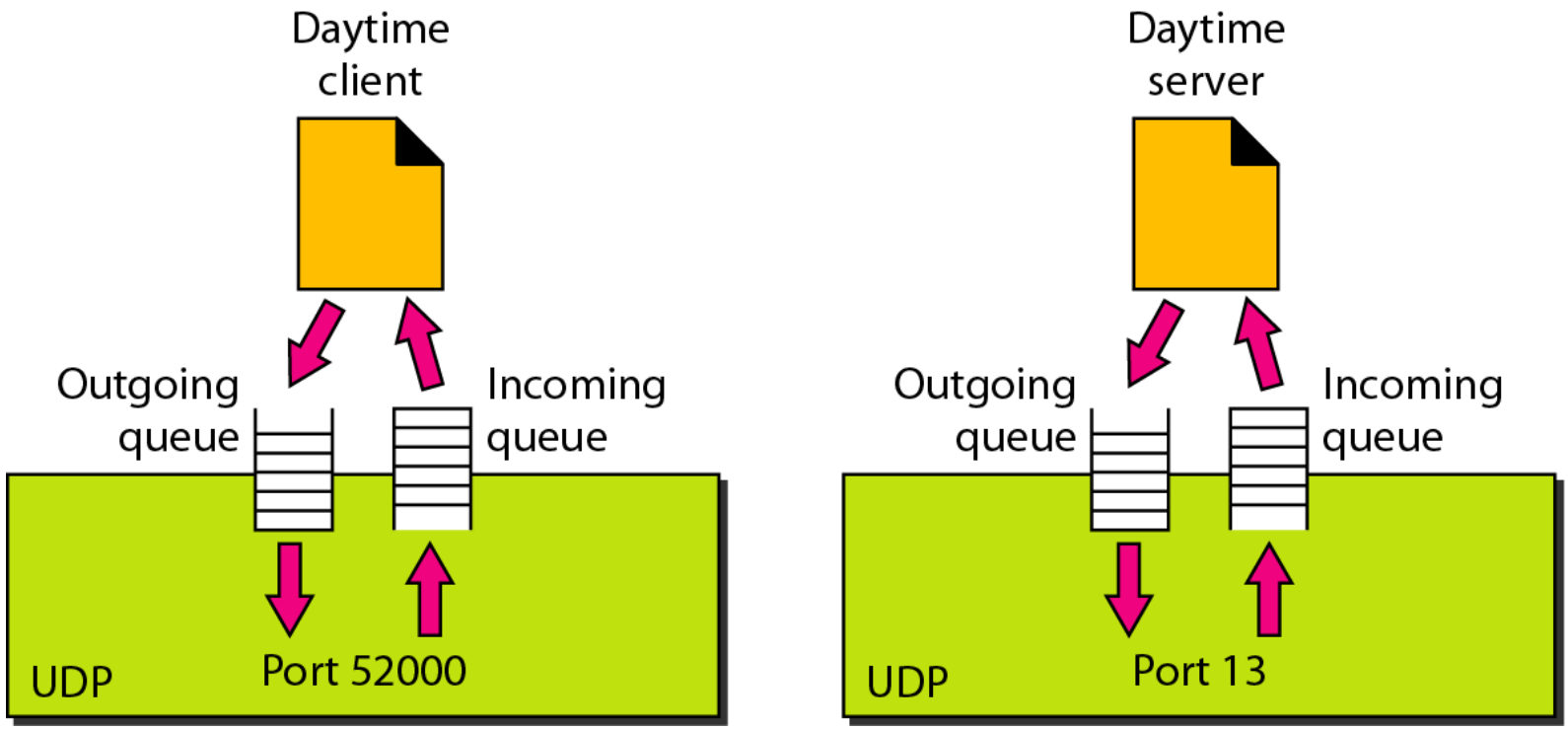
Figure 23.11 *Checksum calculation of a simple UDP user datagram*



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)
10010110 11101011	Sum
01101001 00010100	Checksum

The calculation of the checksum and its inclusion in a user datagram are optional. If the checksum is not calculated, the field is filled with 1s.

Figure 23.12 *Queues in UDP*



Client side

- Even if a process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue.
- The queues opened by the client are, identified by ephemeral port numbers.
- The queues function as long as the process is running. When the process terminates, the queues are destroyed.

- The client process can send messages to the outgoing queue by using the source port number specified in the request.
- UDP removes the messages one by one and, adds the UDP header, delivers them to IP.
- An outgoing queue can overflow.
- If this happens, the operating system can ask the client process to wait before sending any more messages.

Message arrives for a client

- UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.
- If there is such a queue, UDP sends the received user datagram to the end of the queue.
- If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a *port unreachable* message to the server.
- All the incoming messages for one particular client program, whether coming from the same or a different server, are sent to the same queue.
- An incoming queue can overflow. If this happens, UDP drops the user datagram and asks for a port unreachable message to be sent to the server.

Server side

- Server asks for incoming and outgoing queues, using its well-known port when it starts running.
- The queues remain open as long as the server is running.
- When a message arrives for a server, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.
- If there is such a queue, UDP sends the received user datagram to the end of the queue.
- If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the client

- When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port number specified in the request.
- UDP removes the messages one by one and, after adding the UDP header, delivers them to IP.
- Both queue can overflow

23-3 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.

Topics discussed in this section:

TCP Services
TCP Features
Segment
A TCP Connection
Flow Control
Error Control

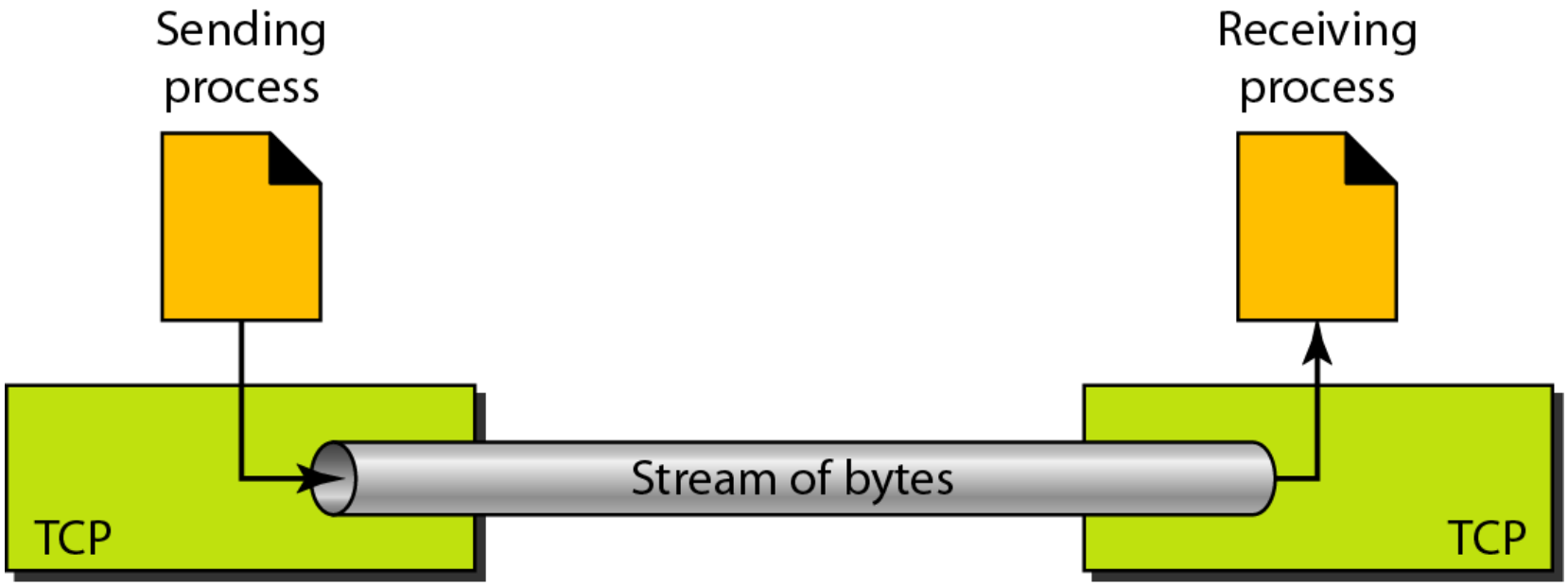
TCP Services

- *Process-to-Process Communication:*
  - *using port number*
- *Stream Delivery Service:*
  - stream-oriented protocol
  - Creates an environment in which the two processes seem to be connected by an imaginary "tube“ that carries their data across the Internet

Table 23.2 *Well-known ports used by TCP*

Port	*Protocol*	*Description*
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

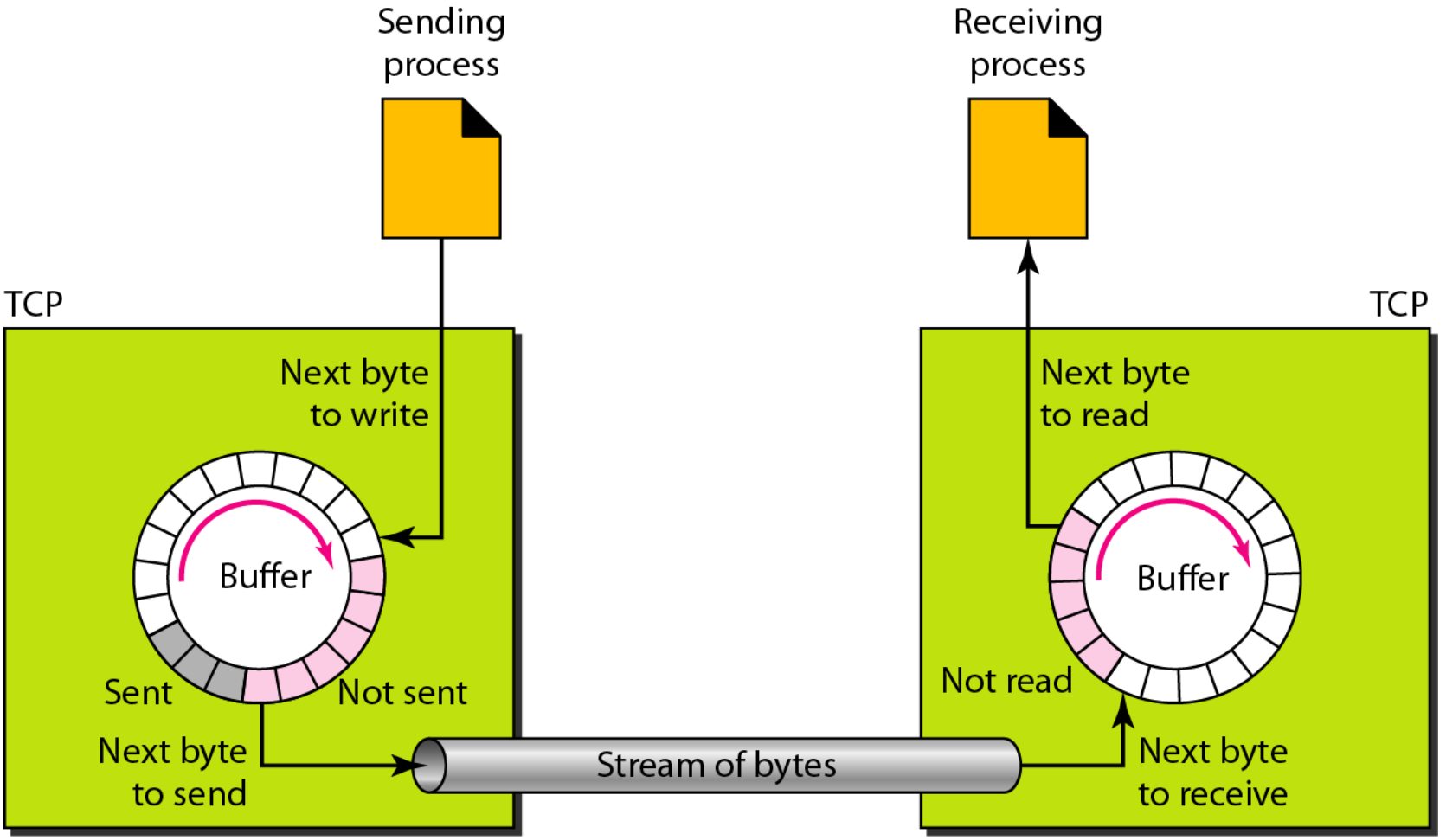
Figure 23.13 *Stream delivery*



Sending and Receiving Buffers

- 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: sending buffer and the receiving buffer

Figure 23.14 *Sending and receiving buffers*

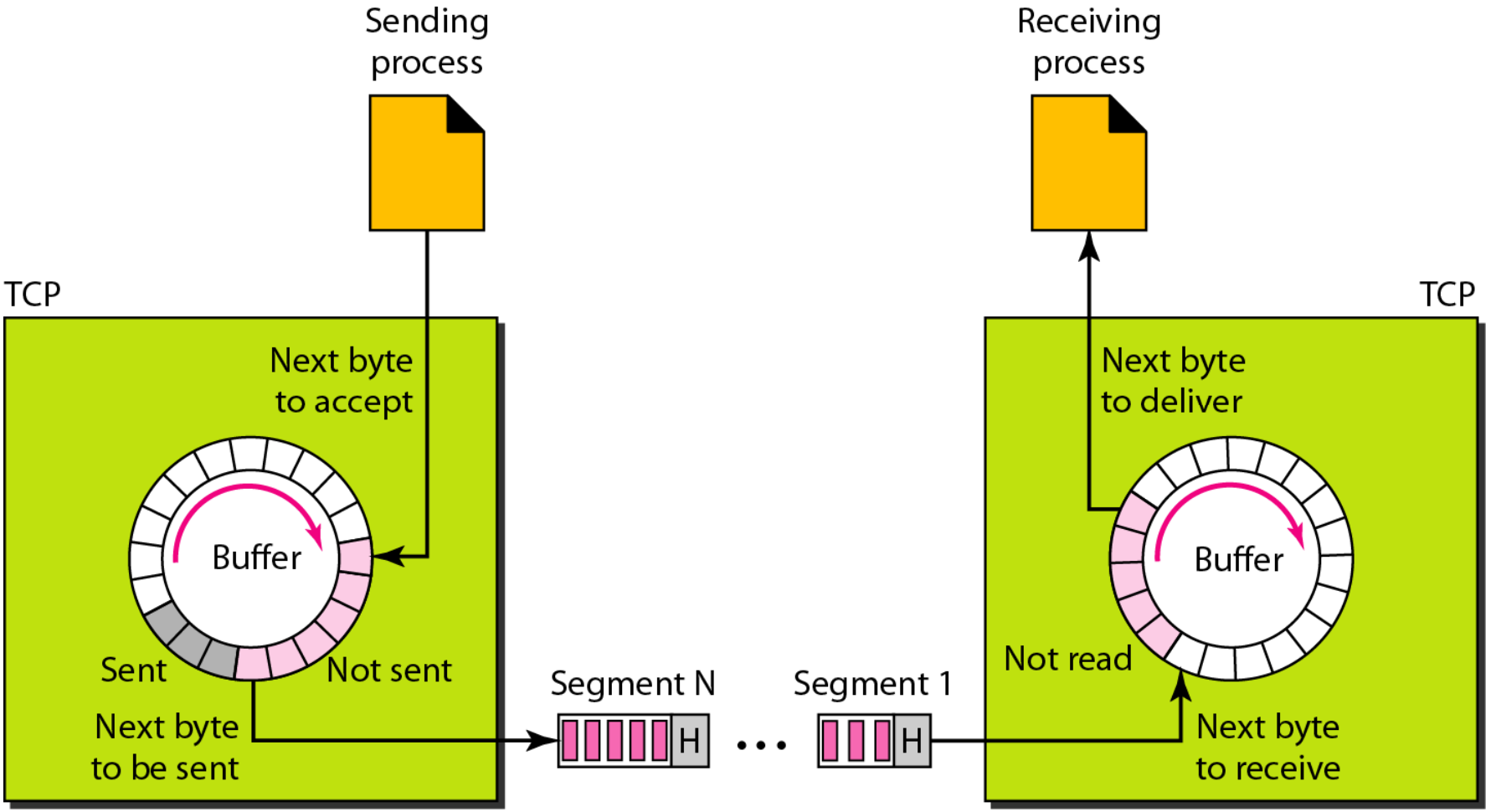


- Buffering handles the disparity between the speed of the producing and consuming processes
- So can we send the data which was given by buffer now?
  - The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes

Segments

- TCP groups a number of bytes together into a packet called a segment.
- TCP adds a header to each segment and delivers the segment to the IP layer for transmission.
- The segments are encapsulated in IP datagrams and transmitted.
- Segments are not necessarily the same size

Figure 23.15 *TCP segments*



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.

Connection-Oriented Service

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

Reliable Service

- TCP is a reliable transport protocol.
- It uses an acknowledgment mechanism to check the safe and sound arrival of data

Numbering System

Note

The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated 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

Example 23.3

Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 10,00l. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

The following shows the sequence number for each segment:

Segment 1	→	Sequence Number: 10,001 (range: 10,001 to 11,000)
Segment 2	→	Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3	→	Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4	→	Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5	→	Sequence Number: 14,001 (range: 14,001 to 15,000)

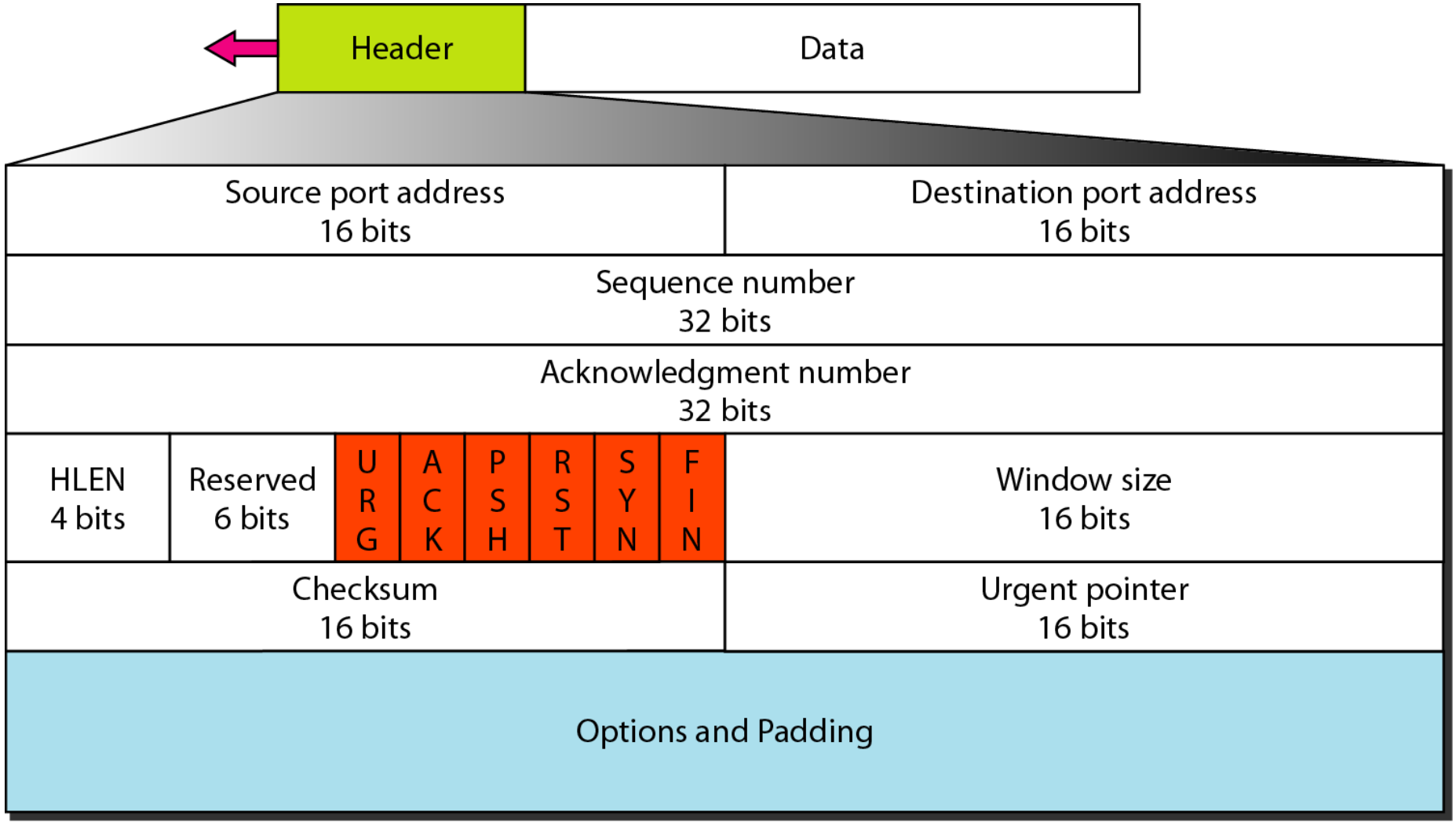
Note

The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.

Note

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.

Figure 23.16 *TCP segment format*



- 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.
- To ensure connectivity, each byte to be transmitted is numbered.
- The sequence number =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

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.
- Value of this field can be between 4 and 15

Figure 23.17 *Control field*

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

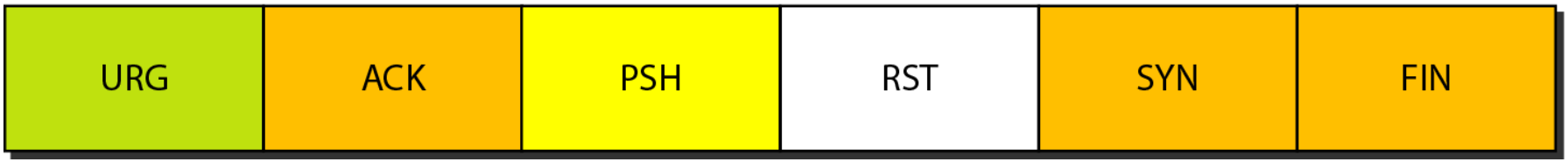


Table 23.3 *Description of flags in the control field*

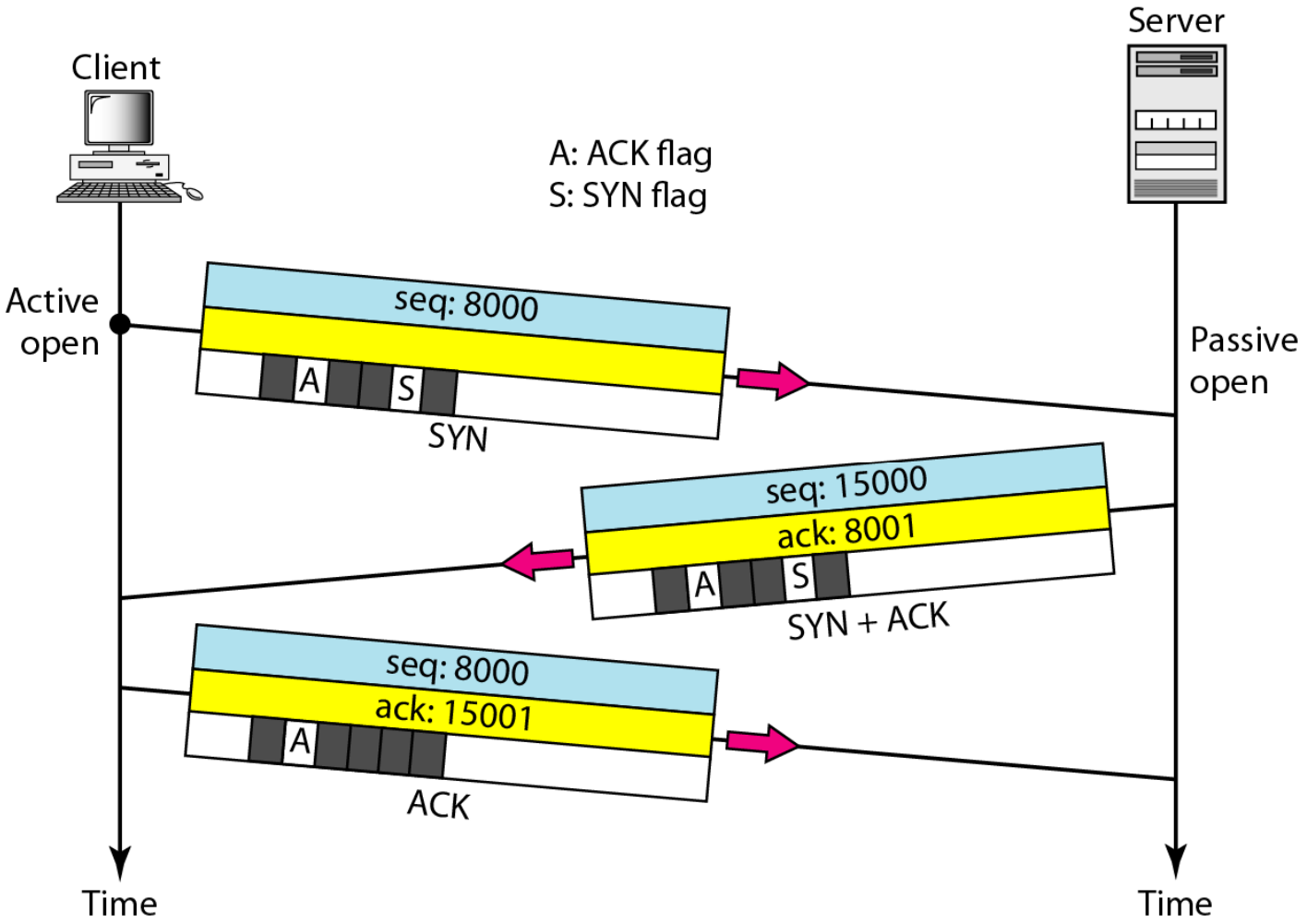
Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

A TCP Connection

- How TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented?
  - TCP connection is virtual, not physical.
  - TCP operates at a higher level.
  - TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself.
  - If a segment is lost or corrupted, it is retransmitted. If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering.

- *Passive open :*The process starts with the server. The server program tells its TCP that it is readyto accept a connection.
- Although the server TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself.
- *Active open:* A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server.

Figure 23.18 *Connection establishment using three-way handshaking*



Note

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

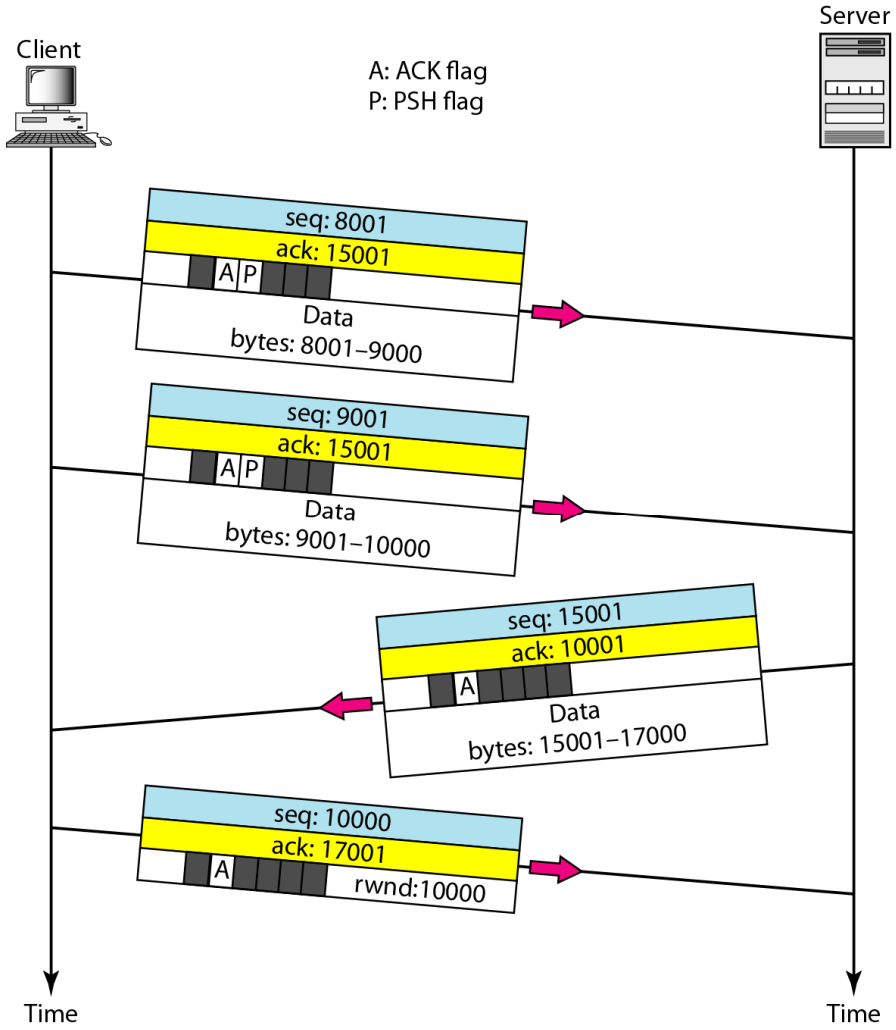
Note

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

Note

An ACK segment, if carrying no data, consumes no sequence number.

Figure 23.19 *Data transfer*



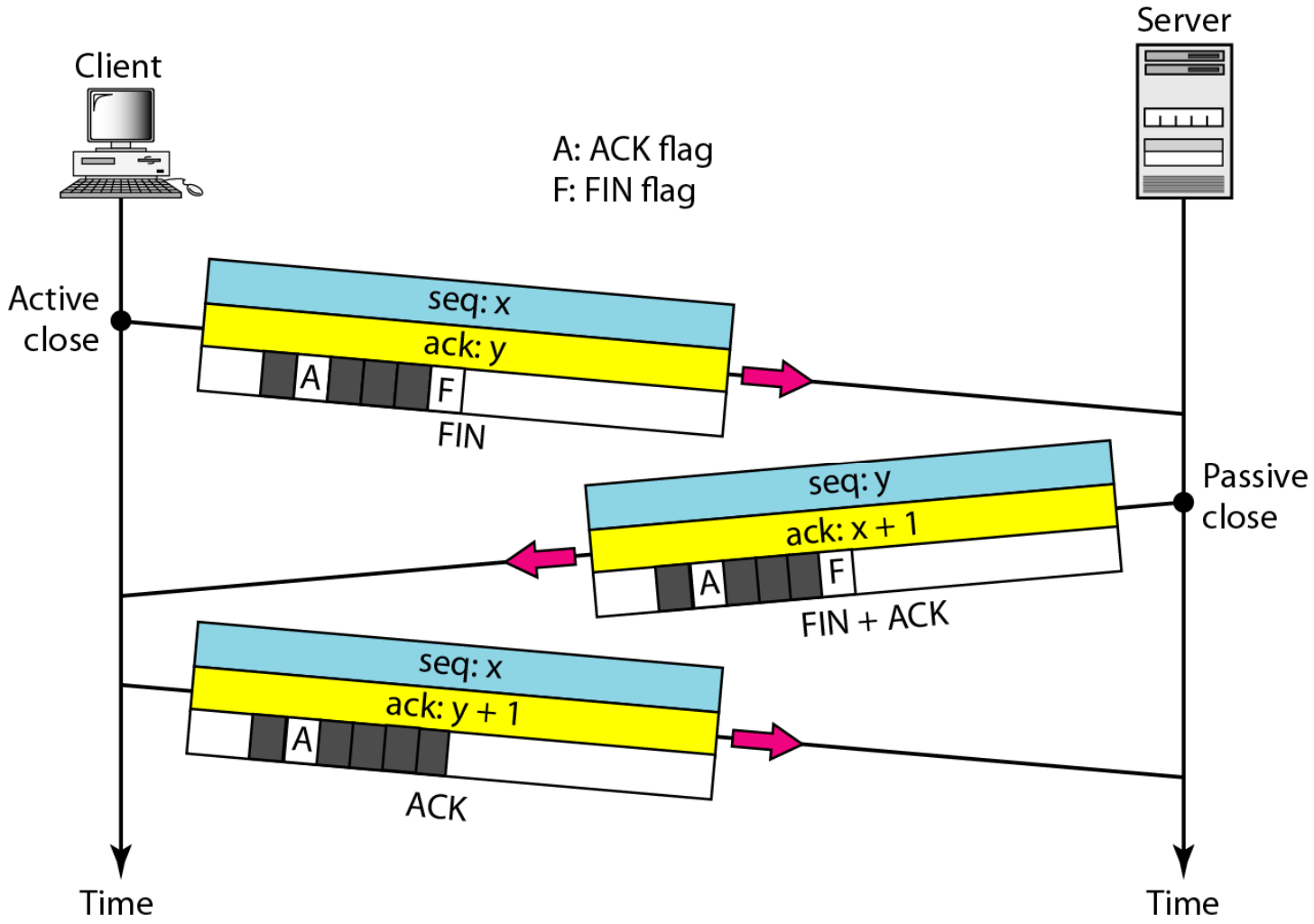
Push DATA

- Interactive communication
- Application requiring immediate response

Urgent Data

- Process: always inorder
- What if application data which have been sent is wrong and needs to be terminated immediately?
  - Urgent bit
  - The sending application program tells the sending TCP that the piece of data is urgent.
  - The sending TCP creates a segment and inserts the urgent data at the beginning of the segment.
  - The rest of the segment can contain normal data from the buffer
  - The urgent pointer field in the header defines the end of the urgent data and the start of normal data.
  - When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment, using the value of the urgent pointer, and delivers them, out of order, to the receiving application program.

Figure 23.20 *Connection termination using three-way handshaking*



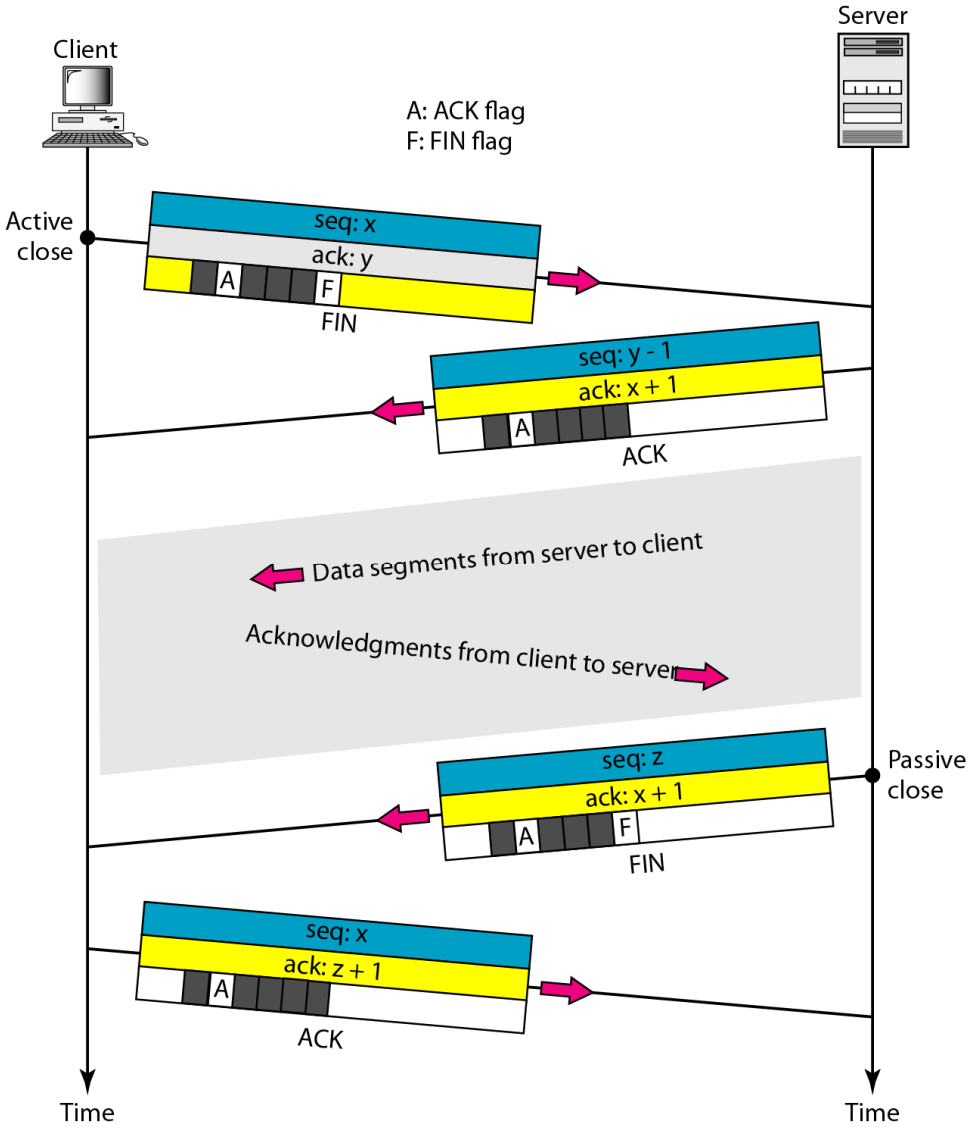
Note

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

Note

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

Figure 23.21 *Half-close*



- After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server.
- The client cannot send any more data to the server.
  - Note the sequence numbers: The second segment (ACK) consumes no sequence number.
  - Although the client has received sequence number y - 1 and is expecting y, the server sequence number is still y - 1.
  - When the connection finally closes, the sequence number of the last ACK segment is still x, because no sequence numbers are consumed during data transfer in that direction.

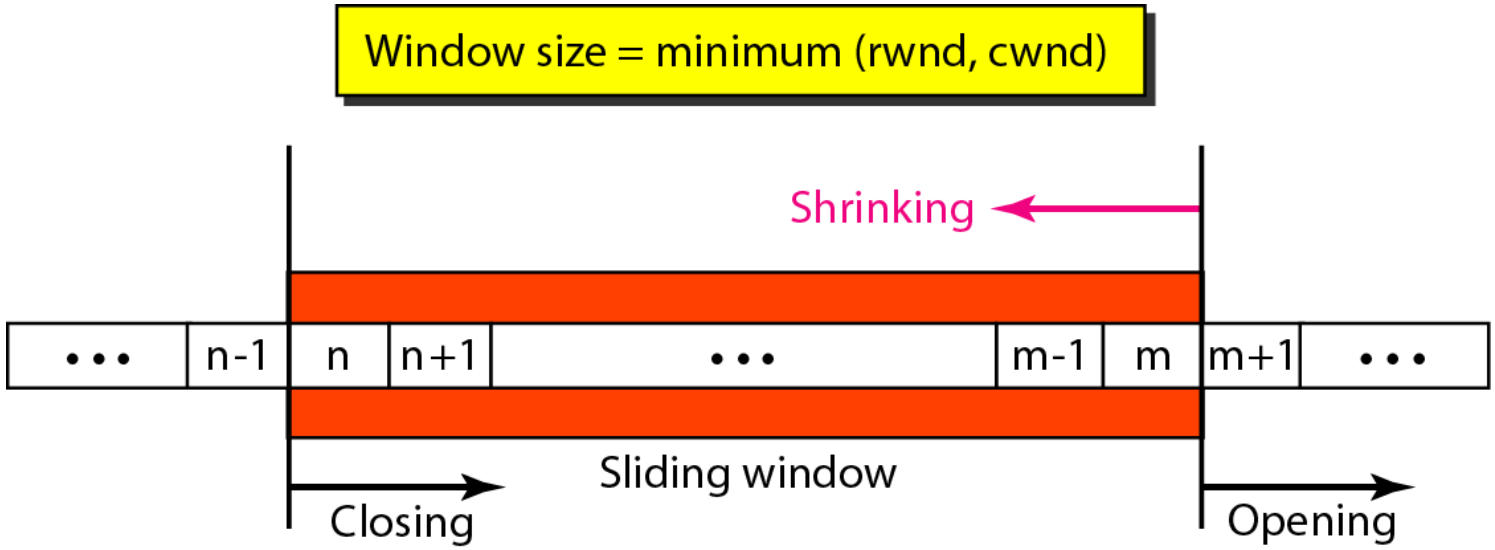
- An IP datagram is carrying a TCP segment destined for address 130.14.16.17/16. The destination port address is corrupted, and it arrives at destination 130.14.16.19/16. How does the receiving TCP react to this error?

- In TCP, if the value of HLEN is 0111, how many bytes of option are included in the segment?

Flow Control

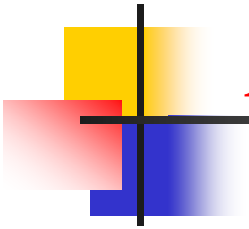
- The *receiver window* is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded.
- The congestion window is a value determined by the network to avoid congestion

Figure 23.22 *Sliding window*



Note

A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data.
TCP sliding windows are byte-oriented.

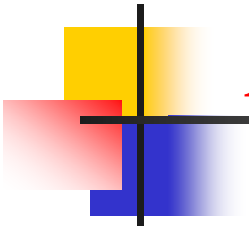


Example 23.4

What is the value of the receiver window (rwnd) for host A if the receiver, host B, has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?

Solution

The value of rwnd = 5000 – 1000 = 4000. Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A.

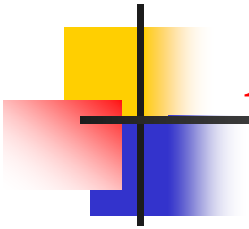


Example 23.5

What is the size of the window for host A if the value of rwnd is 3000 bytes and the value of cwnd is 3500 bytes?

Solution

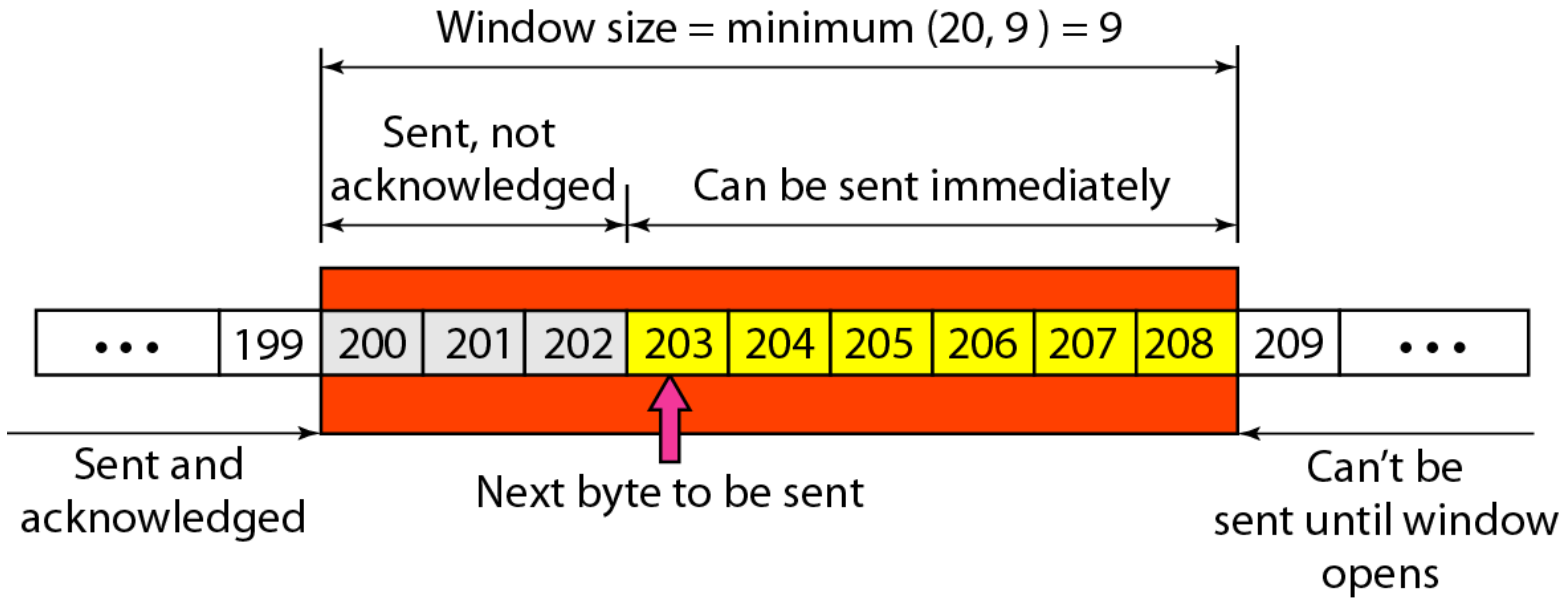
The size of the window is the smaller of rwnd and cwnd, which is 3000 bytes.



Example 23.6

Figure 23.23 shows an unrealistic example of a sliding window. The sender has sent bytes up to 202. We assume that cwnd is 20 (in reality this value is thousands of bytes). The receiver has sent an acknowledgment number of 200 with an rwnd of 9 bytes (in reality this value is thousands of bytes). The size of the sender window is the minimum of rwnd and cwnd, or 9 bytes. Bytes 200 to 202 are sent, but not acknowledged. Bytes 203 to 208 can be sent without worrying about acknowledgment. Bytes 209 and above cannot be sent.

Figure 23.23 *Example 23.6*



Note

Some points about TCP sliding windows:

- The size of the window is the lesser of rwnd and cwnd.
- The source does not have to send a full window's worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.
- The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

Error Control

1. 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.

cknowledgement

Note

Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged..

Retransmission

Note

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.

Note

No retransmission timer is set for an ACK segment.

Note

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

Figure 23.24 *Normal operation*

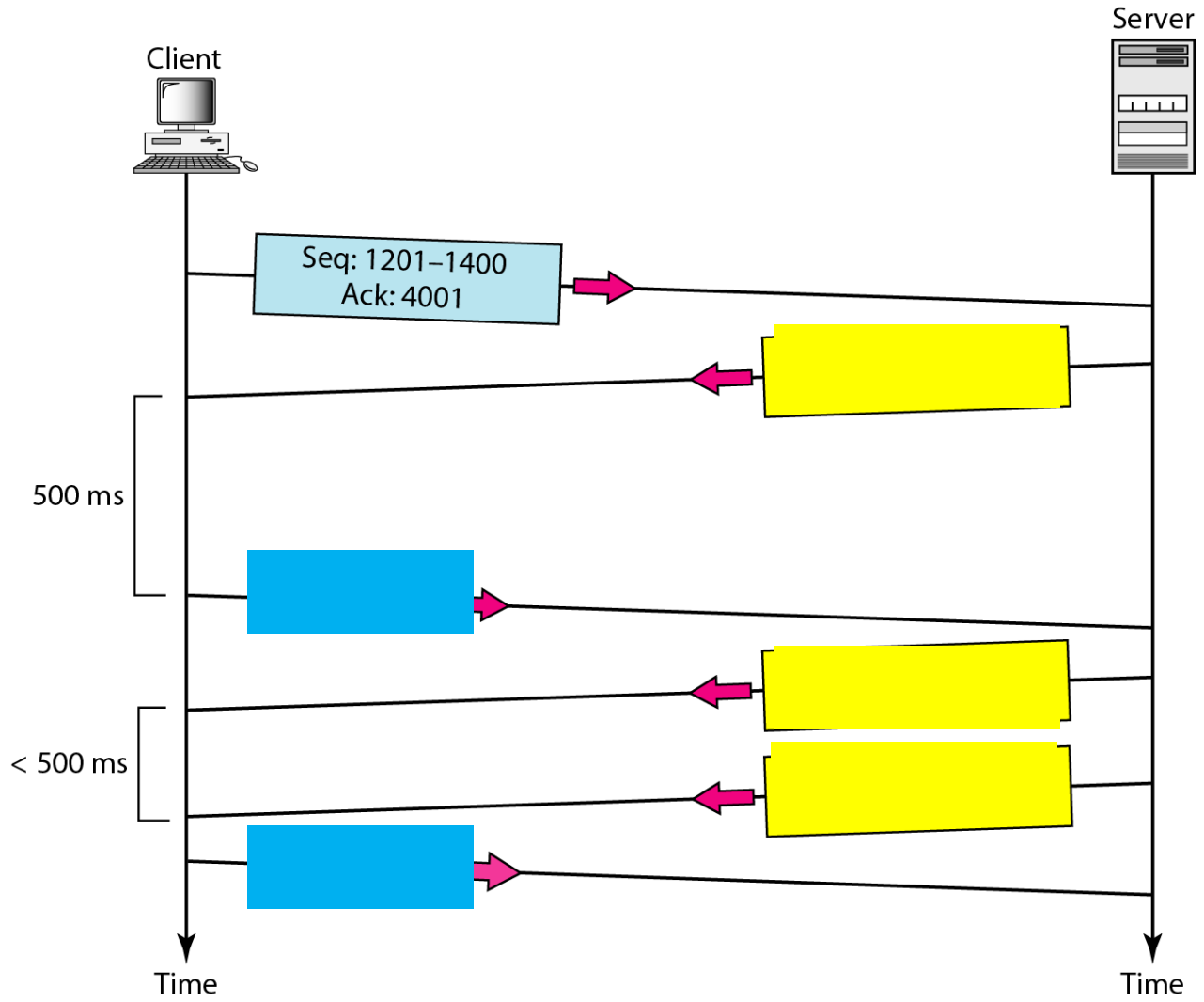
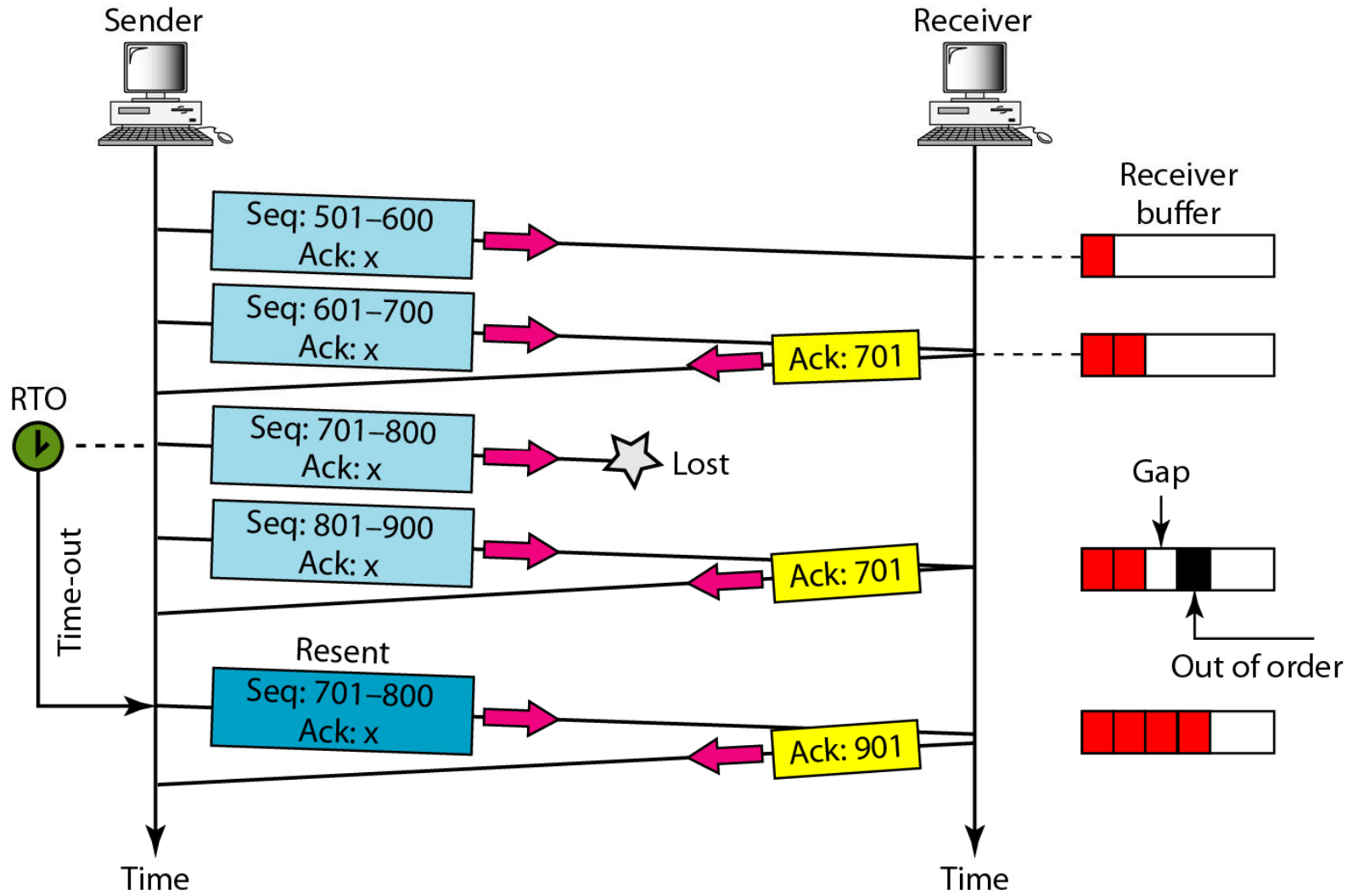


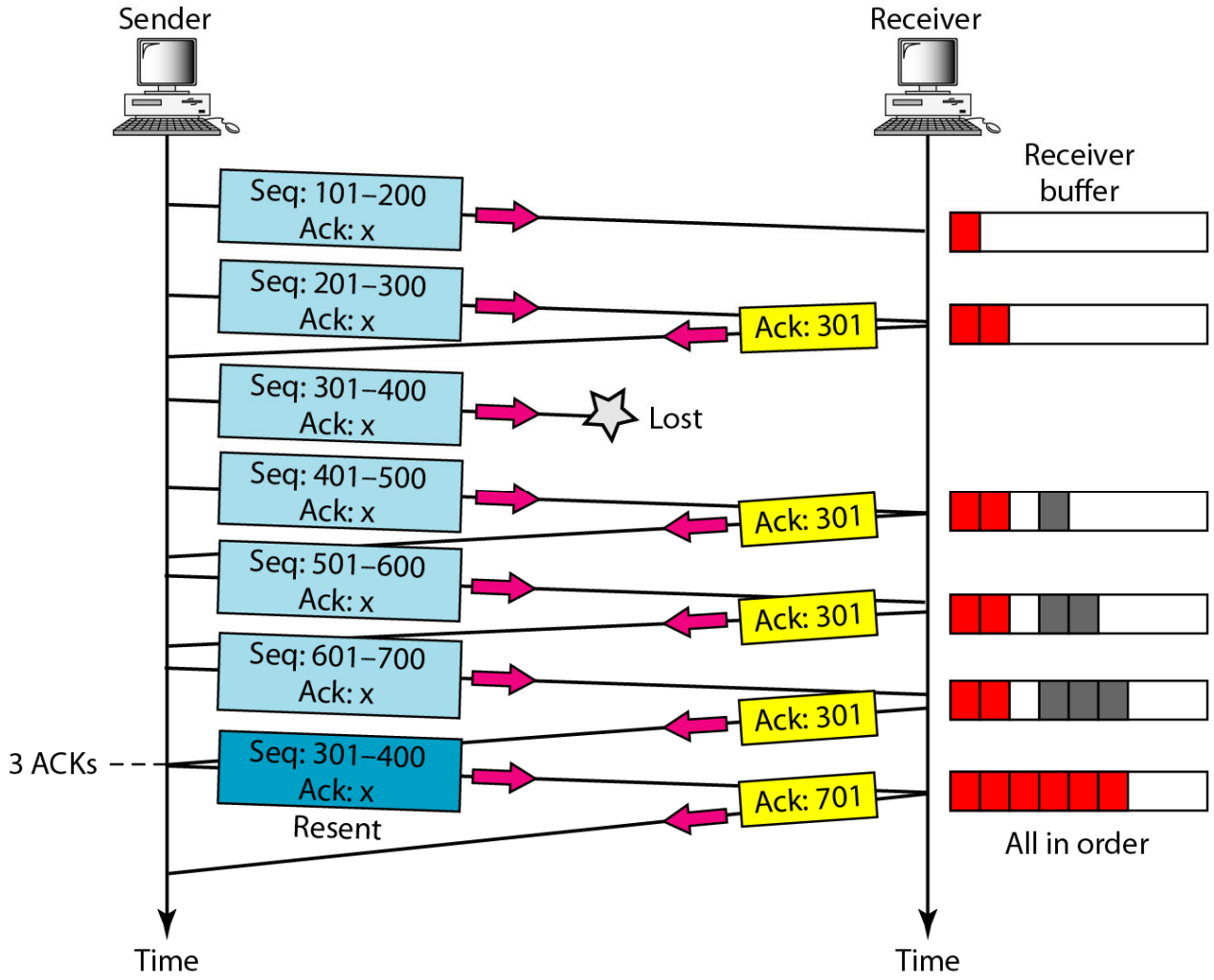
Figure 23.25 *Lost segment*



Note

The receiver TCP delivers only ordered data to the process.

Figure 23.26 *Fast retransmission*



- What can you say about the TCP segment in which the value of the control field is one of the following?
- a. 000000
- b. 000001
- c. 010001

a. None of the control bits are set. The segment is part of a data transmission without piggybacked acknowledgment.

b. The *FIN* bit is set. This is a FIN segment request to terminate the connection.

c. The *ACK* and the *FIN* bits are set. This is a *FIN+ACK* in response to a received *FIN* segment.

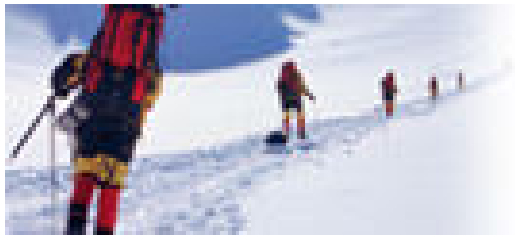
- To make the initial sequence number a random number, most systems start the counter at 1 during bootstrap and increment the counter by 64,000 every 0.5 s. How long does it take for the counter to wrap around?

Every second the counter is incremented by 64,000 × 2 = **128,000**. The sequence number field is 32 bits long and can hold only $2^{32}-1$. So it takes $(2^{32}-1)/(128{,}000)$ seconds or **33,554** seconds.

23.99

- TCP is sending data at 1 Mbyte/s. If the sequence number starts with 7000, how long does it take before the sequence number goes back to zero?

The largest number in the sequence number field is $2^{32} - 1$. If we start at 7000, it takes $[(2^{32} - 1) - 7000] / 1{,}000{,}000 = \mathbf{4295}$ seconds.



Data Communications and Networking Fourth Edition

Forouzan

Chapter 24

Congestion Control and Quality of Service

24-1 DATA TRAFFIC

The main focus of congestion control and quality of service is data traffic. In congestion control we try to avoid traffic congestion. In quality of service, we try to create an appropriate environment for the traffic. So, before talking about congestion control and quality of service, we discuss the data traffic itself.

Topics discussed in this section:

Traffic Descriptor
Traffic Profiles

Figure 24.1 *Traffic descriptors*

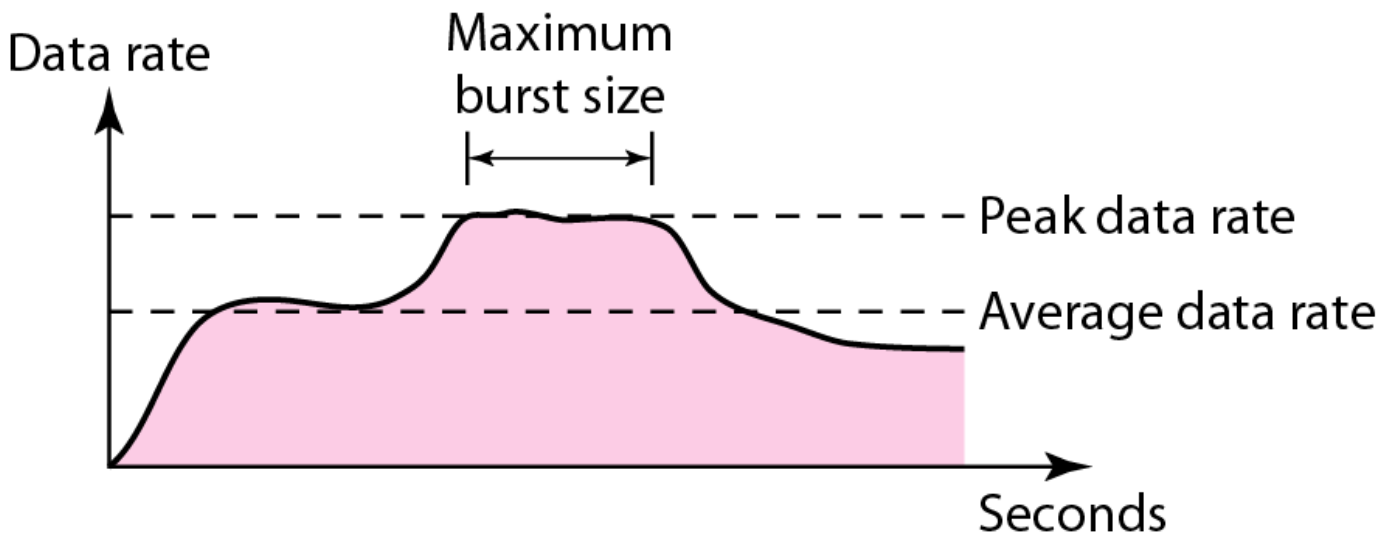
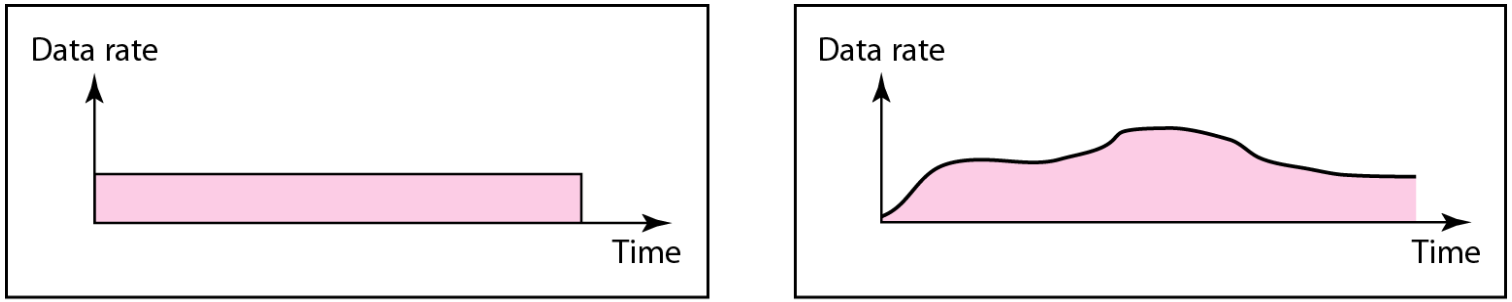


Figure 24.2 *Three traffic profiles*



a. Constant bit rate

b. Variable bit rate

c. Bursty

24-2 CONGESTION

Congestion in a network may occur if the load on the network—the number of packets sent to the network—is greater than the capacity of the network—the number of packets a network can handle. Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

Topics discussed in this section:

Network Performance

Figure 24.3 *Queues in a router*

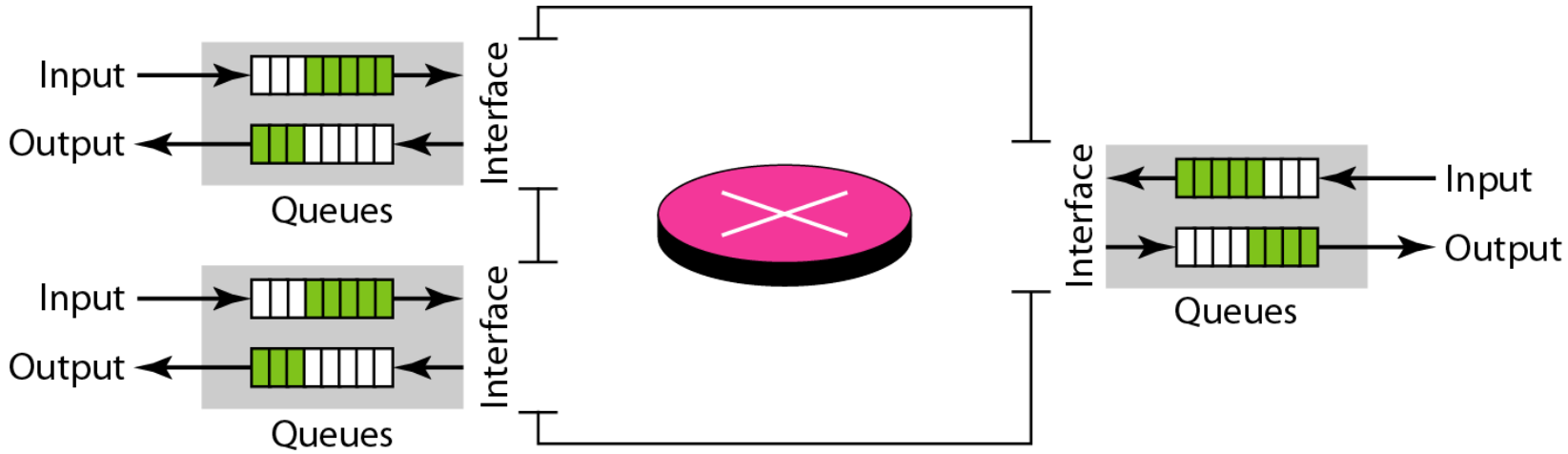
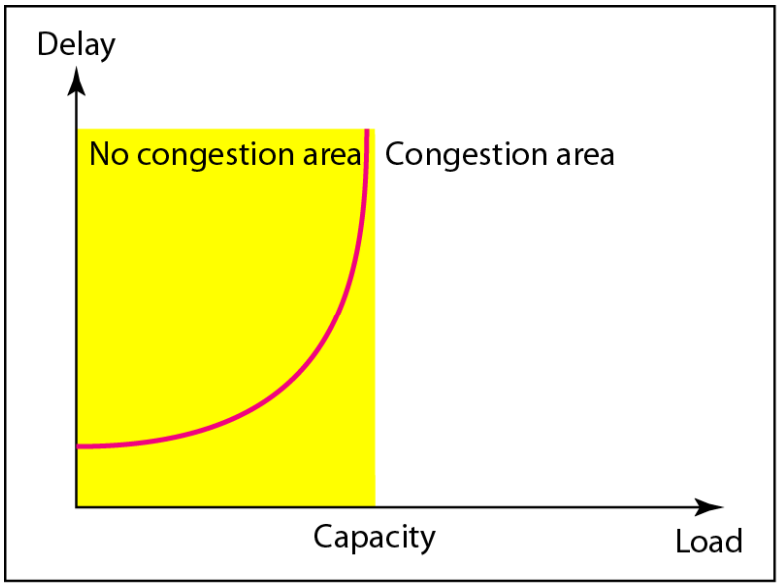
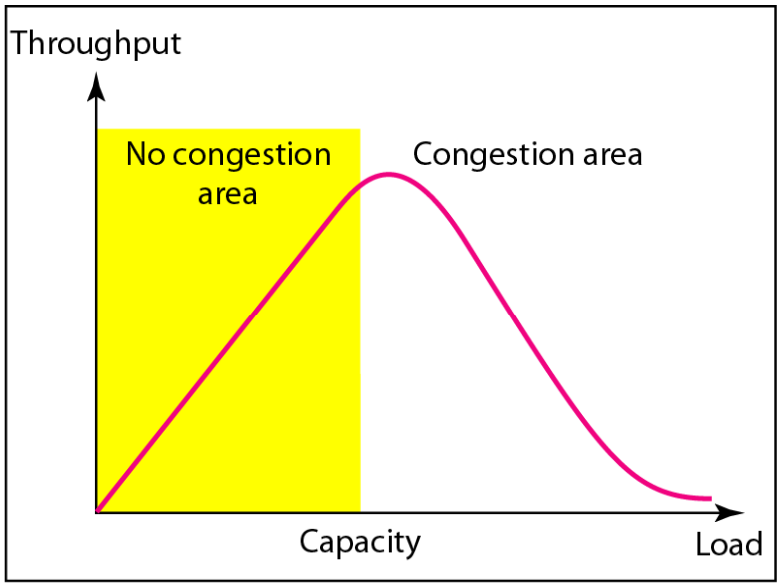


Figure *Packet delay and throughput as functions of load*



a. Delay as a function of load



b. Throughput as a function of load

24-3 CONGESTION CONTROL

Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal).

Topics discussed in this section:

Open-Loop Congestion Control
Closed-Loop Congestion Control

Figure 24.5 *Congestion control categories*

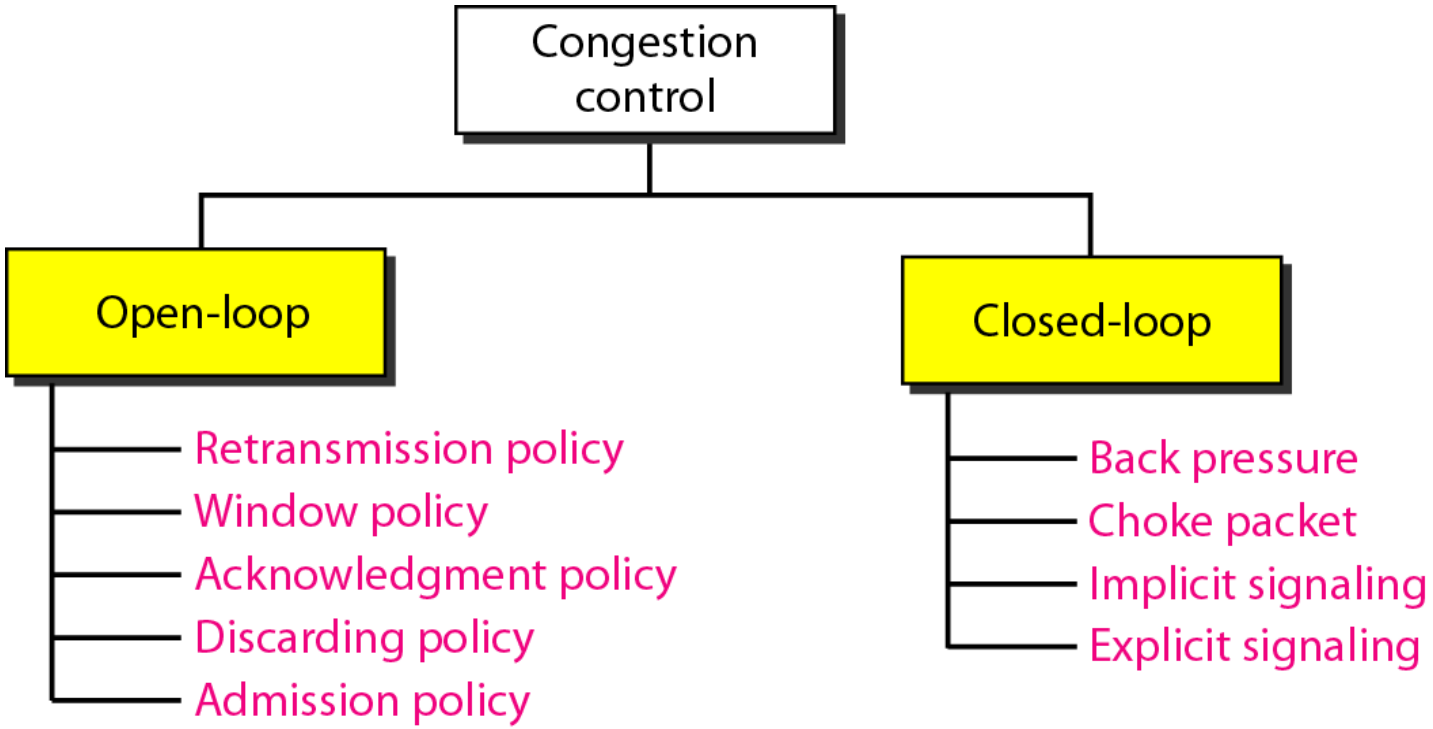


Figure 24.6 *Backpressure method for alleviating congestion*

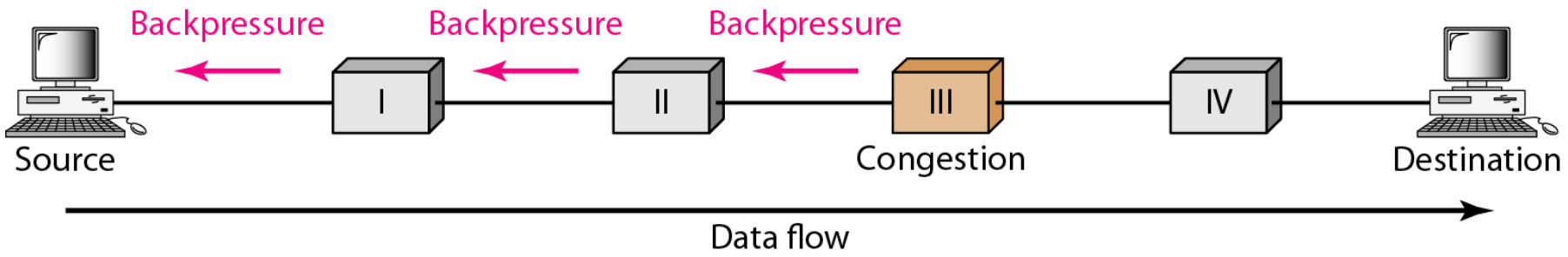
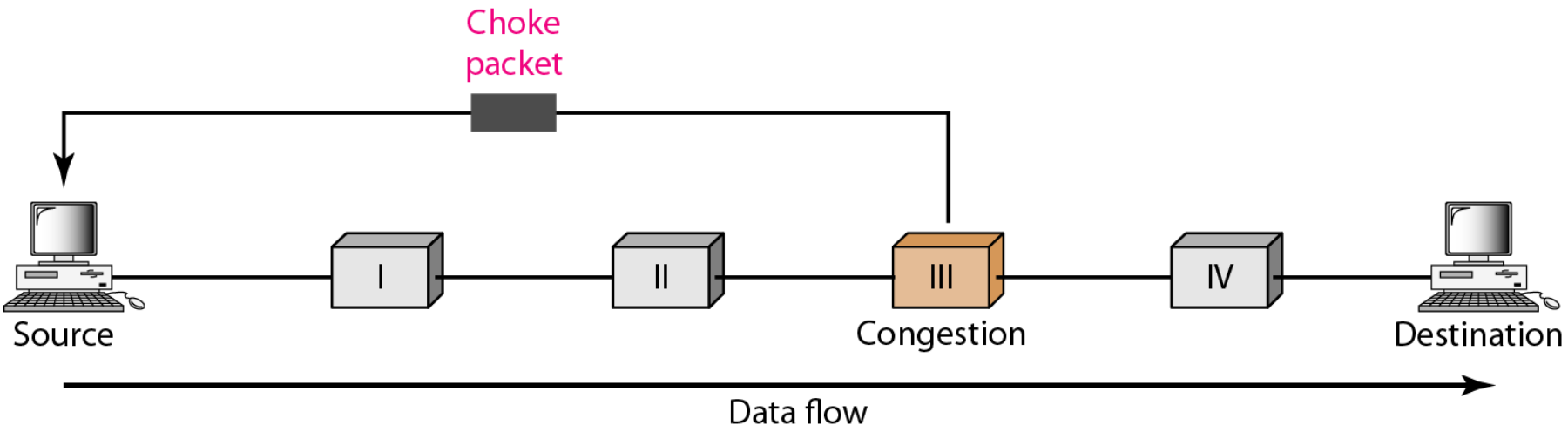


Figure 24.7 *Choke packet*



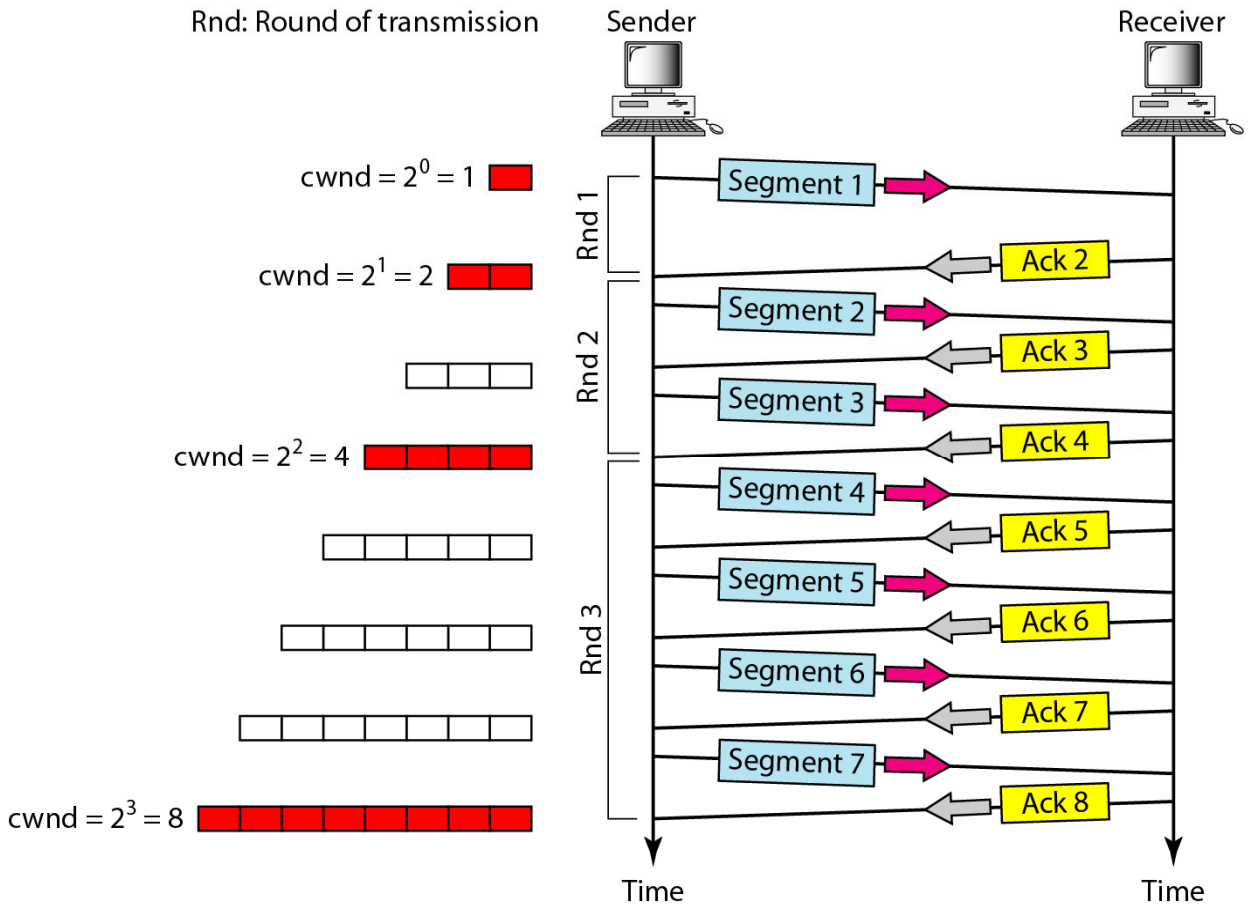
24-4 TWO EXAMPLES

To better understand the concept of congestion control, let us give two examples: one in TCP and the other in Frame Relay.

Topics discussed in this section:

Congestion Control in TCP
Congestion Control in Frame Relay

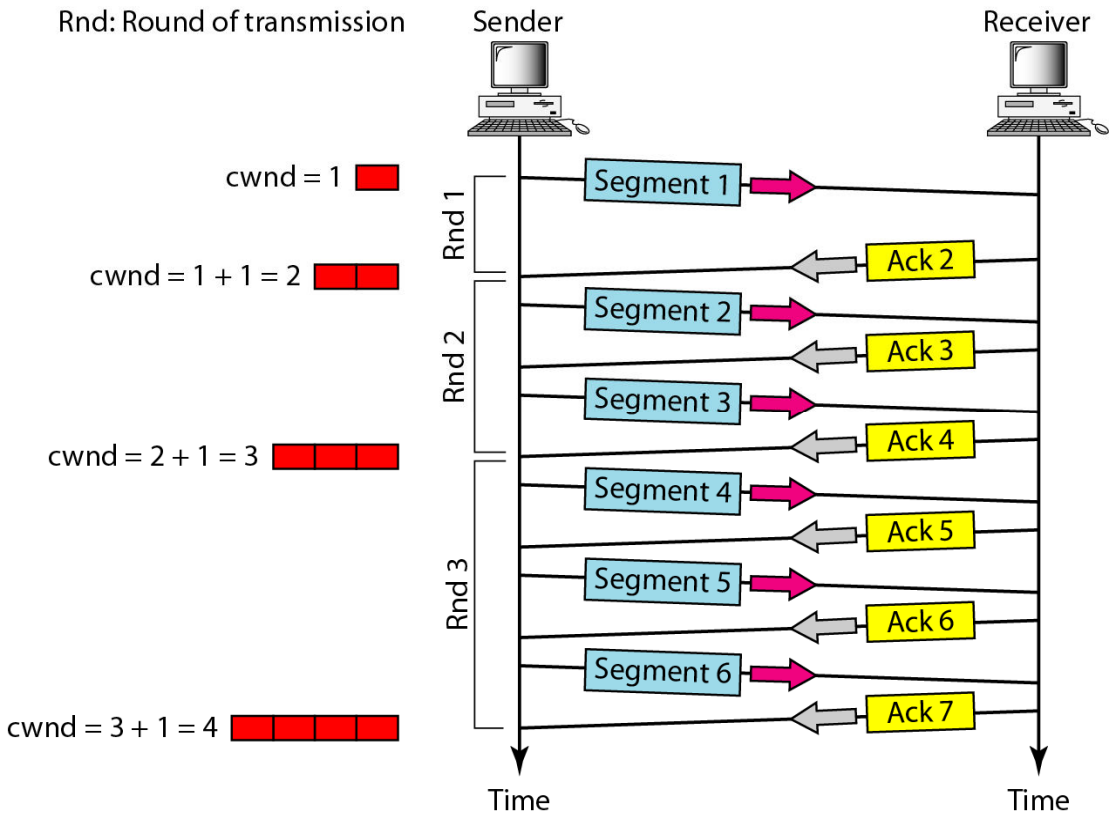
Figure 24.8 *Slow start, exponential increase*



Note

In the slow-start algorithm, the size of the congestion window increases exponentially until it reaches a threshold.

Figure 24.9 *Congestion avoidance, additive increase*



Note

In the congestion avoidance algorithm, the size of the congestion window increases additively until congestion is detected.

Note

An implementation reacts to congestion detection in one of the following ways:

- **If detection is by time-out, a new slow start phase starts.**
- **If detection is by three ACKs, a new congestion avoidance phase starts.**

Figure 24.10 *TCP congestion policy summary*

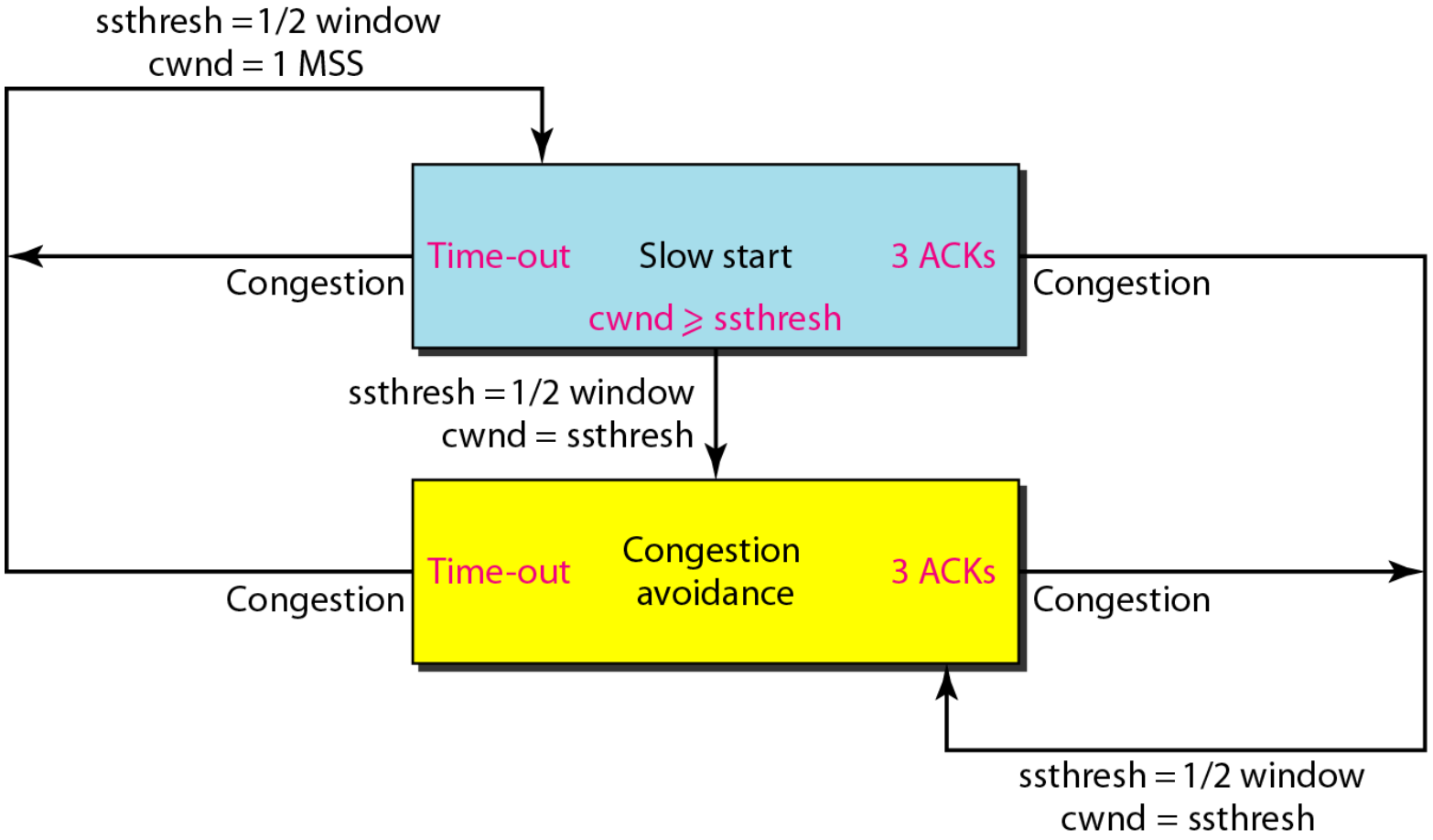


Figure 24.11 *Congestion example*

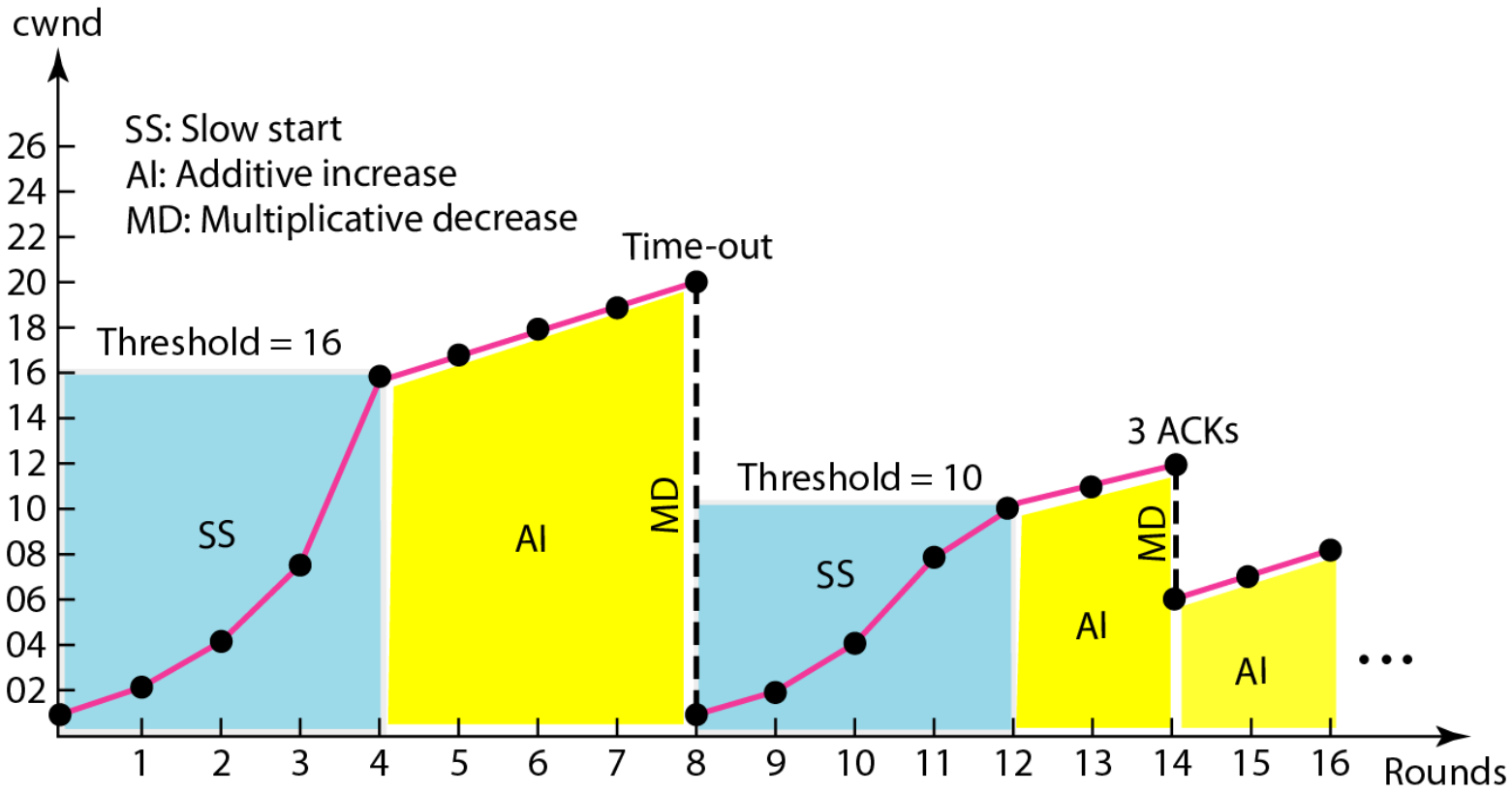


Figure 24.12 *BECN*

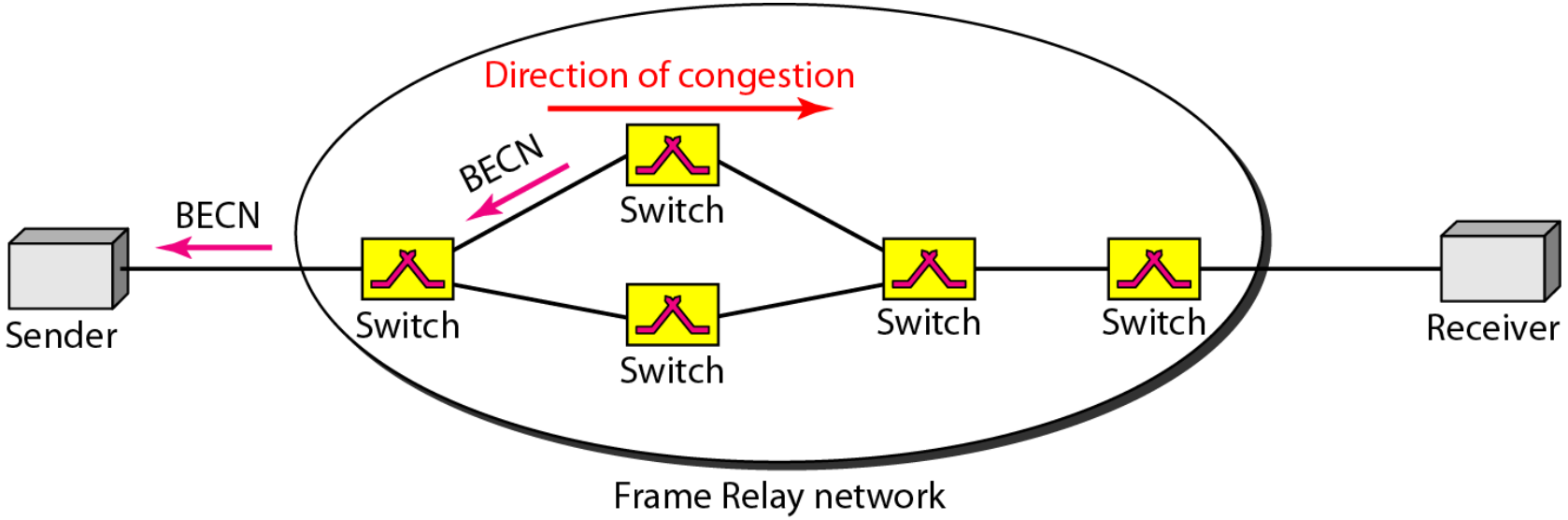


Figure 24.13 *FECN*

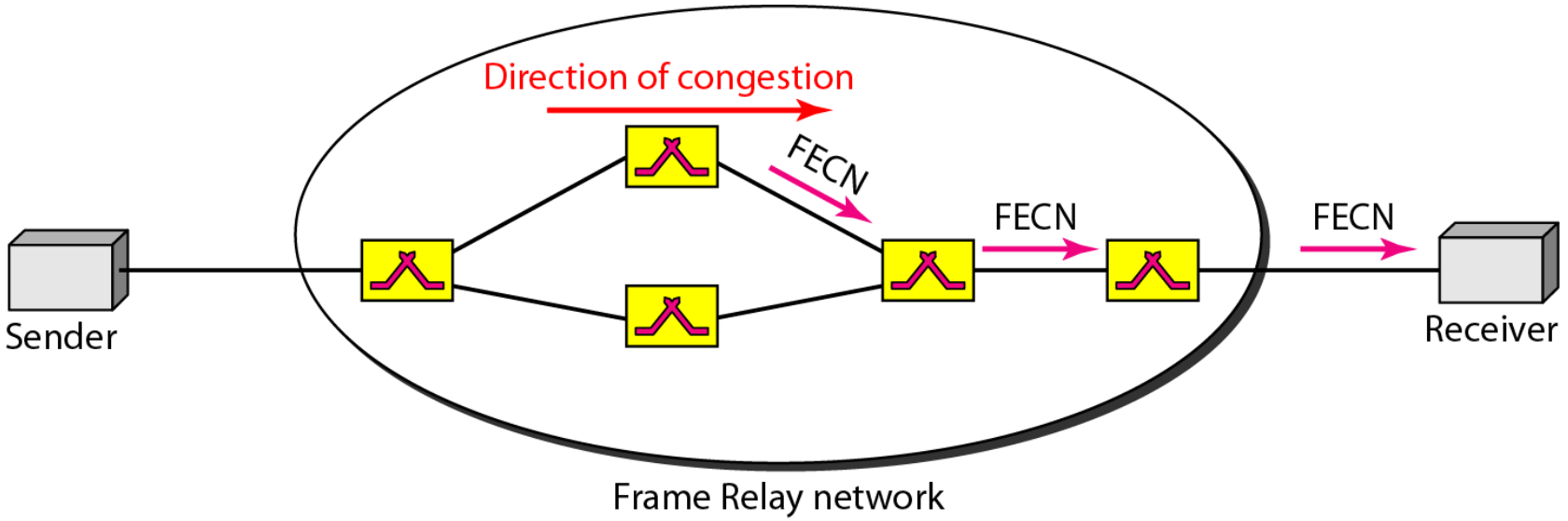
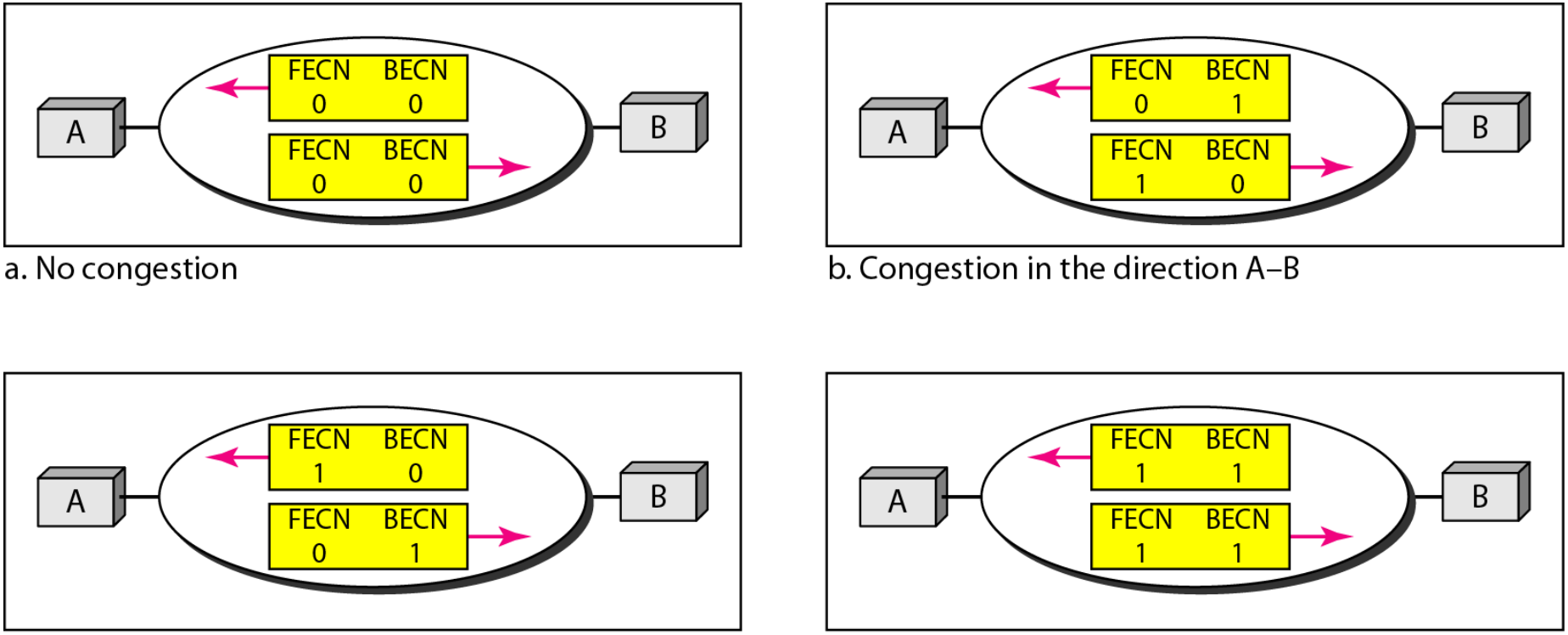


Figure 24.14 *Four cases of congestion*



a. No congestion

b. Congestion in the direction A–B

c. Congestion in the direction B–A

d. Congestion in both directions

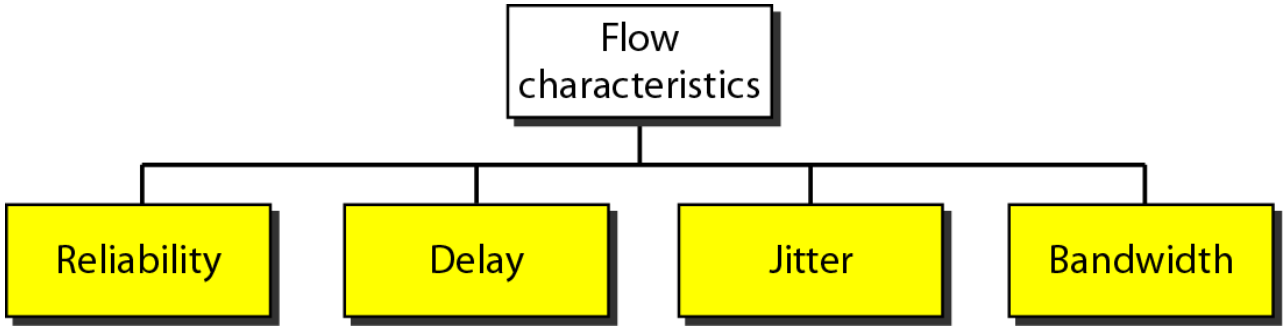
24-5 QUALITY OF SERVICE

Quality of service (QoS) is an internetworking issue that has been discussed more than defined. We can informally define quality of service as something a flow seeks to attain.

Topics discussed in this section:

Flow Characteristics
Flow Classes

Figure 24.15 *Flow characteristics*



24-6 TECHNIQUES TO IMPROVE QoS

In Section 24.5 we tried to define QoS in terms of its characteristics. In this section, we discuss some techniques that can be used to improve the quality of service. We briefly discuss four common methods: scheduling, traffic shaping, admission control, and resource reservation.

Topics discussed in this section:

Scheduling
Traffic Shaping
Resource Reservation
Admission Control

Figure 24.16 *FIFO queue*

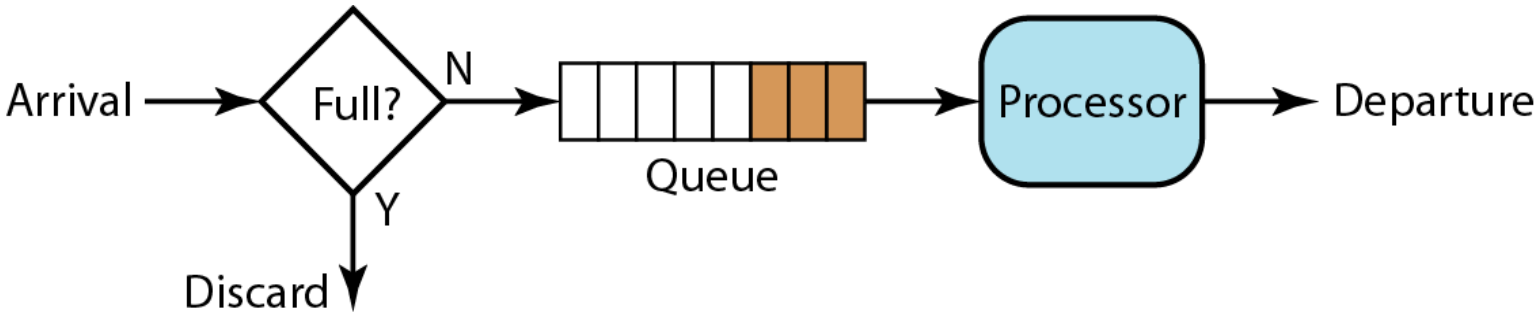


Figure 24.17 *Priority queuing*

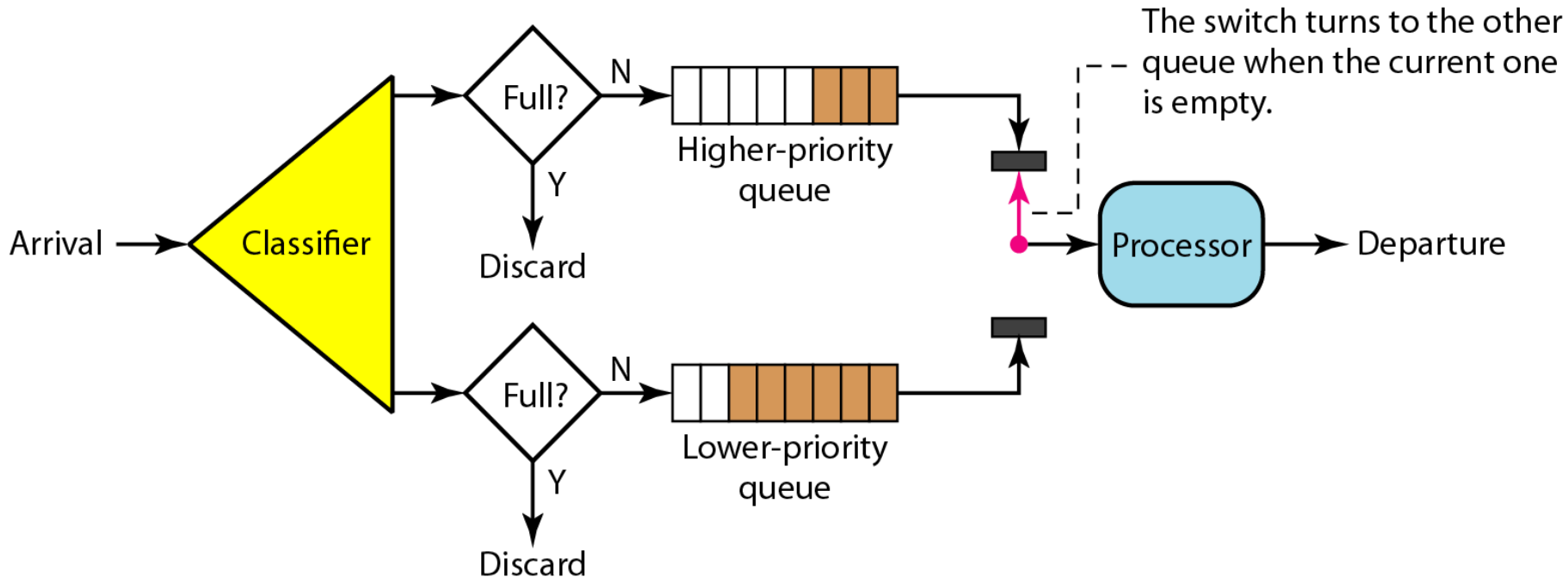


Figure 24.18 *Weighted fair queuing*

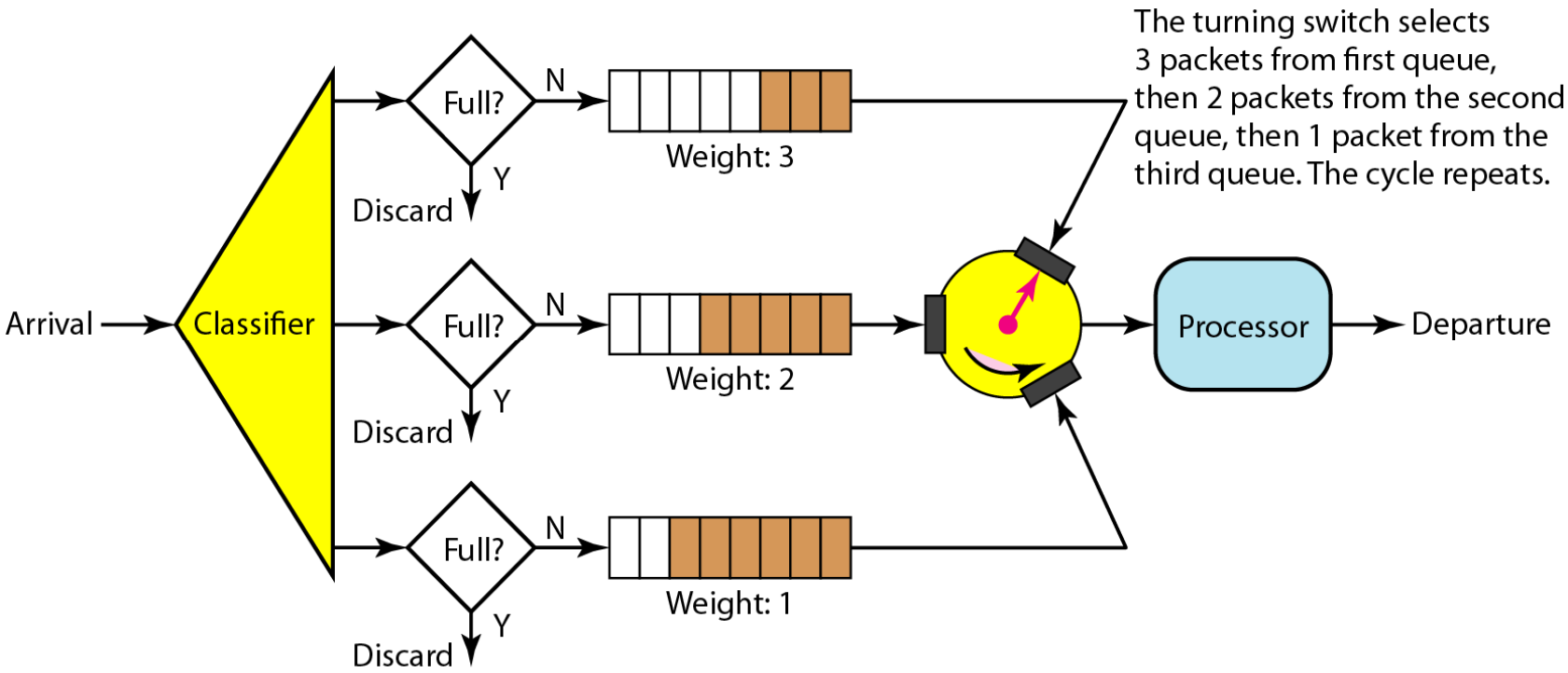


Figure 24.19 *Leaky bucket*

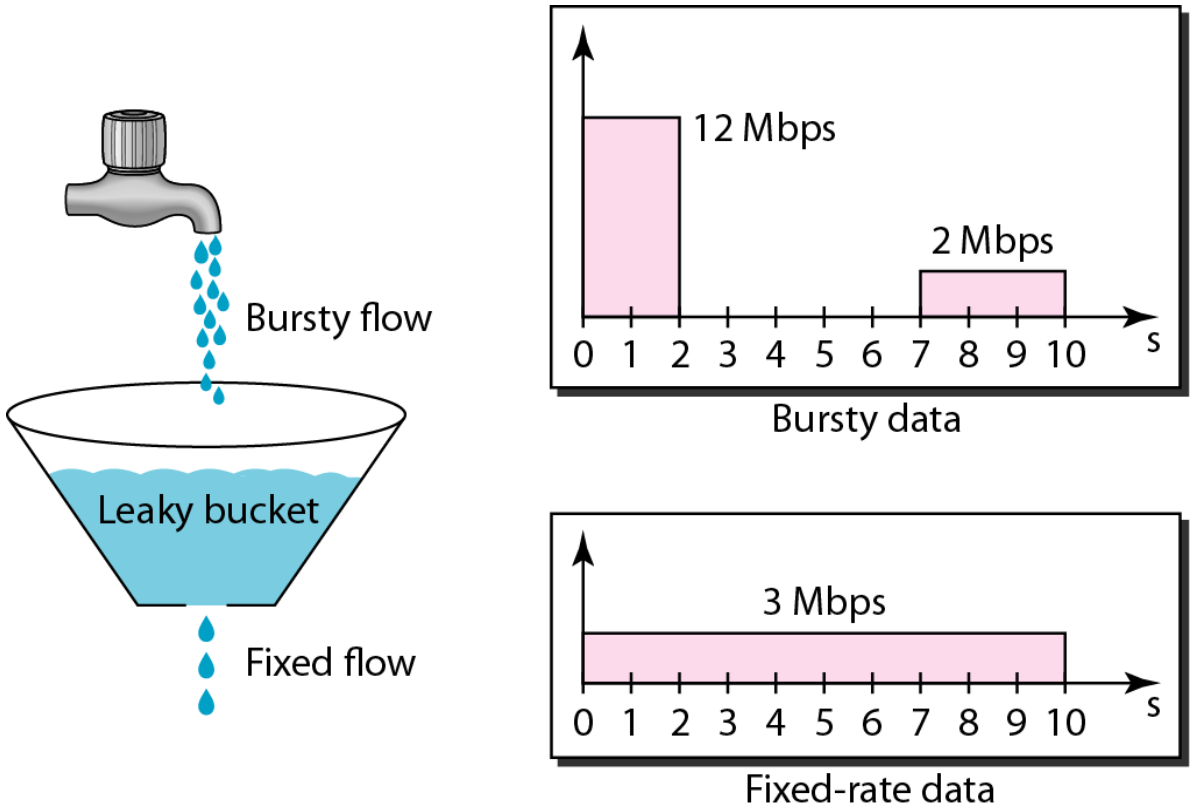
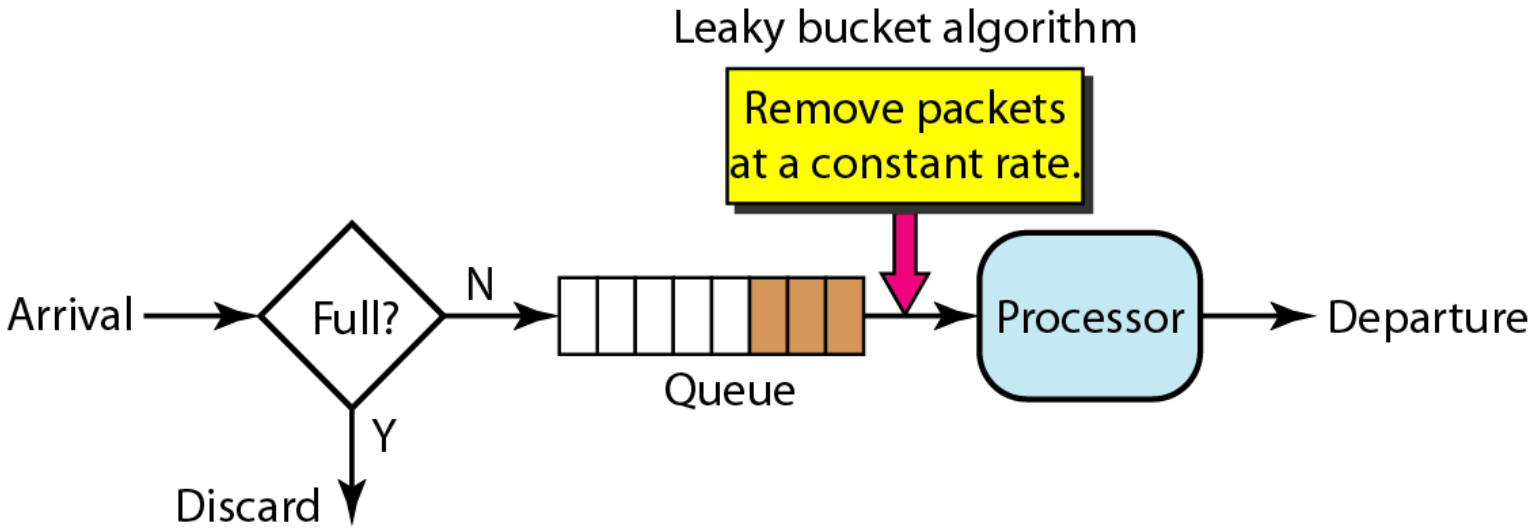


Figure 24.20 *Leaky bucket implementation*



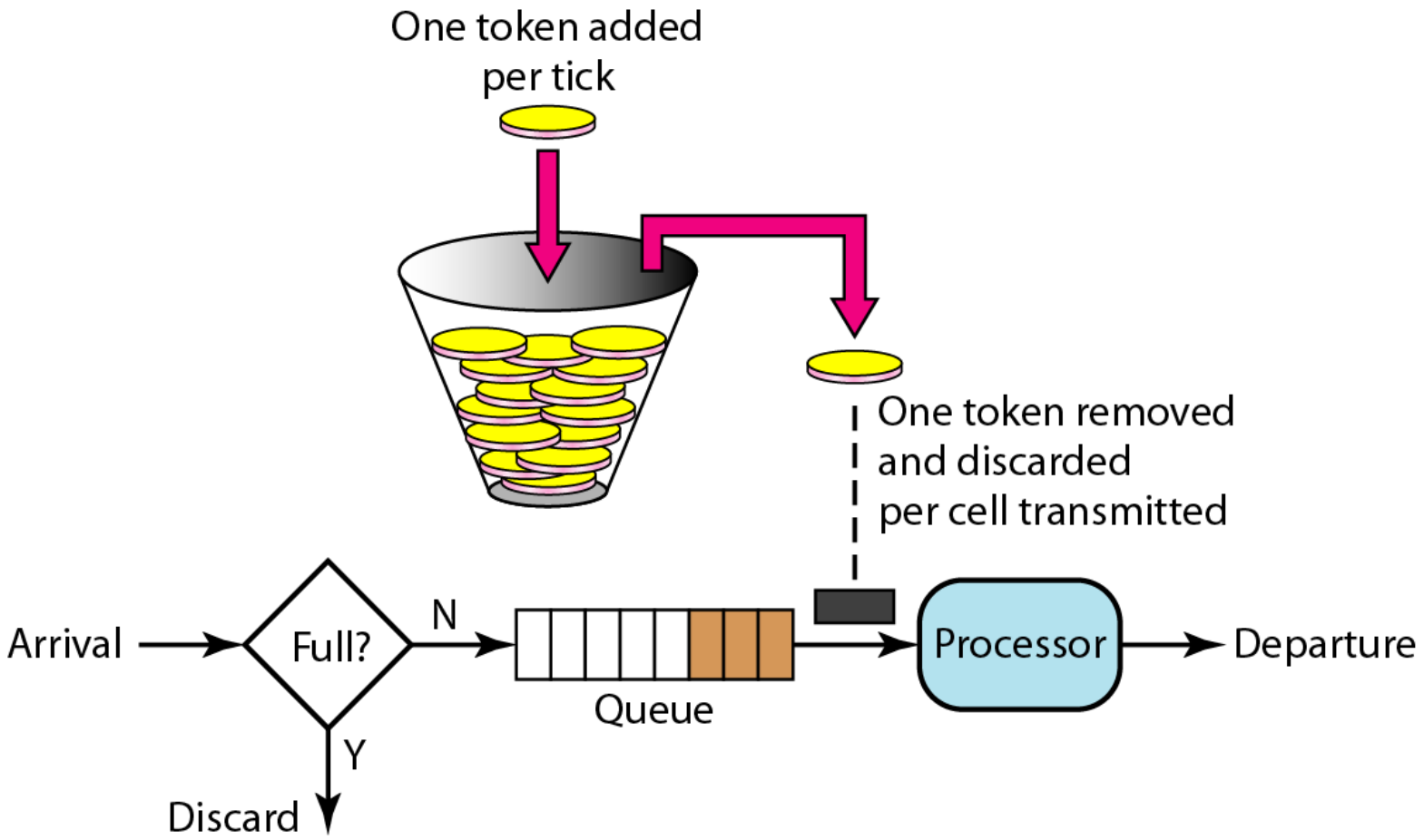
Note

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

Note

The token bucket allows bursty traffic at a regulated maximum rate.

Figure 24.21 *Token bucket*



24-7 INTEGRATED SERVICES

Two models have been designed to provide quality of service in the Internet: Integrated Services and Differentiated Services. We discuss the first model here.

Topics discussed in this section:

Signaling
Flow Specification
Admission
Service Classes
RSVP
Problems with Integrated Services

Note

Integrated Services is a flow-based QoS model designed for IP.

Figure 24.22 *Path messages*

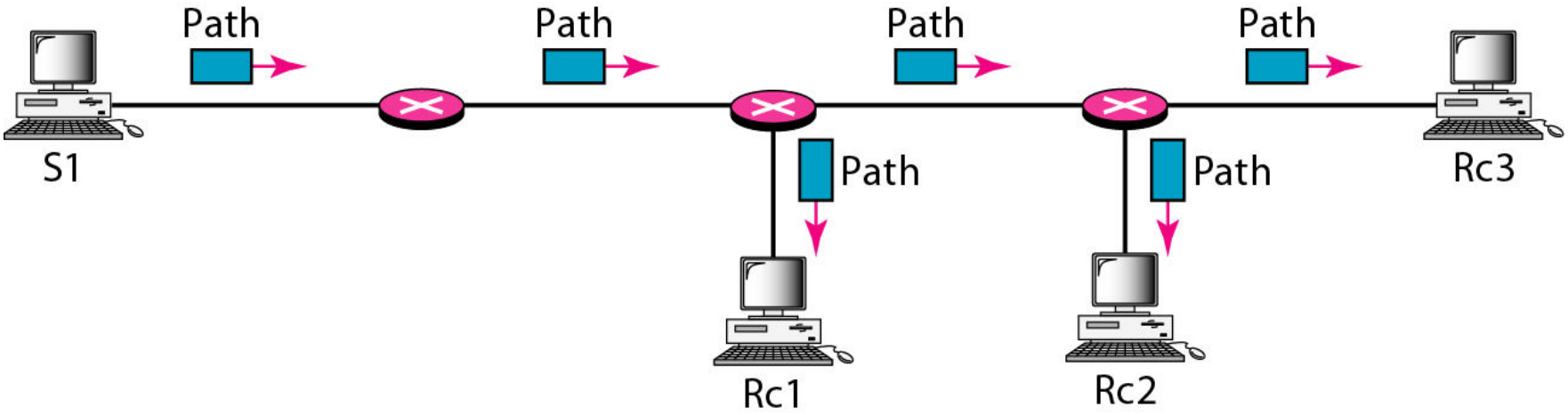


Figure 24.23 *Resv messages*

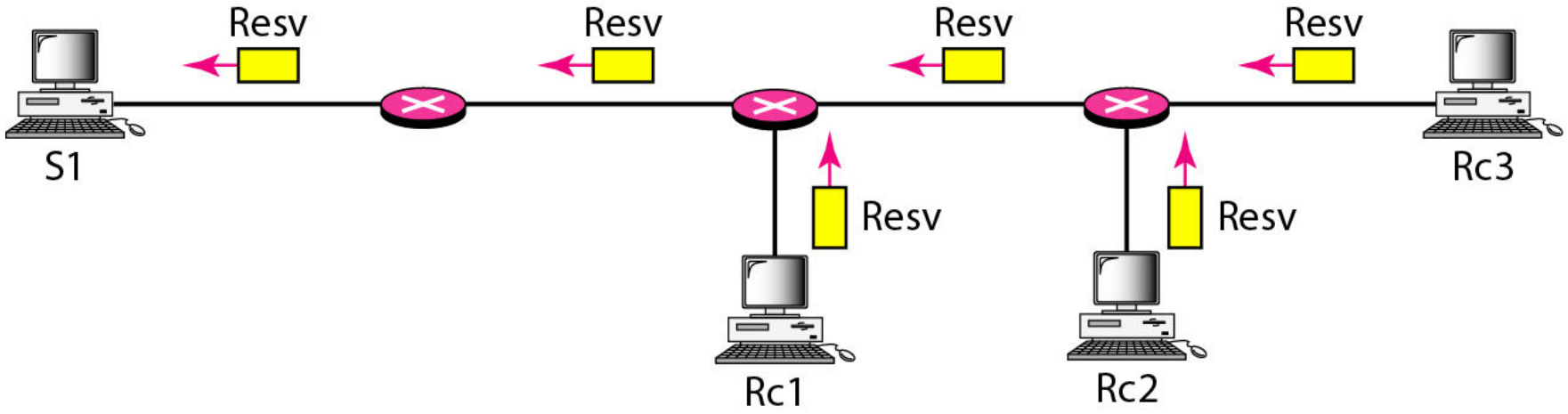


Figure 24.24 *Reservation merging*

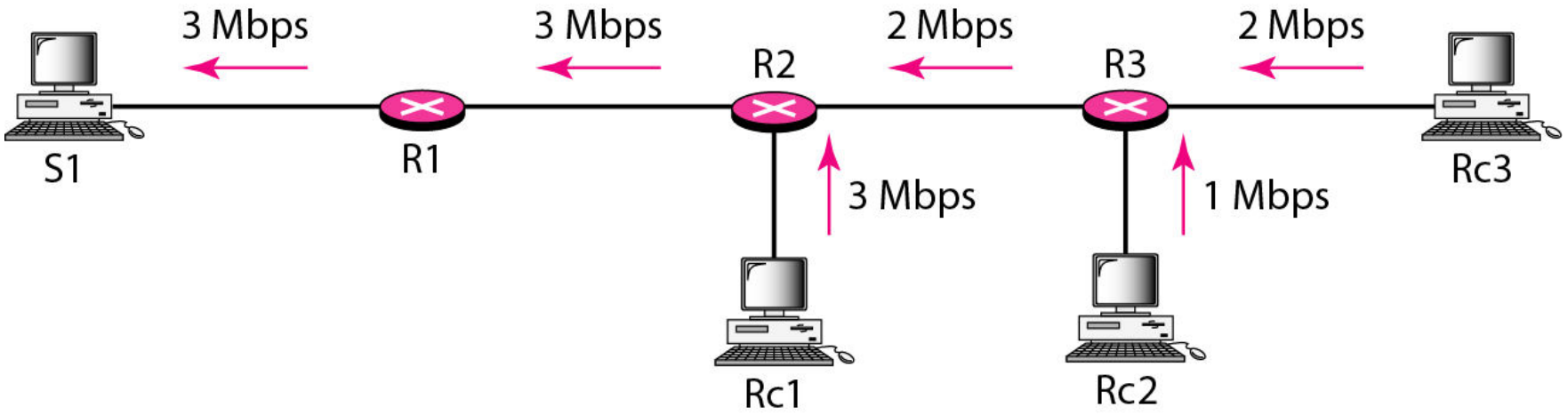
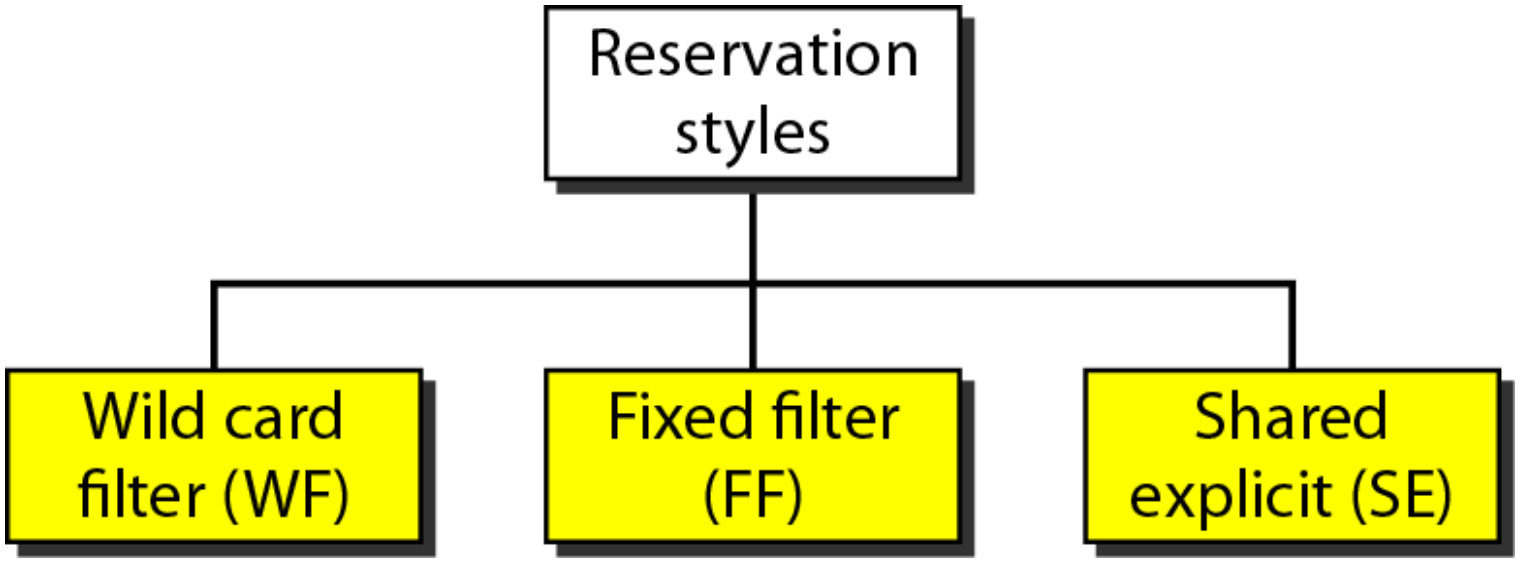


Figure 24.25 *Reservation styles*



24-8 DIFFERENTIATED SERVICES

Differentiated Services (DS or Diffserv) was introduced by the IETF (Internet Engineering Task Force) to handle the shortcomings of Integrated Services.

Topics discussed in this section:

DS Field

Note

Differentiated Services is a class-based QoS model designed for IP.

Figure 24.26 *DS field*

Figure 24.27 *Traffic conditioner*