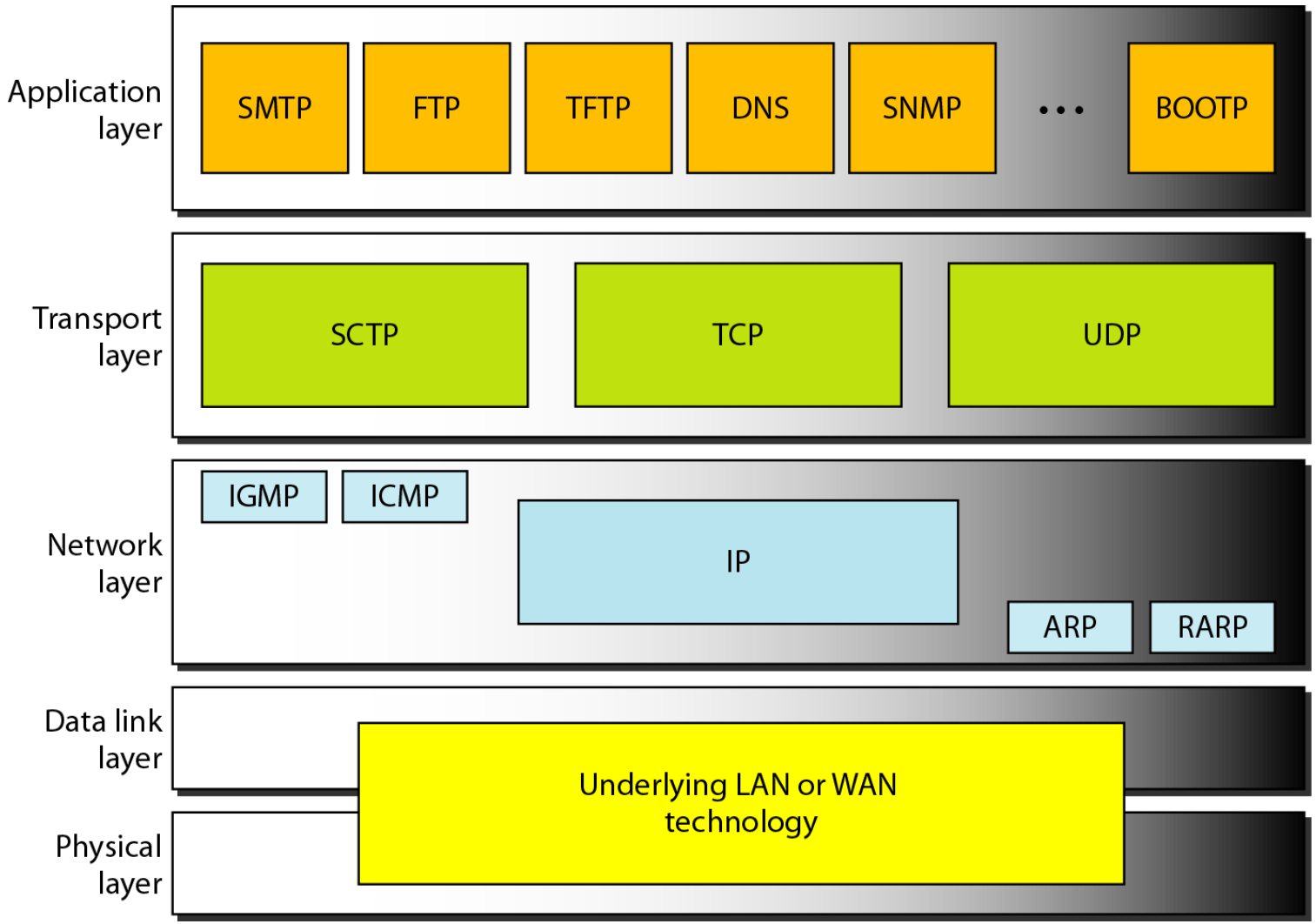
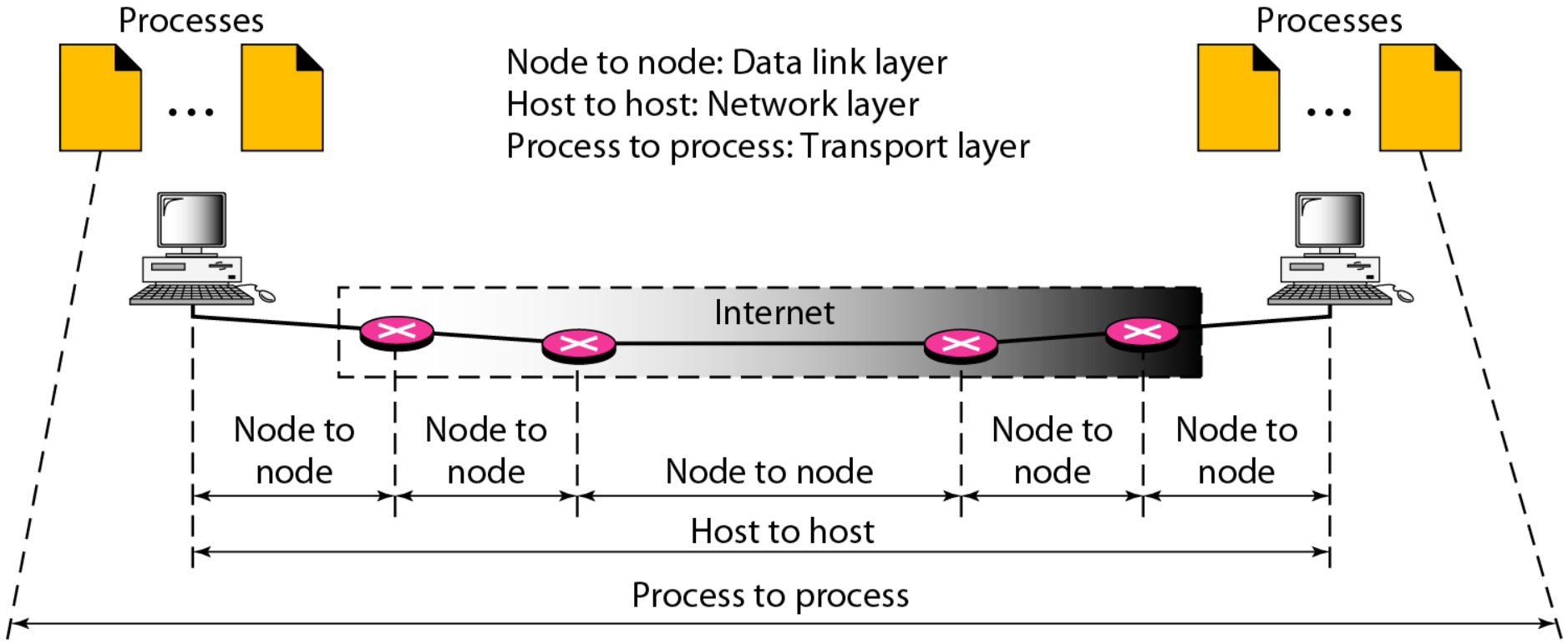
Transport Layer



Why we need transport layer?

- *Network layer is responsible for end to end (Host to Host) communication (IP address).*
- *There are several network application running in OS.*
- *How did NIC knows which packet belongs to which process/application?*

Types of data deliveries: Internet Stack

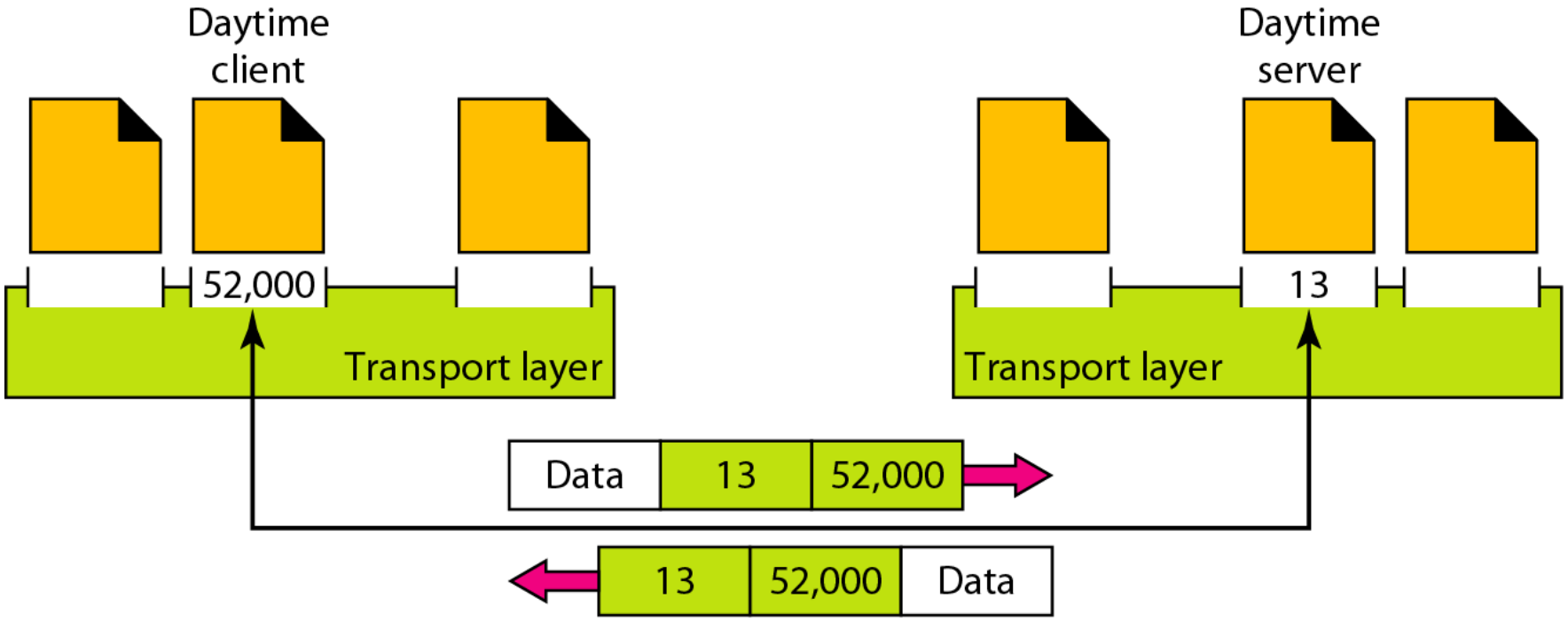


Continued..

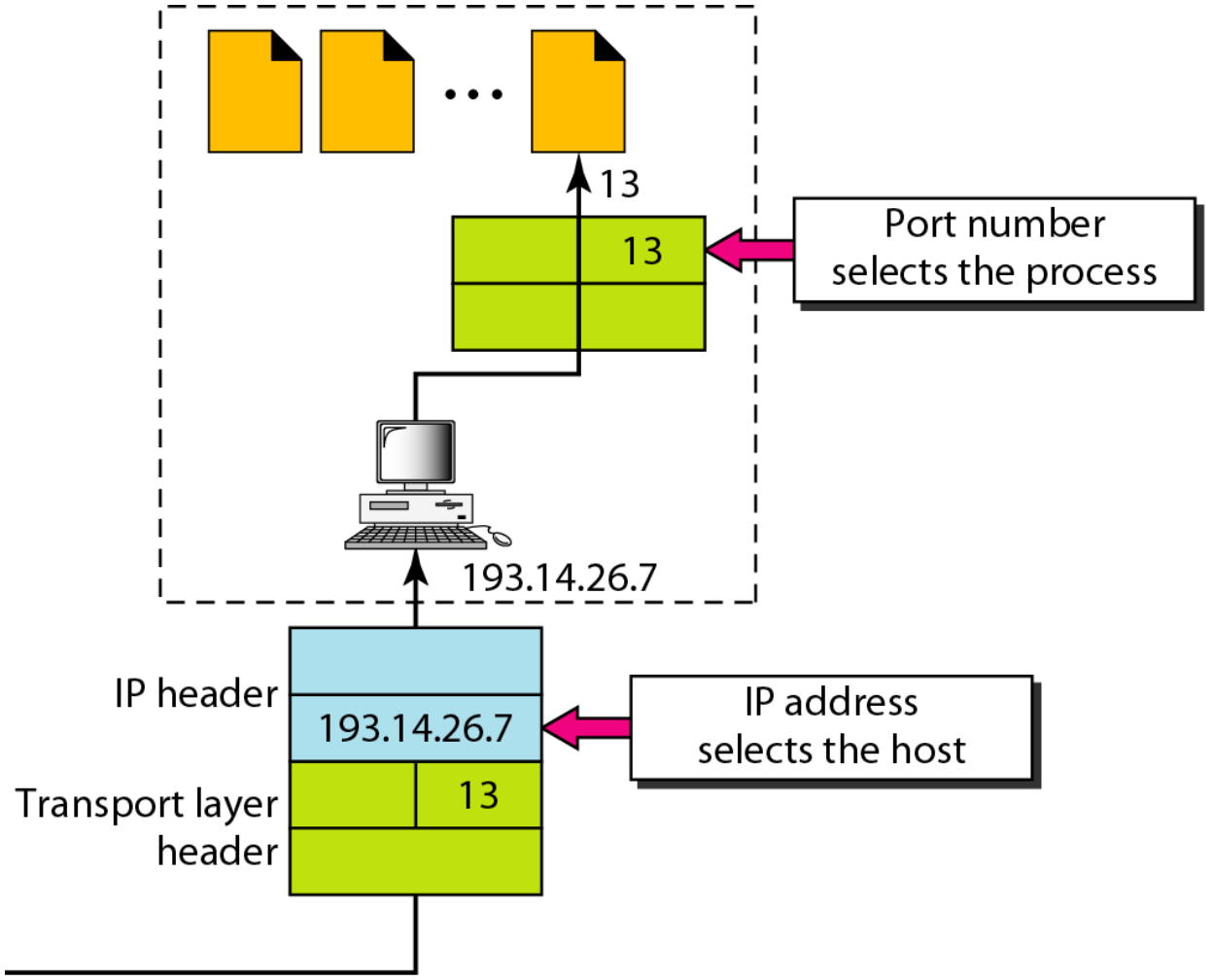
- The transport layer is responsible for process-to-process delivery—the delivery of a packet, part of a message, from one process to another.
- It provides *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP

How It delivers messages to specific process

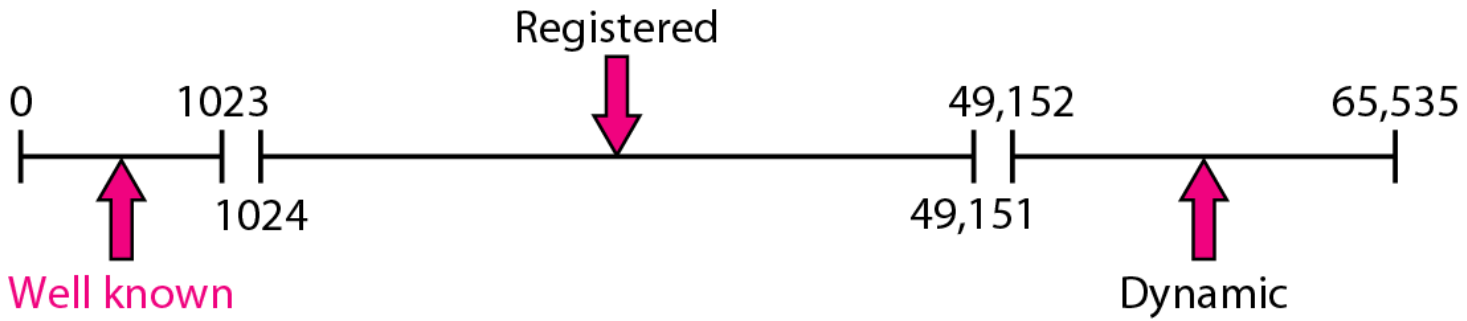
- Using Port address (16 bit)
- Ranges from 0 to 65,535



IP addresses versus port numbers



IANA ranges


Registered
0
1023
49,152
65,535
1024
49,151
Well known
Dynamic

Well-known ports or system ports are from 0 - 1023. The operating system uses them for processes that provide widely used types of network services.

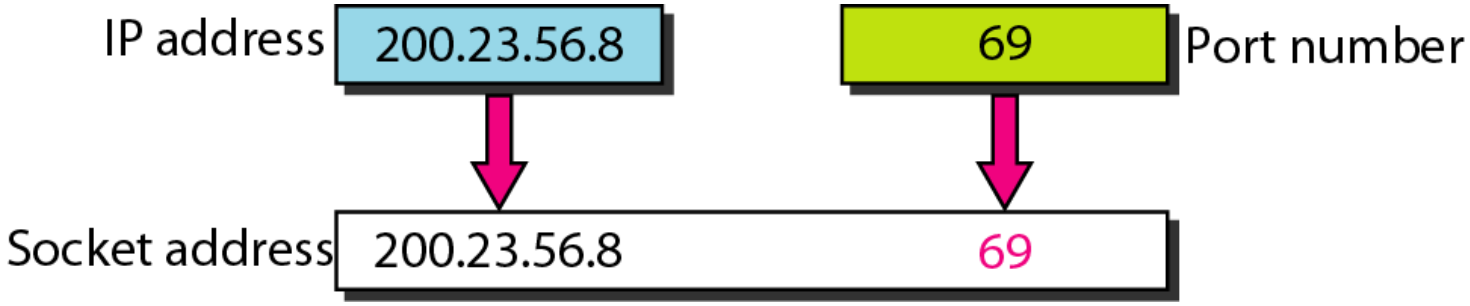
PROTOCOL	USE	PORT
HTTP	Web traffic	80
SMTP	Sending email	25
IMAP4	Fetching email	143
IMAPS	Fetching emial over TLS	993
POP3	Fetching email	110
FTP	File server	21
FTPS	File server of TLS /SSL	990
DNS	Domain name system	53

Registered ports have numbers from 1024 - 49151. They are for specific services that companies have registered with IANA (The Internet Assigned Numbers Authority oversees Internet Protocol-related symbols and numbers such as IP address allocation, media types, and port numbers.)

SERVICE	PORT
Skype	23399
MySQL	3306
BitTorrent	6902-6968
XBOX live	3074
Playstation network	3479/80
Apple remote desktop	3283
MS Terminal Server RDP	3389

Momentary ports have numbers from 49152-65535, they are used for temporary purposes (such as tabs in a browser) also known as dynamic or private ports. They are used for private, customized services, or for temporary purposes

Socket address (IP address + Port Address)



What is the use of socket?

- The socket mechanism provides a means of inter-process communication (IPC).
- Socket is basically an API for enabling communication between two end points.
- A **socket** is one endpoint of a **two way** communication link between two programs running on the network.

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.

Stream delivery

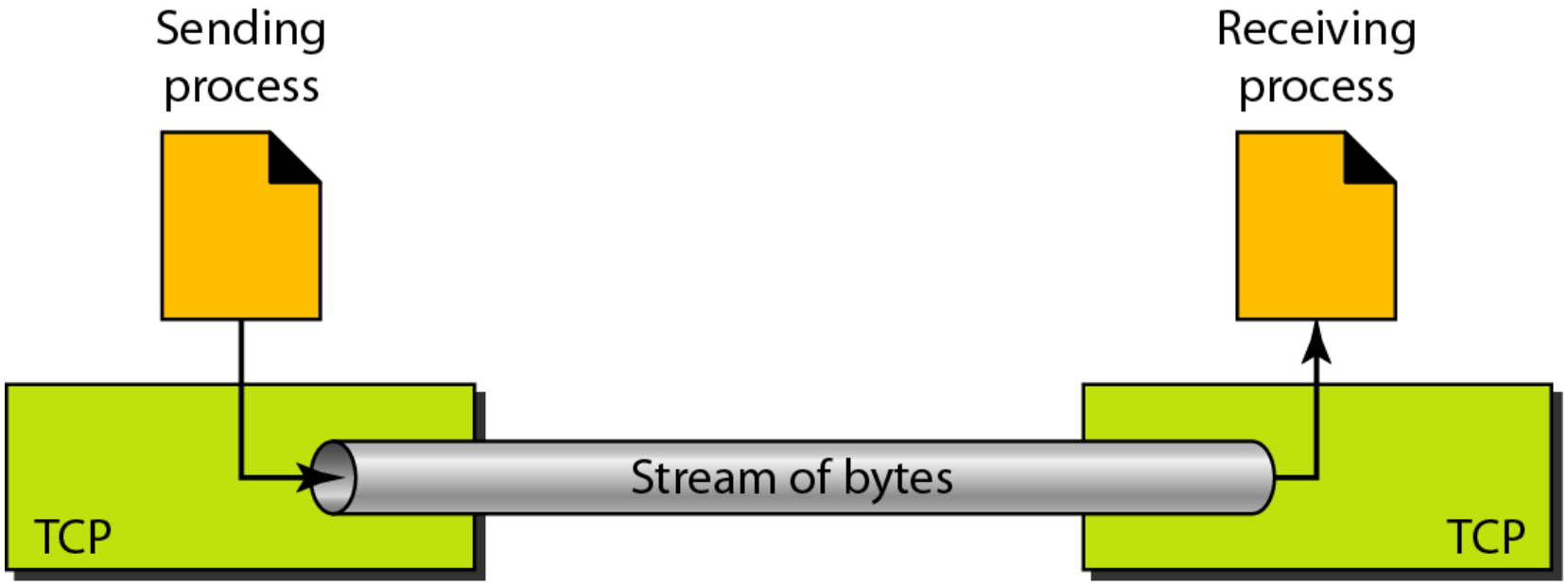
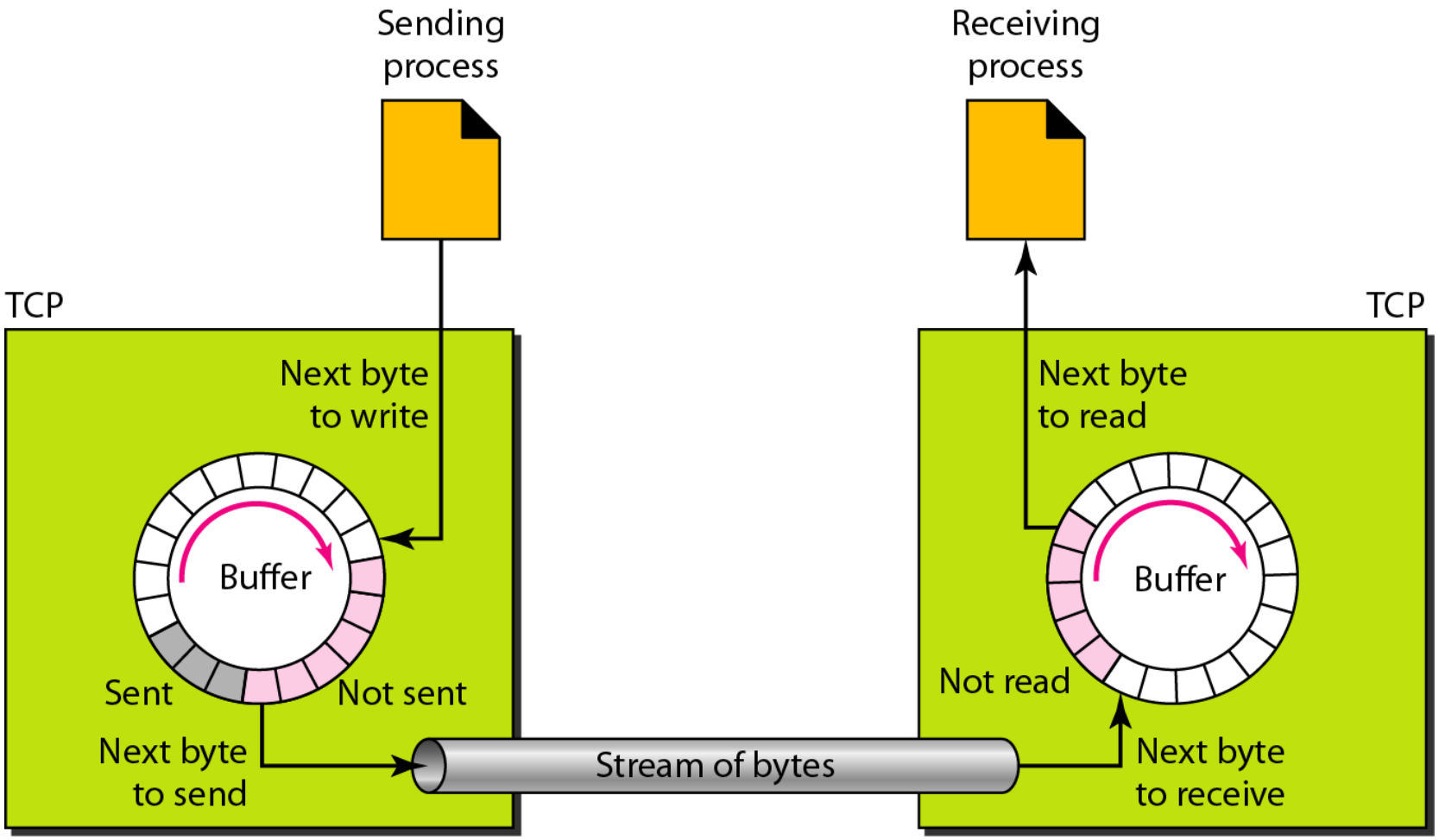
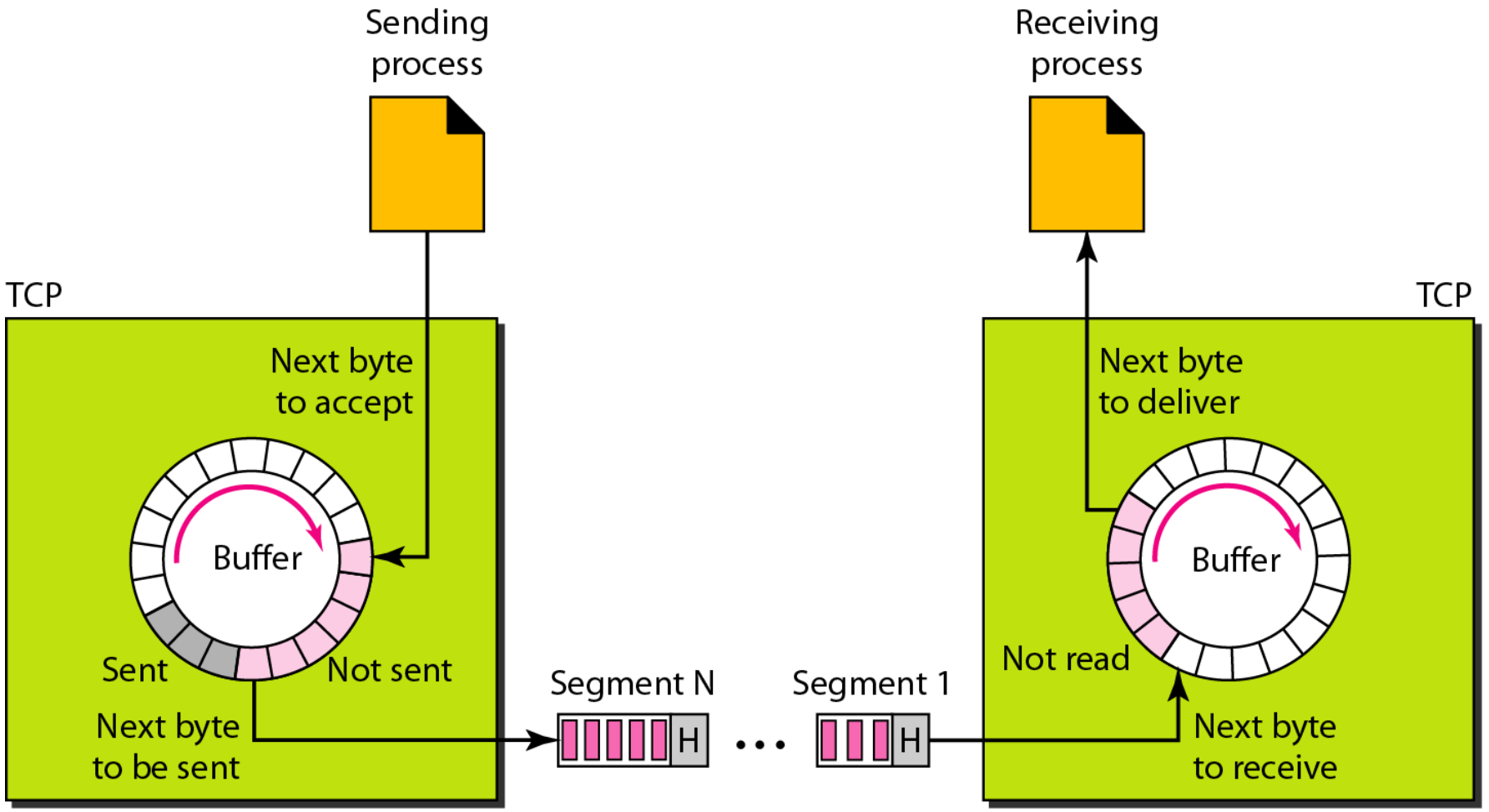


Figure *Sending and receiving buffers*



TCP segments



Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

Solution

The following shows the sequence number for each segment:

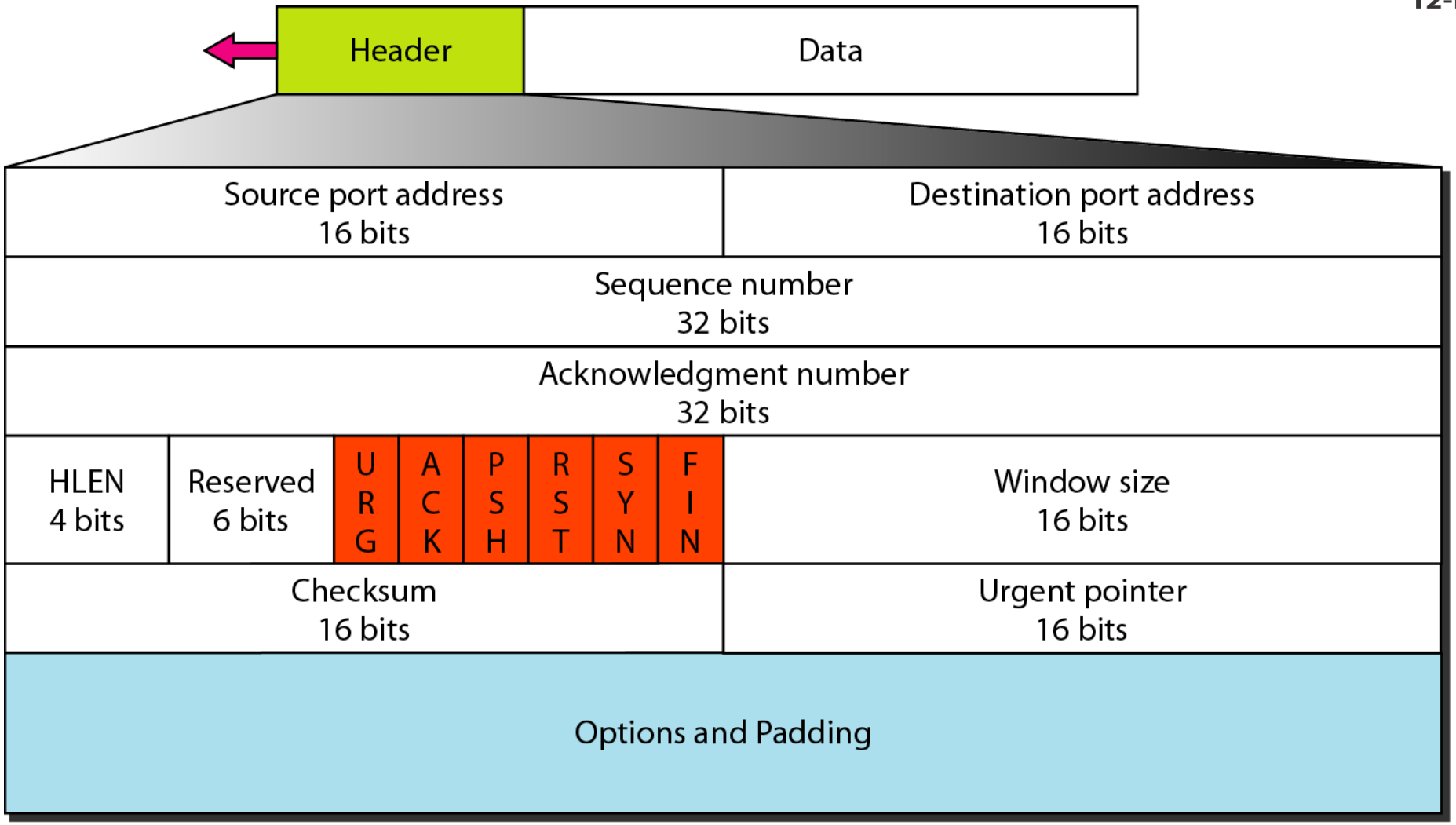
Segment 1	→	**Sequence Number:**	**10,001**	**Range:**	**10,001**	to	**11,000**
Segment 2	→	**Sequence Number:**	**11,001**	**Range:**	**11,001**	to	**12,000**
Segment 3	→	**Sequence Number:**	**12,001**	**Range:**	**12,001**	to	**13,000**
Segment 4	→	**Sequence Number:**	**13,001**	**Range:**	**13,001**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

SEGMENT

Before discussing TCP in more detail, let us discuss the TCP packets themselves. A packet in TCP is called a segment.

The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

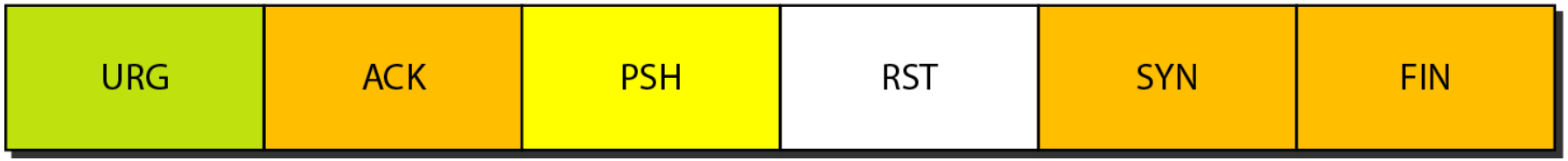
TCP segment format



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



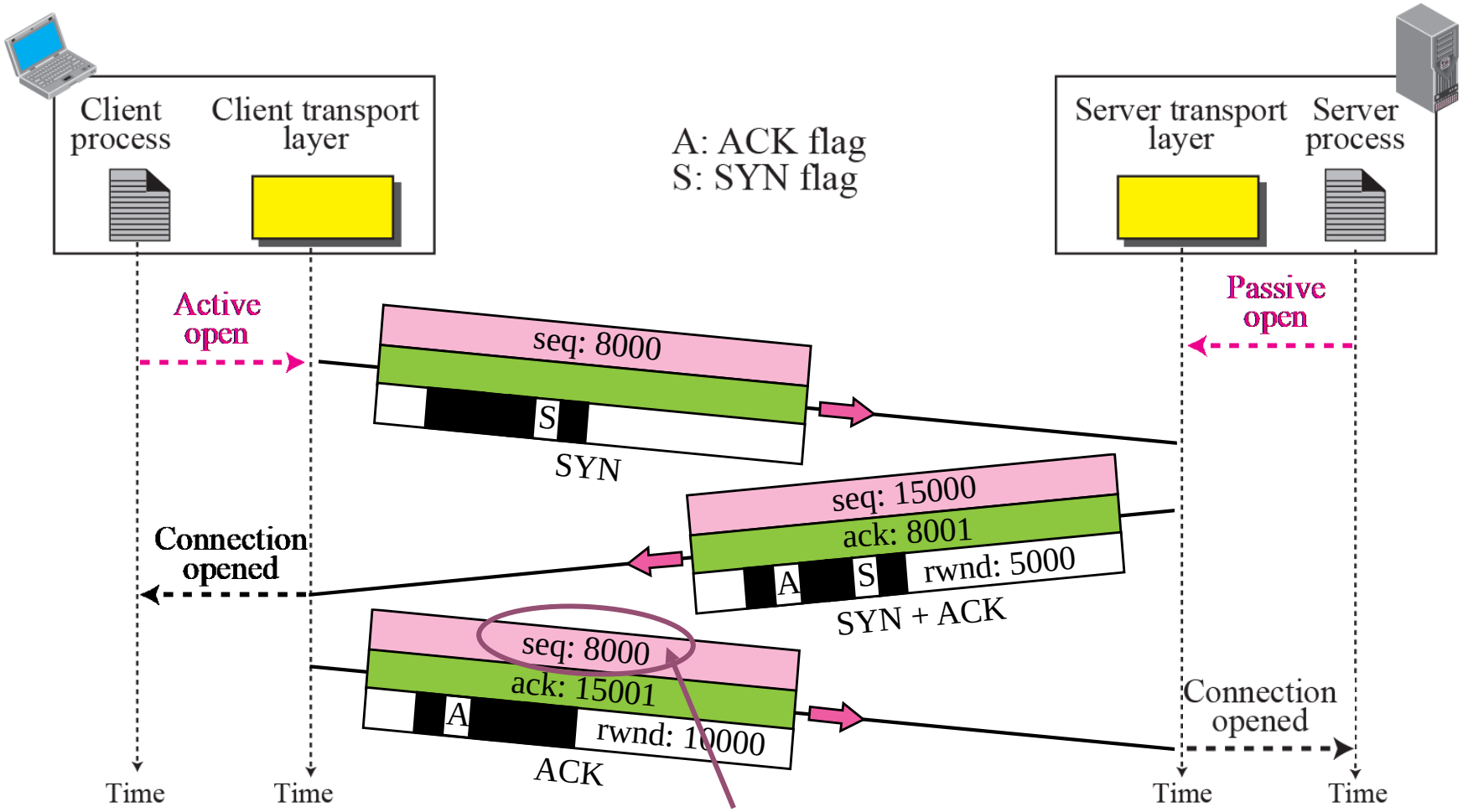
Description of flags in the control field

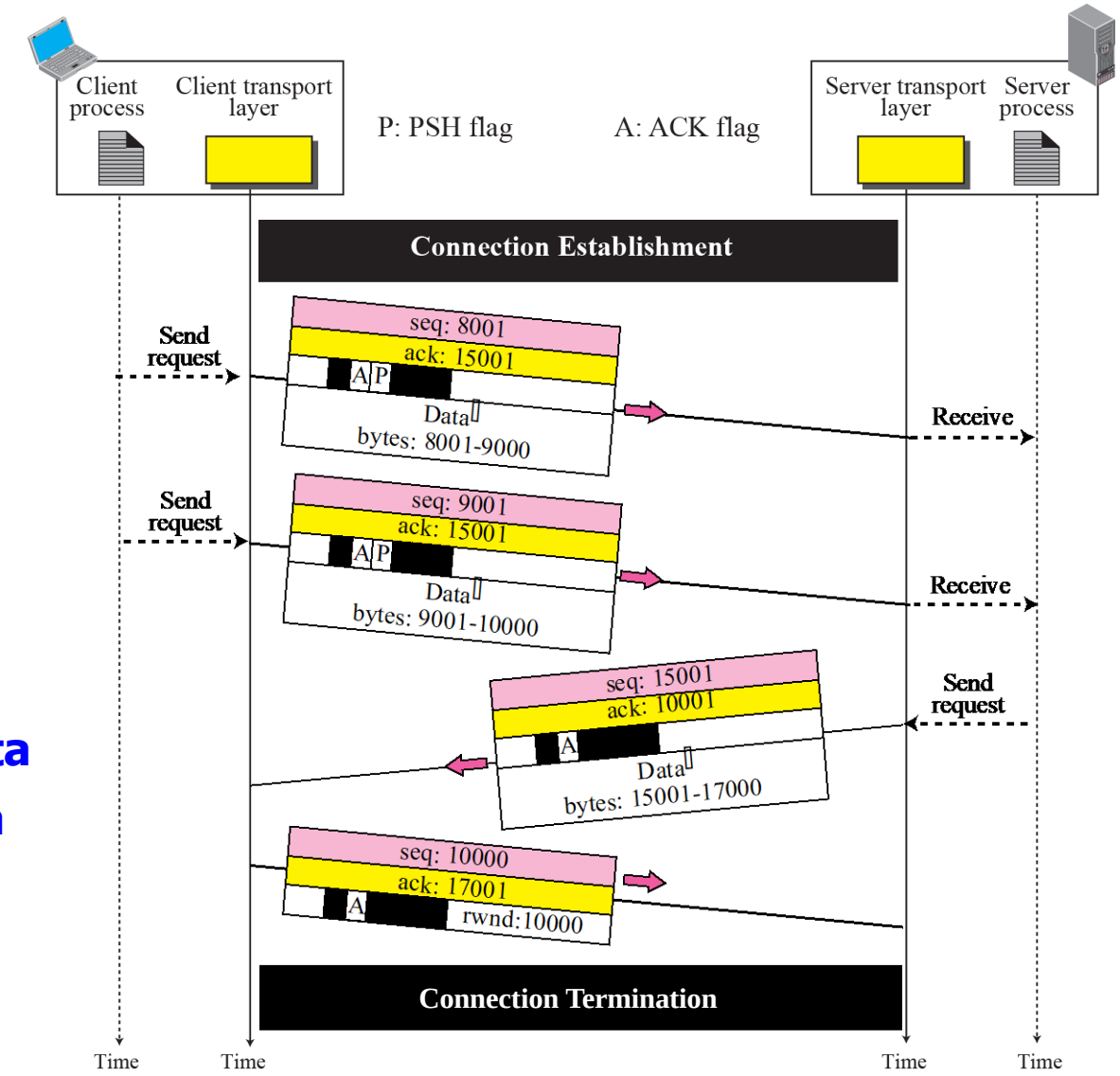
Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

TCP CONNECTION

TCP is connection-oriented. It establishes a virtual path between the source and destination. All of the segments belonging to a message are then sent over this virtual path. You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented. The point is that a 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.

Figure 15.9 *Connection establishment using three-way handshake*

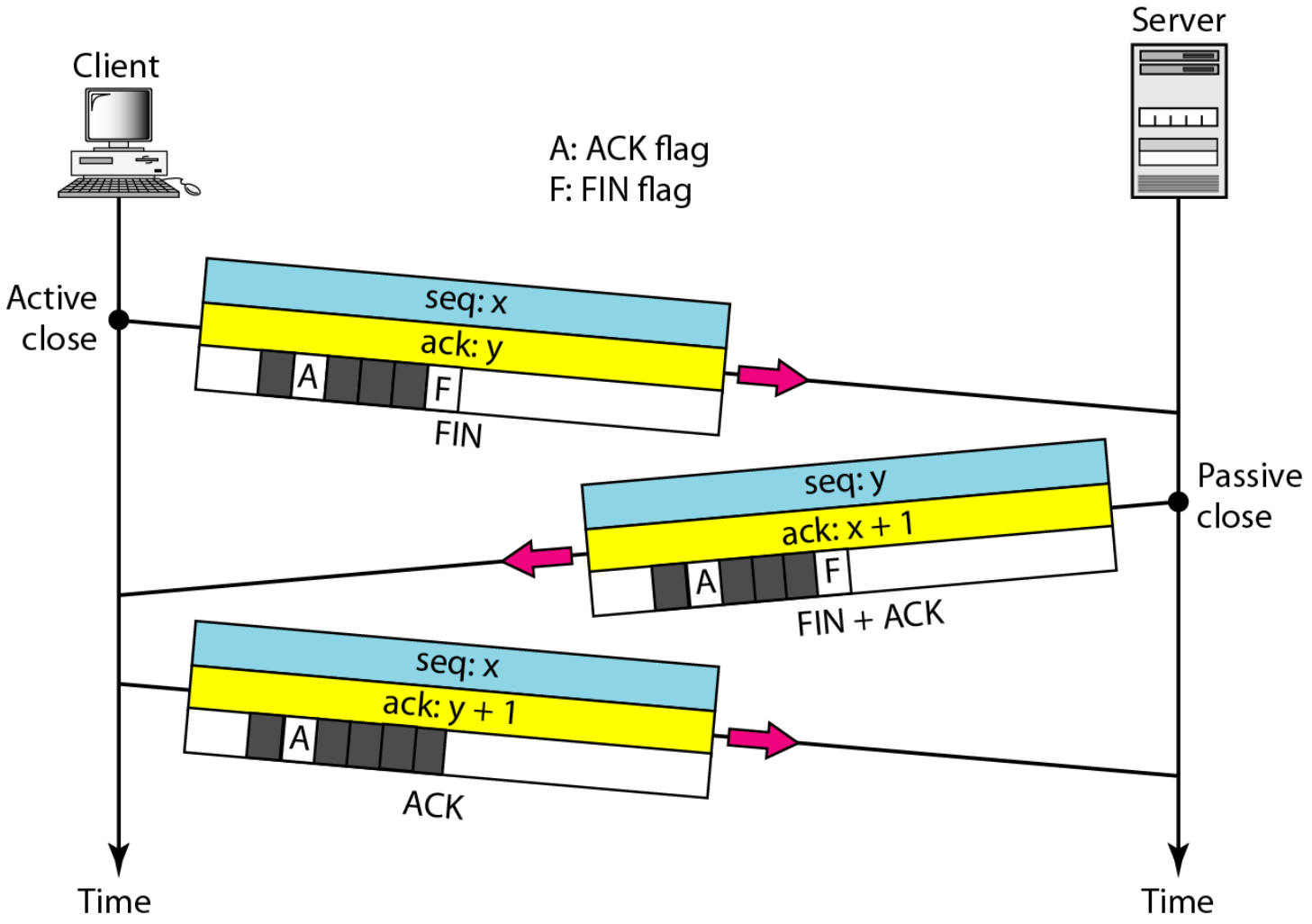




Pushing data

Urgent data

Connection termination using three-way handshaking



TCP client P successfully establishes a connection to TCP server Q. Let N_P denote the sequence number in the SYN sent from P to Q. Let N_Q denote the acknowledgement number in the SYN ACK from Q to P. Which of the following statements is/are CORRECT?

A. The sequence number N_P is chosen randomly by P

B. The sequence number N_P is always 0 for a new connection

C. The acknowledgement number N_Q is equal to N_P

D. The acknowledgement number N_Q is equal to $N_P + 1$

GATE CSE 2024 SET-1 Computer Network

Consider the three-way handshake mechanism followed during TCP connection establishment between hosts P and Q. Let X and Y be two random 32-bit starting sequence numbers chosen by P and Q respectively. Suppose P sends a TCP connection request message to Q with a TCP segment having SYN bit =1, SEQ number =X, and ACK bit =0. Suppose Q accepts the connection request. Which one of the following choices represents the information present in the TCP segment header that is sent by Q to P?

A SYN bit =1, SEQ number =X+1, ACK bit =0, ACK number =Y, FIN bit =0

B SYN bit =0, SEQ number =X+1, ACK bit =0, ACK number =Y, FIN bit =1

C SYN bit =1, SEQ number =Y, ACK bit =1, ACK number =X+1, FIN bit =0

D SYN bit =1, SEQ number =Y, ACK bit =1, ACK number =X, FIN bit =0

GATE CSE 2021 SET-2 Computer Network

TCP header itself is of 10 fields as below and size may vary between 20 to 60bytes

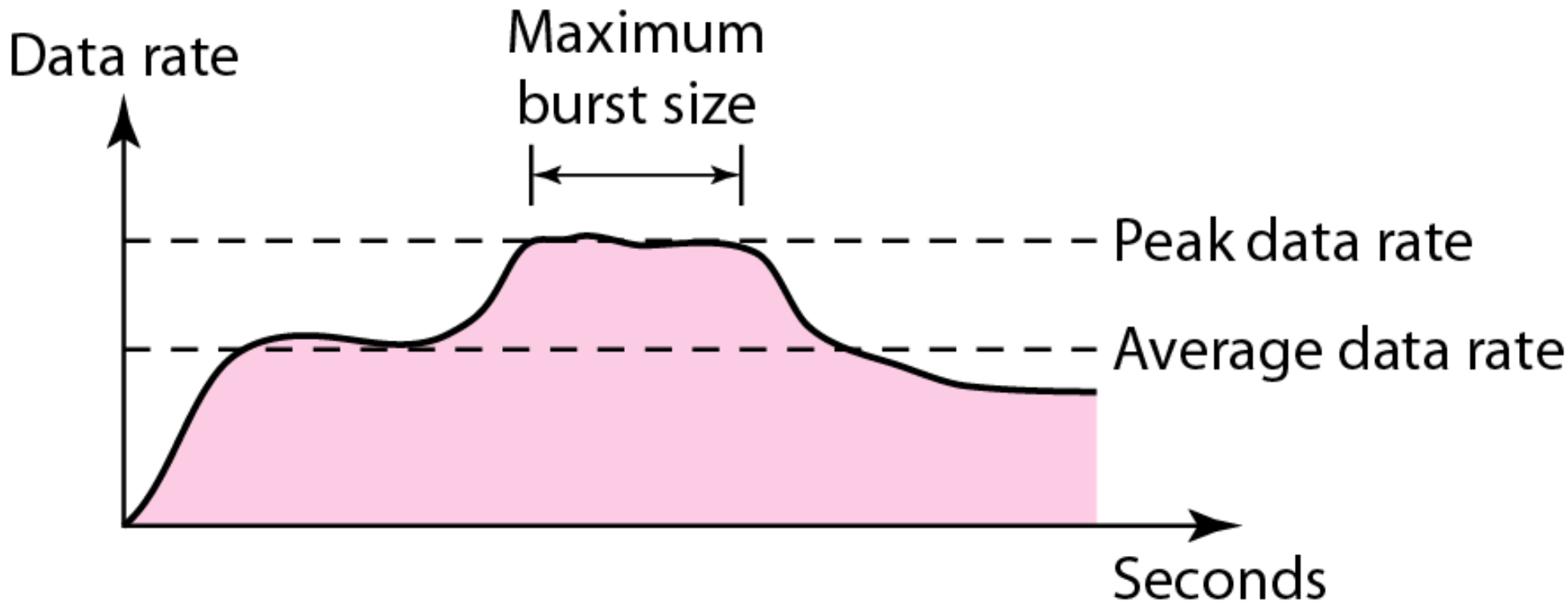
1.Source port - 2 bytes
2.destination port - 2 bytes
3.SEQ NUM-4 bytes
4.ACK NUM- 4 bytes
5.HLEN-1 word
6.RESERVED-6bits
7.CONTROL-6bits
8.WINDOW SIZE-2 bytes
9.CHECKSUM-2 bytes
10.URGENT POINTERS-2bytes

TCP header(in hex)=05320017 00000001 000000000 500207FF 00000000

since each hex = 4 bits , we need to first split the above hex as such
05 32 00 17 00 00 00 01 00 00 00 00 50 02 07 FF 00 00 00 00

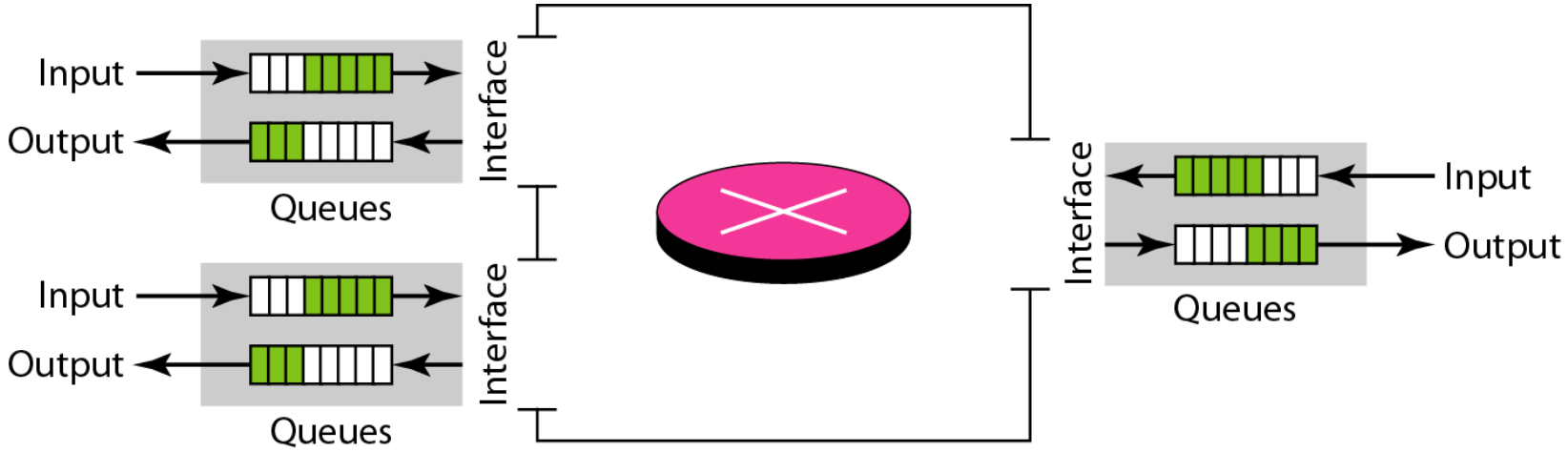
source port is 2 bytes take 05 32 = 1330
next 2 bytes as destination address 00 17 == 23 (default TCP port)
next 4 bytes as sequence number 00 00 00 01 ==1
next 4 bytes as ack 00 00 00 00 == 0
next 4 bits as HLEN 5 ==5 -- this indicates number of sets of 4 bytes which makes the header lenght = 20bytes..
next 6 bits are reserved i.e.0 =0000and 2 bits from hex 0
next 6 bits are control bits = remaining 2 bits from hex 0 and 4 bits of 2
next 2 bytes indicate the window length 07 FF == 2047 bytes

Checksum 2 bytes 00 00 = 0
Urgent pointer 2bytes 00 00 =0



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.

Queues in a router



1. **The packet is put at the end of the input queue while waiting to be checked.**

2. **The processing module of the router removes the packet from the input queue once it reaches the front of the queue and uses its routing table and the destination address to find the route.**

3. **The packet is put in the appropriate output queue and waits its time to be sent.**

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).

Open-Loop Congestion Control
Closed-Loop Congestion Control

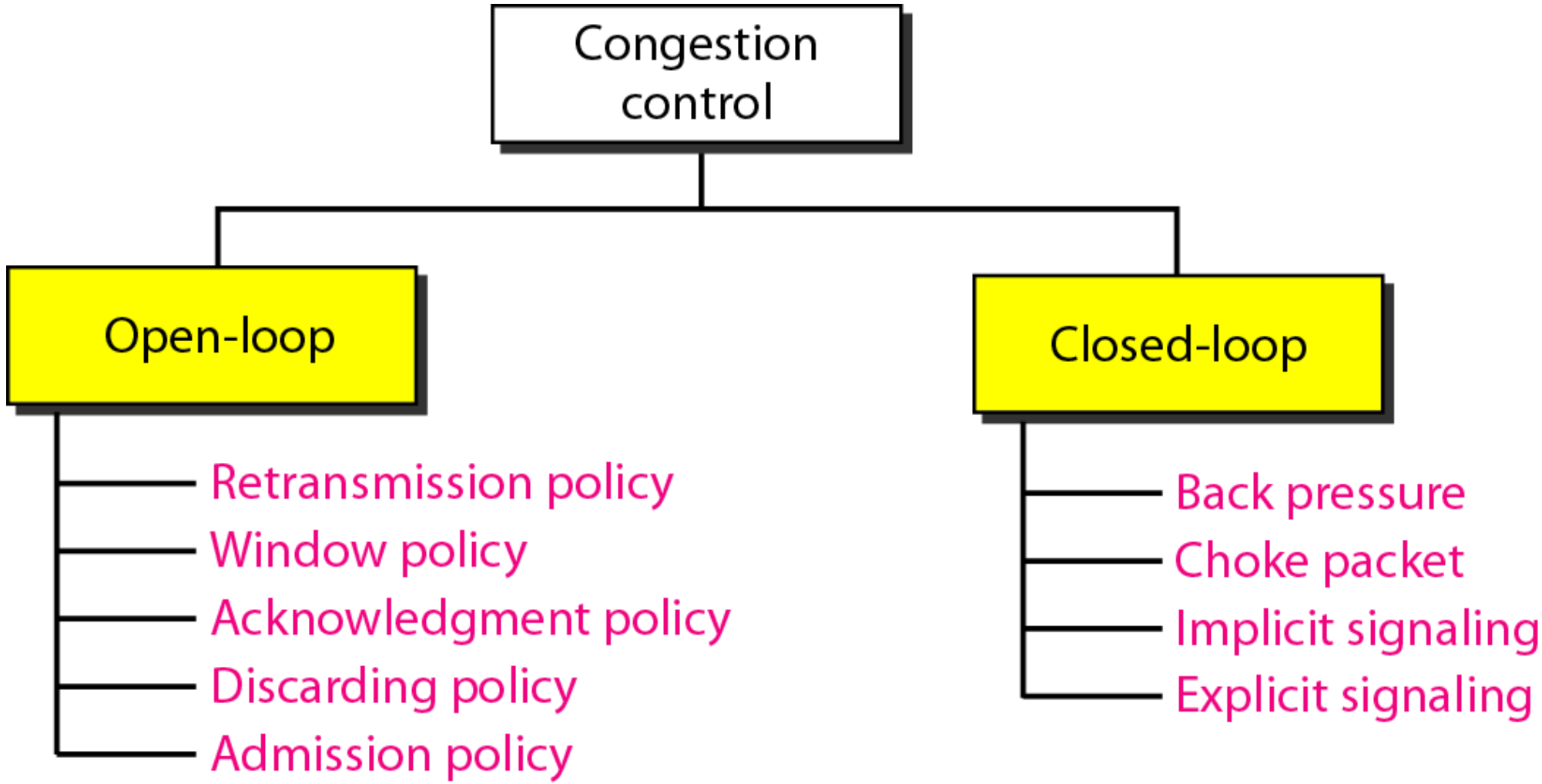
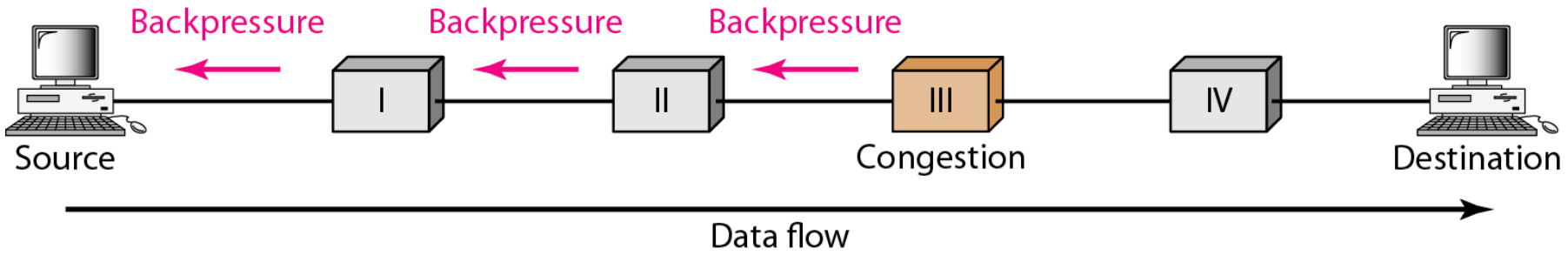
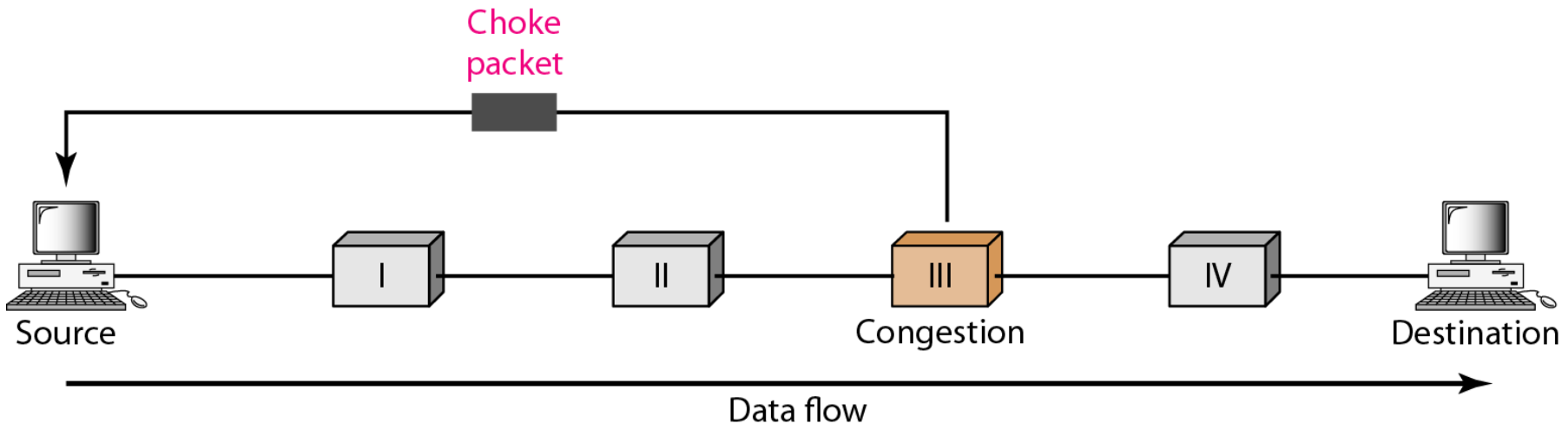


Figure 24.6 *Backpressure method for alleviating congestion*



The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is corning

Figure 24.7 *Choke packet*



example of this type of control in ICMP.

Implicit Signaling

In implicit signaling, there is no communication between the congested node or nodes and the source.
For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested

Explicit Signaling

The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signaling method, however, is different from the choke packet

method. in the explicit signaling method, the signal is included in the packets that carry data

Backward Signaling

Forward Signaling

congestion control in TCP

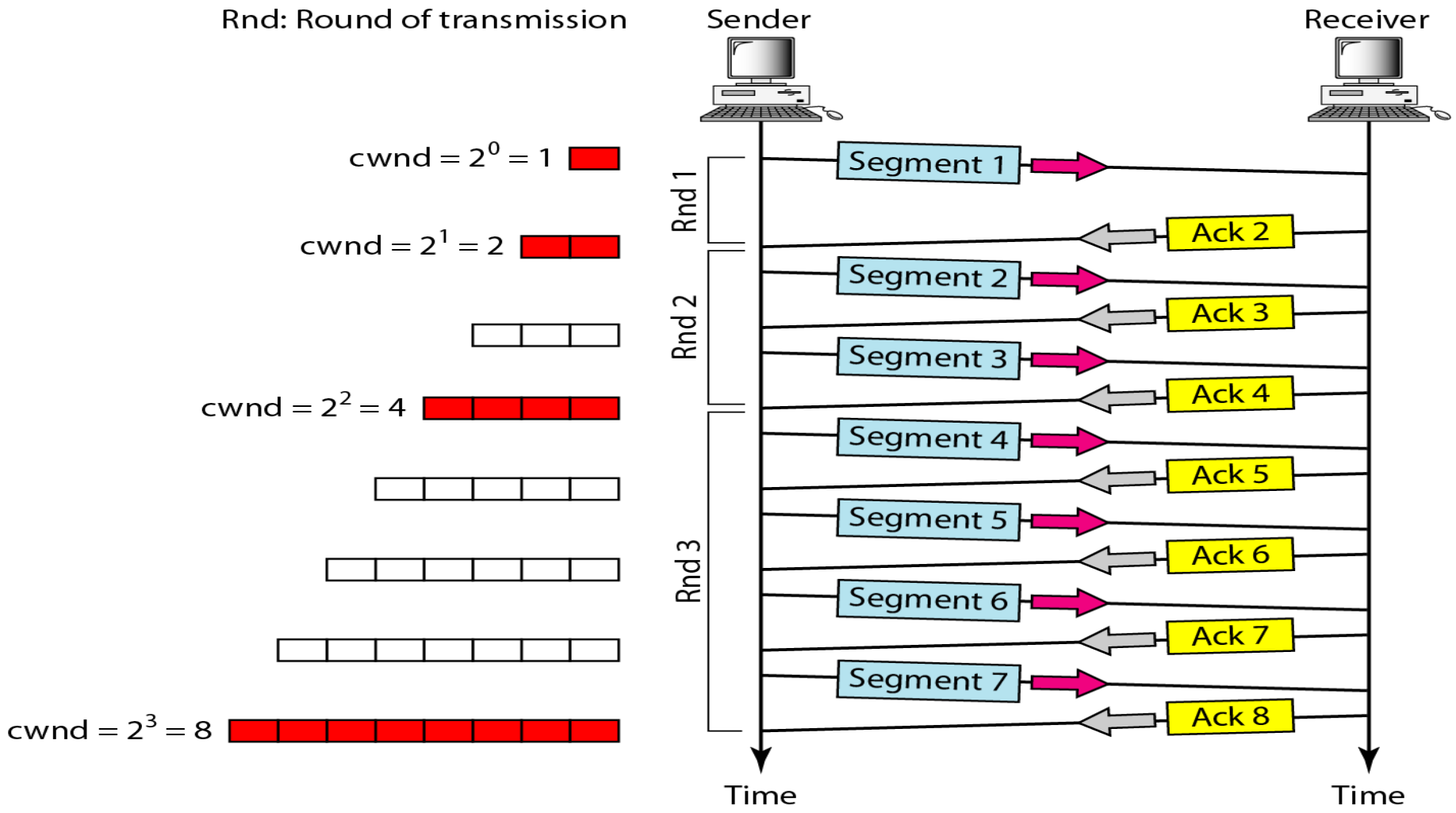
Congestion Policy in TCP

1.Slow Start Phase: Starts slow increment is exponential to the threshold.

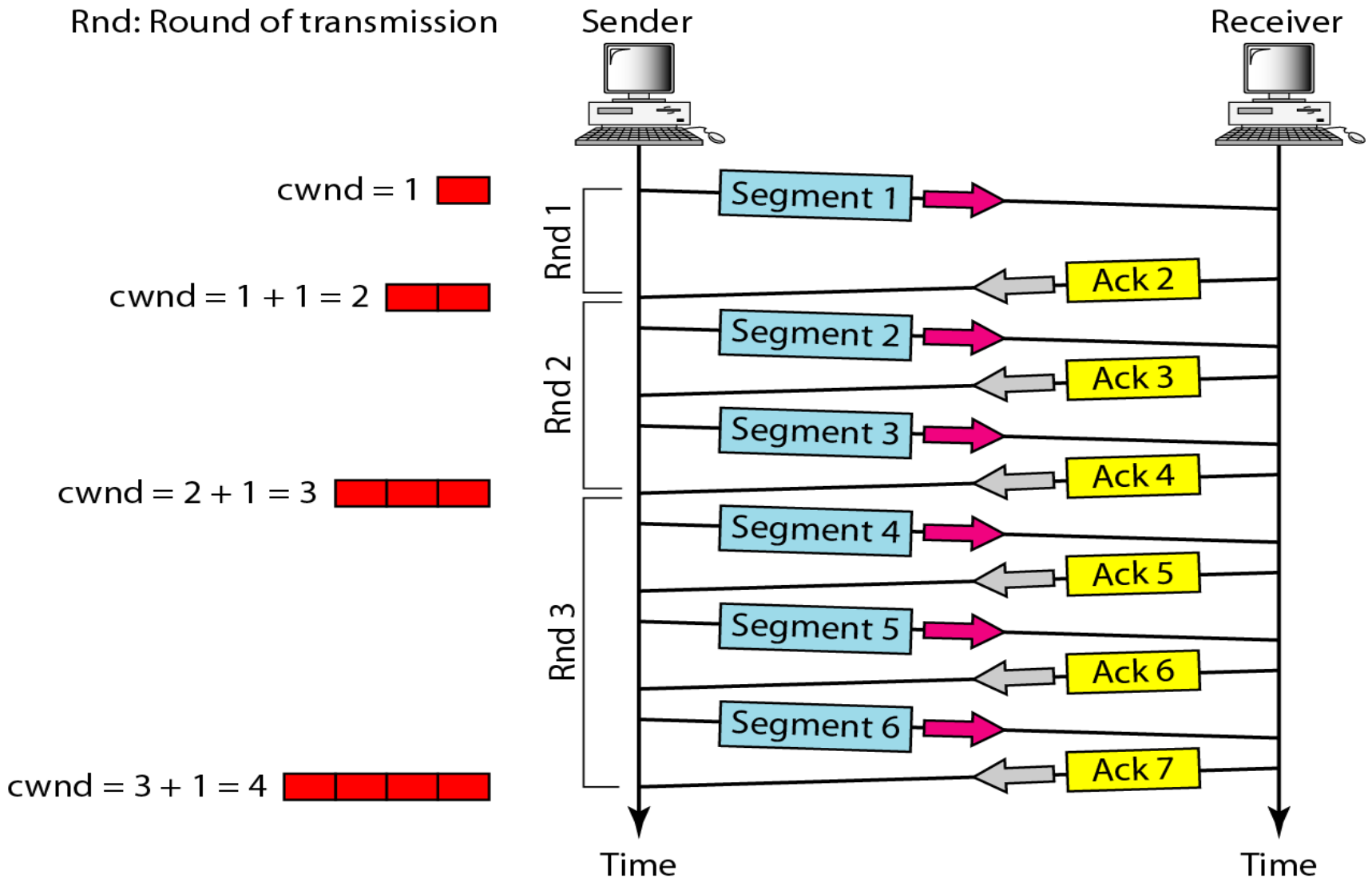
2.Congestion Avoidance Phase: After reaching the threshold increment is by 1.

3.Congestion Detection Phase: The sender goes back to the Slow start phase or the Congestion avoidance phase.

Slow start, exponential increase



Congestion avoidance, additive increase



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

If detection is by time-out.

If detection is by three ACKs

If a time-out occurs, there is a stronger possibility of congestion; a segment has probably been dropped in the network, and there is no news about the sent segments. In this case TCP reacts strongly:

a. It sets the value of the threshold to one-half of the current window size.

b. It sets cwnd to the size of one segment.

c. It starts the slow-start phase again.

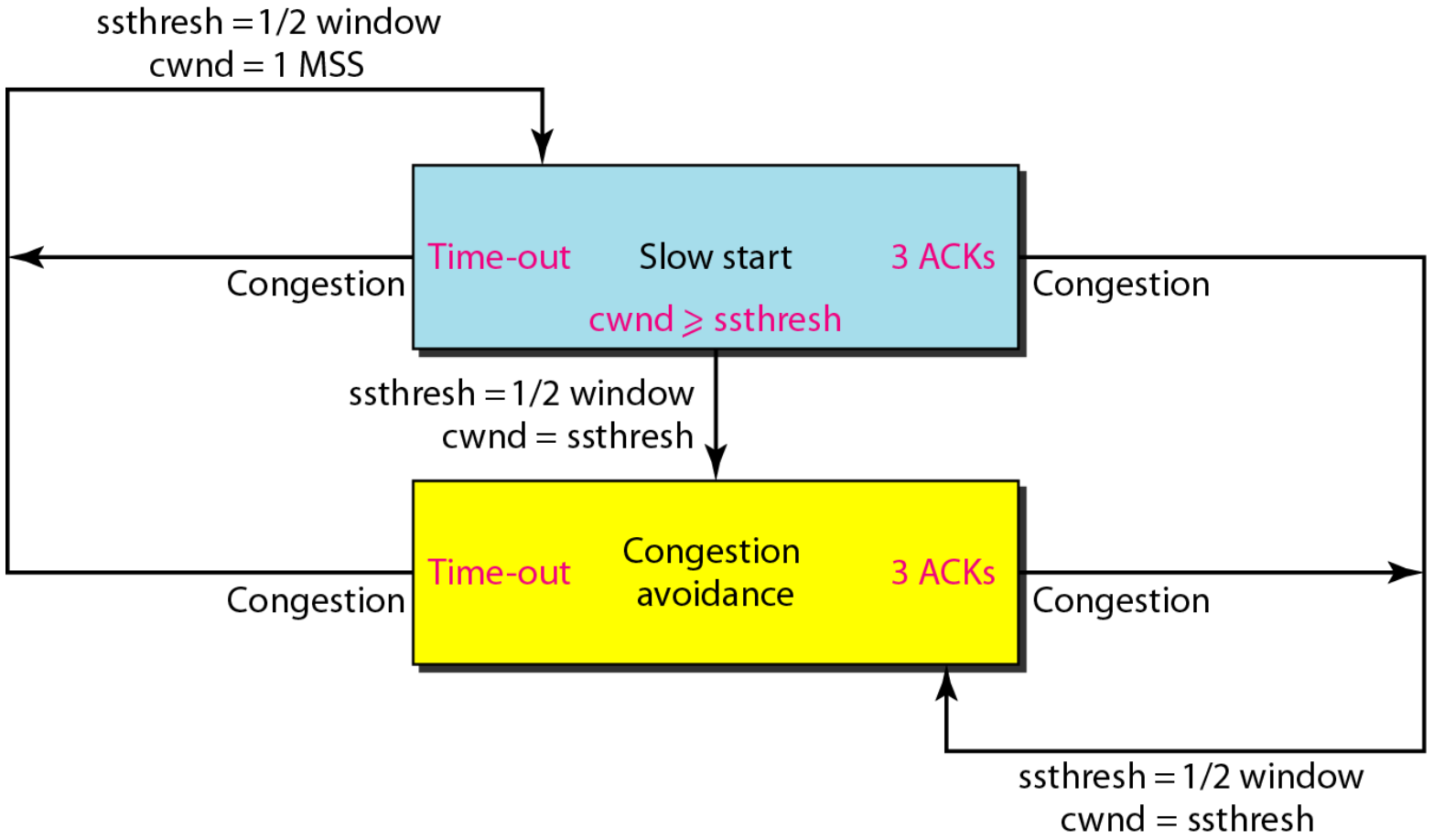
If three ACKs are received, there is a weaker possibility of congestion; a segment may have been dropped, but some segments after that may have arrived safely since three ACKs are received. This is called fast transmission and fast recovery. In this case, TCP has a weaker reaction:

a. It sets the value of the threshold to one-half of the current window size.

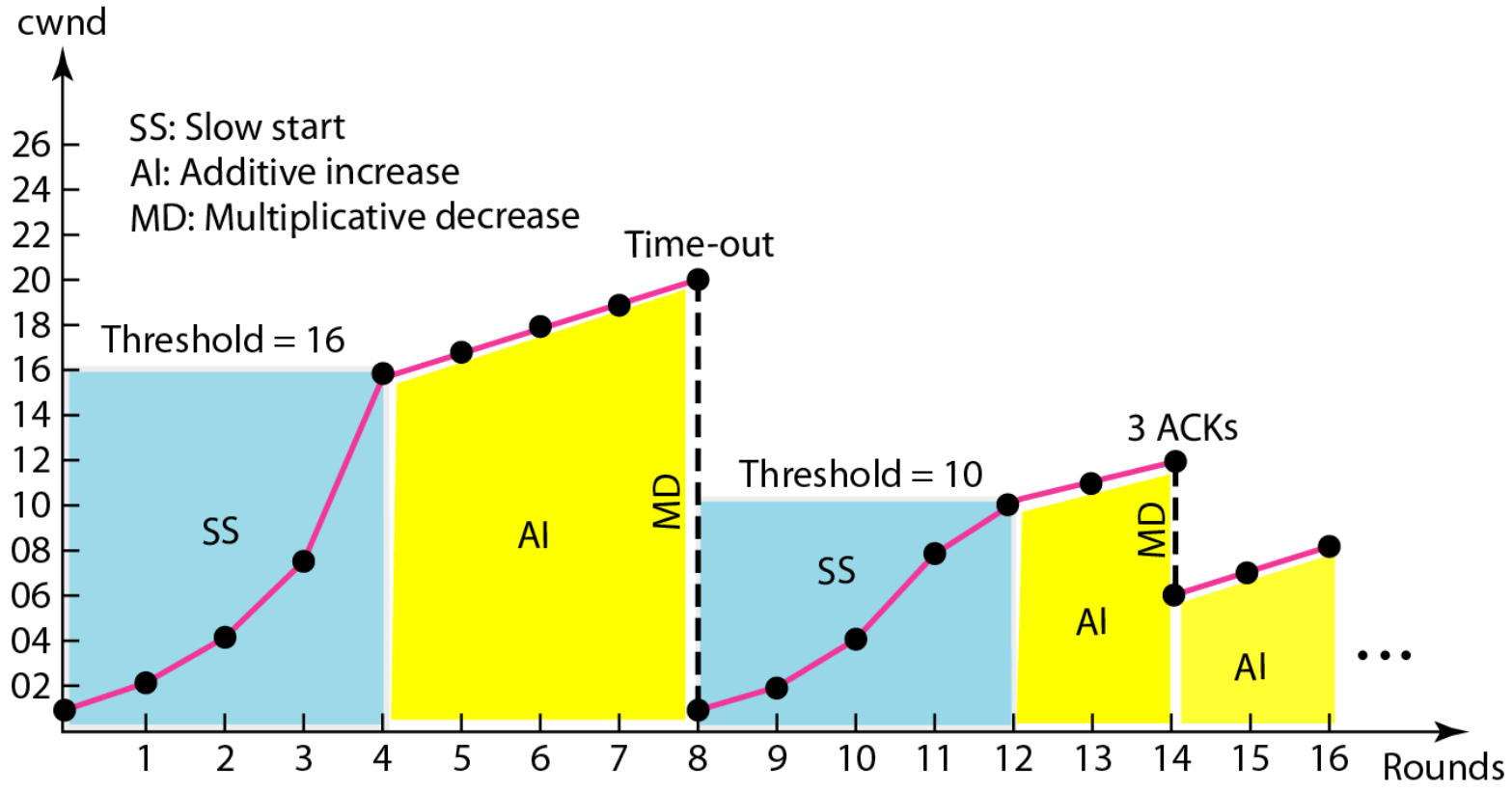
b. It sets cwnd to the value of the threshold (some implementations add three segment sizes to the threshold).

c. It starts the congestion avoidance phase.

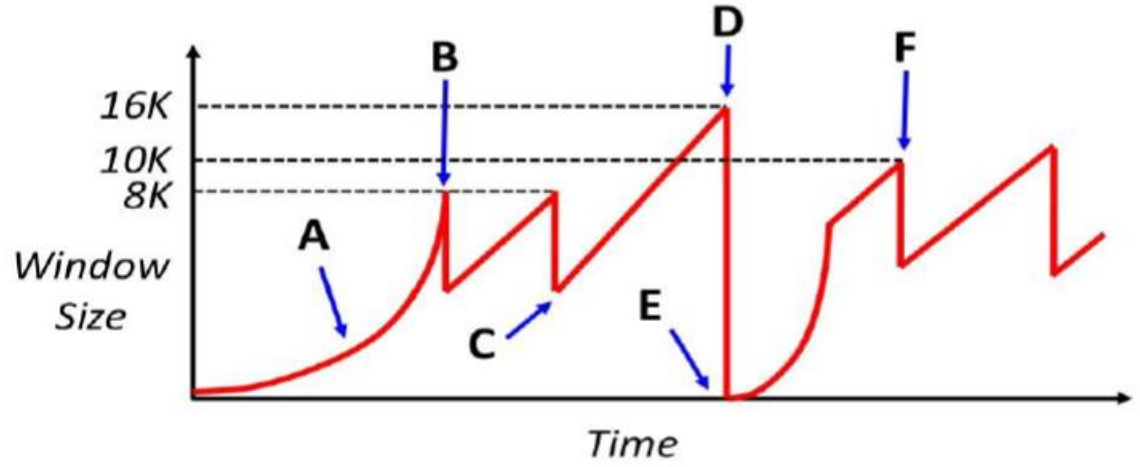
TCP congestion policy summary



Congestion example



Name the event at B which occurs that causes the sender to decrease its window



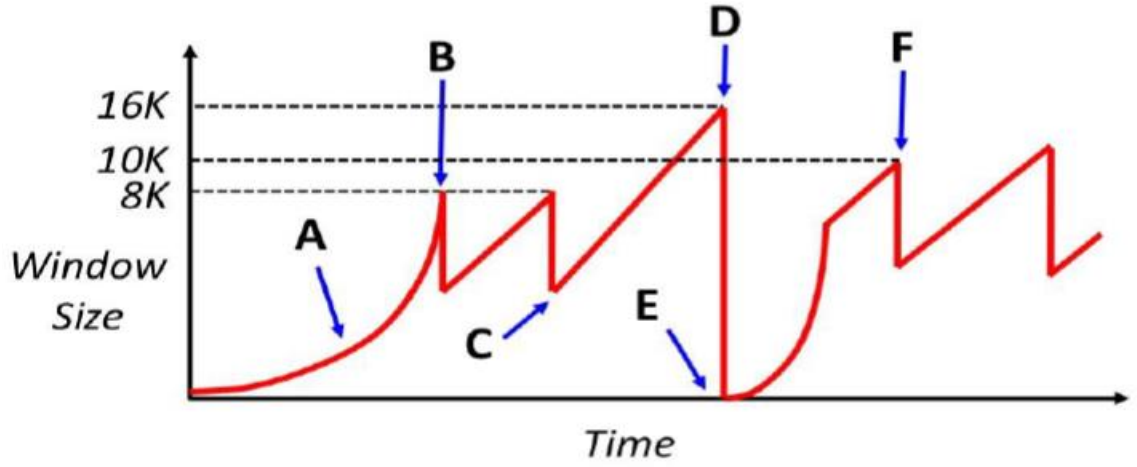
(a) Triple Duplicate Ack

(b) Slow Start

(c) Packet loss

(d) Time out

Name the event at D which occurs that causes the sender to decrease its window.



(a) Triple Duplicate Ack

(b) Slow Start

(c) Packet loss

(d) Time out

Consider an instance of TCP's Additive Increase Multiplicative Decrease(AIMD) algorithm where the window size at the start of the slow start phase is 2 MSS and the threshold at the start of the first transmission is 8 MSS. Assume that a time out occurs during the fifth transmission. Find the congestion window size at the end of the tenth transmission. (GATE 2012)

Window size for 1st transmission = 2 MSS
Window size for 2nd transmission = 4 MSS
Window size for 3rd transmission = 8 MSS
threshold reached, increase linearly (according to AIMD)
Window size for 4th transmission = 9 MSS
Window size for 5th transmission = 10 MSS
time out occurs, resend 5th with window size starts with as slow start.
Window size for 6th transmission = 2 MSS
Window size for 7th transmission = 4 MSS
threshold reached, now increase linearly (according to AIMD)
Additive Increase: 5 MSS (since 8 MSS isn't permissible anymore)
Window size for 8th transmission = 5 MSS
Window size for 9th transmission = 6 MSS
Window size for 10th transmission = 7 MSS

Suppose that the TCP congestion window is set to 18 KB and a time out occurs. If Ssthreshold=9 then how big will the window be if the next four transmission bursts are all successful? Assume that the MSS is 1 KB.

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

A TCP connection is established between two processes P1 and P2 running on different hosts. Segment size is set to 800 bytes for this connection. Initially, the ssthresh at P1 is 7000 bytes. Then, P1 receives a TCP ACK reporting a rwnd size of 4000 bytes. After reception and processing of this ACK, the cwnd value is 7200 bytes. How many maximum-sized segments can P1 transmit now (i.e., after the ACK processing)?

Given segment size = 800 bytes New rwnd size = 4000 bytes(through ACK reporting)& cwnd size=7200 bytes
Since window size= min(cwnd ,rwnd) => min(7200,4000)= 4000 bytes
Therefore, number of maximum-size segment accommodated =4000/800= 5

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*

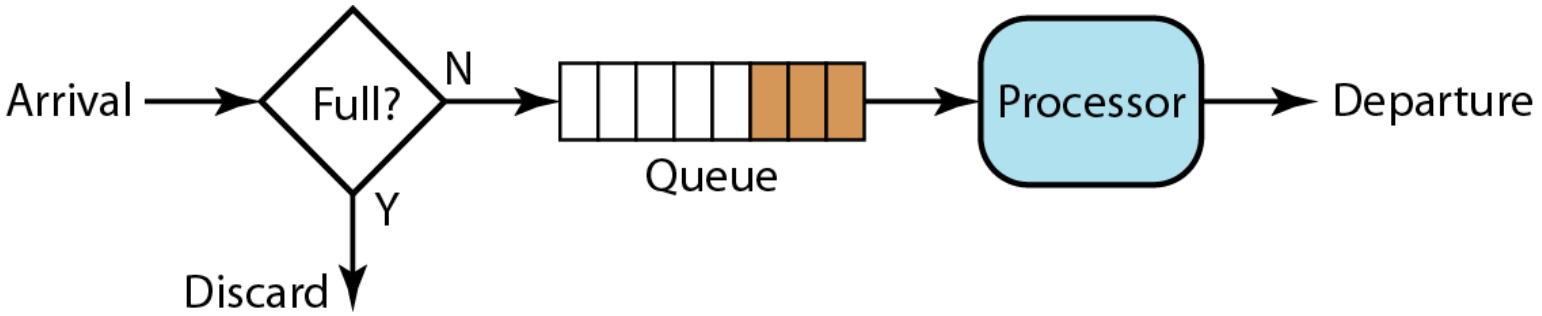


Figure 24.17 *Priority queuing*

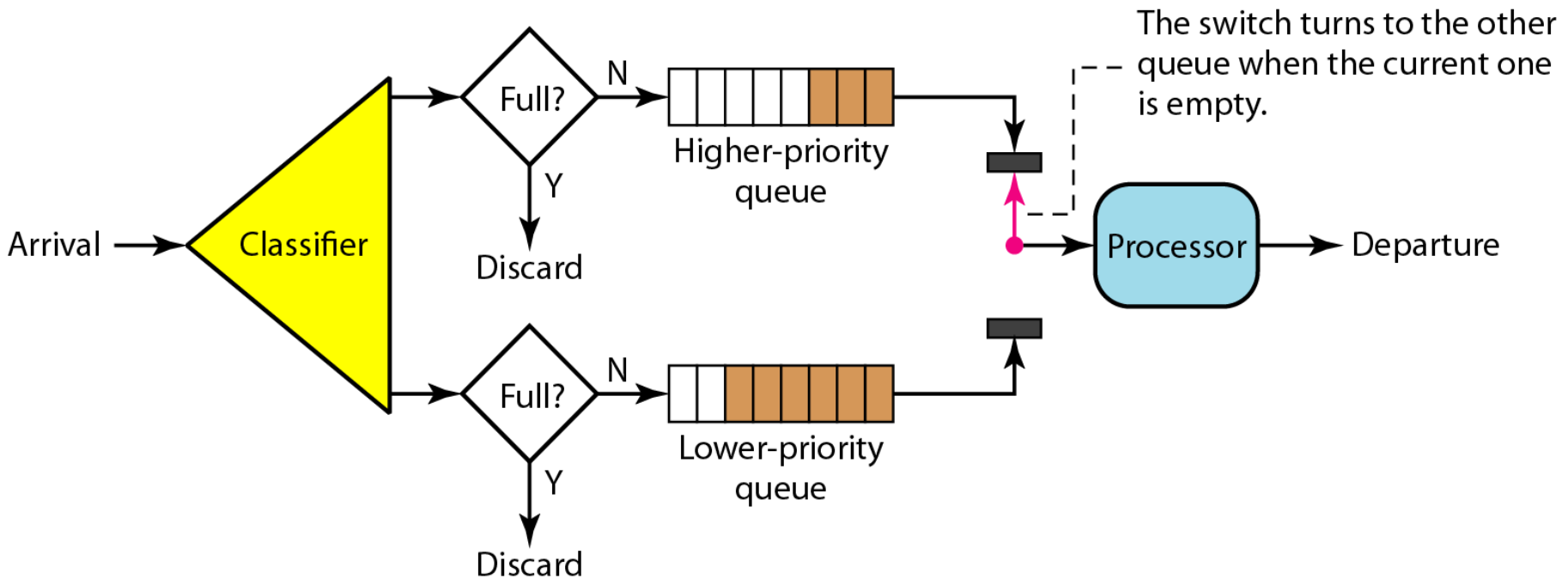
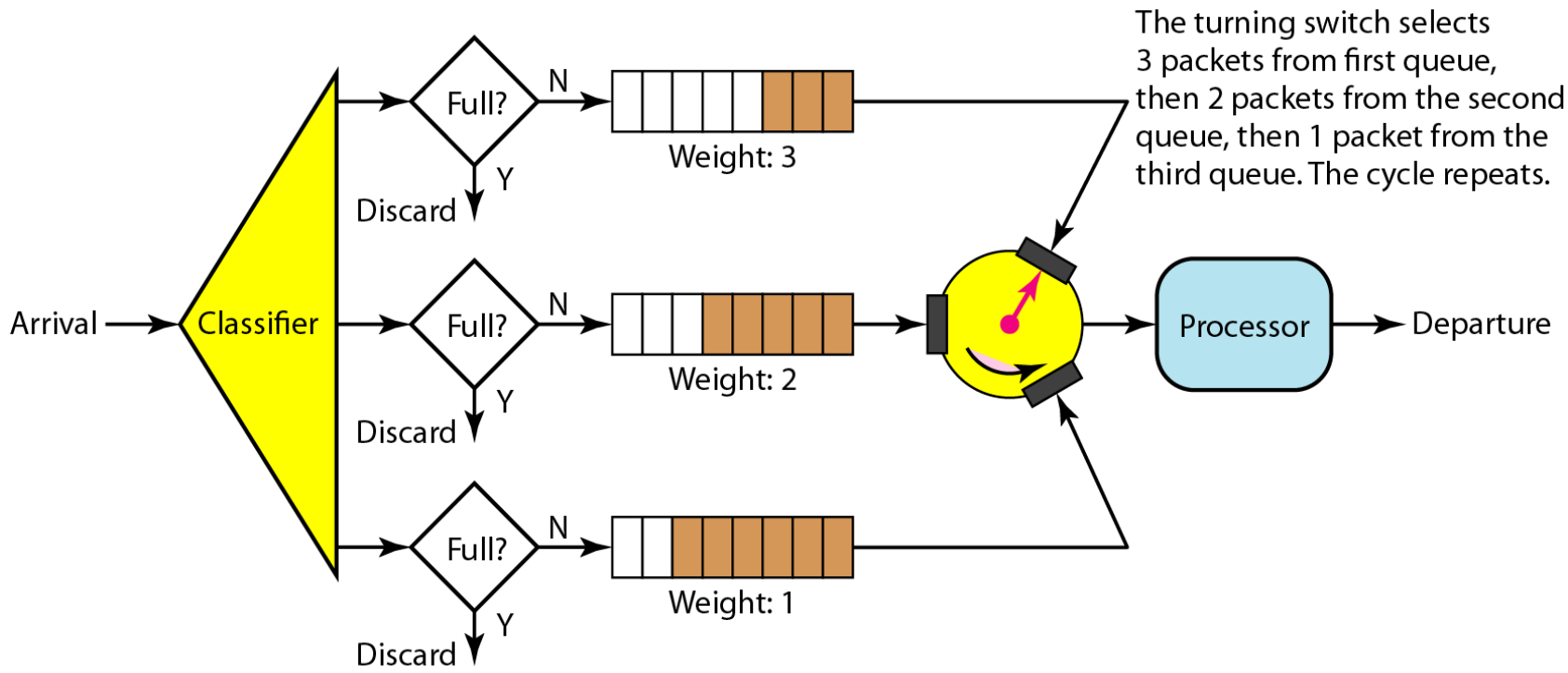


Figure 24.18 *Weighted fair queuing*



Traffic Shaping

Figure 24.19 *Leaky bucket*

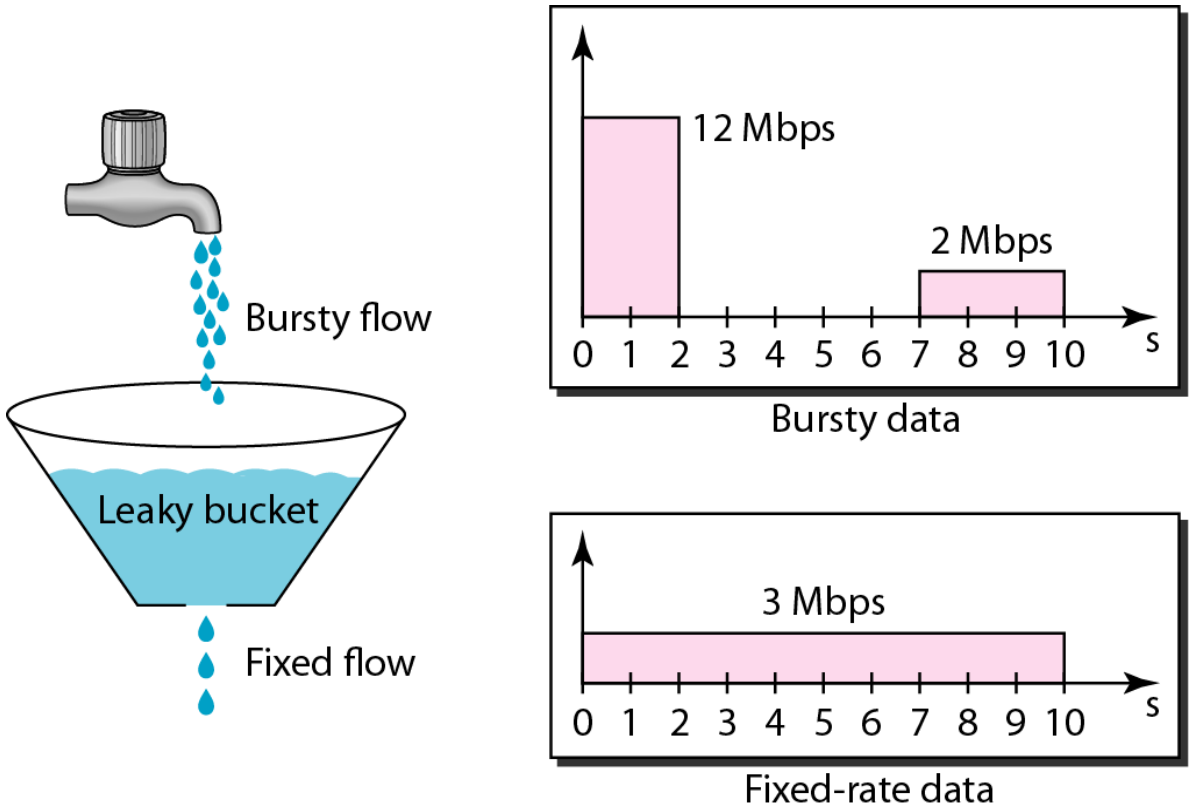
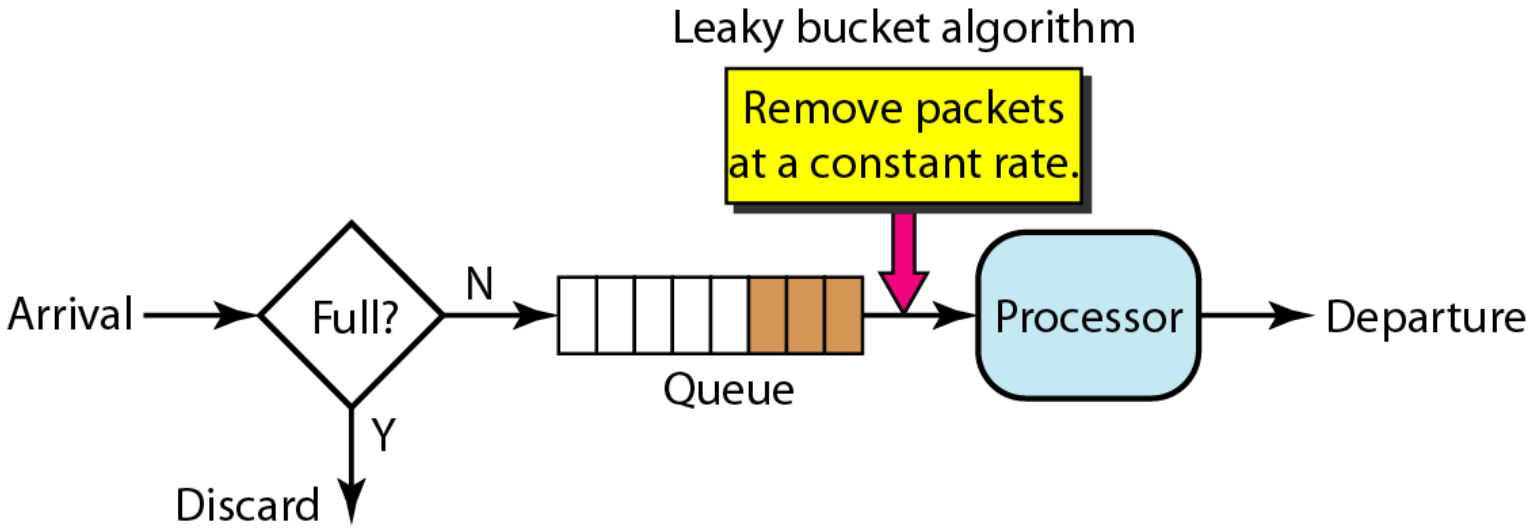


Figure 24.20 *Leaky bucket implementation*



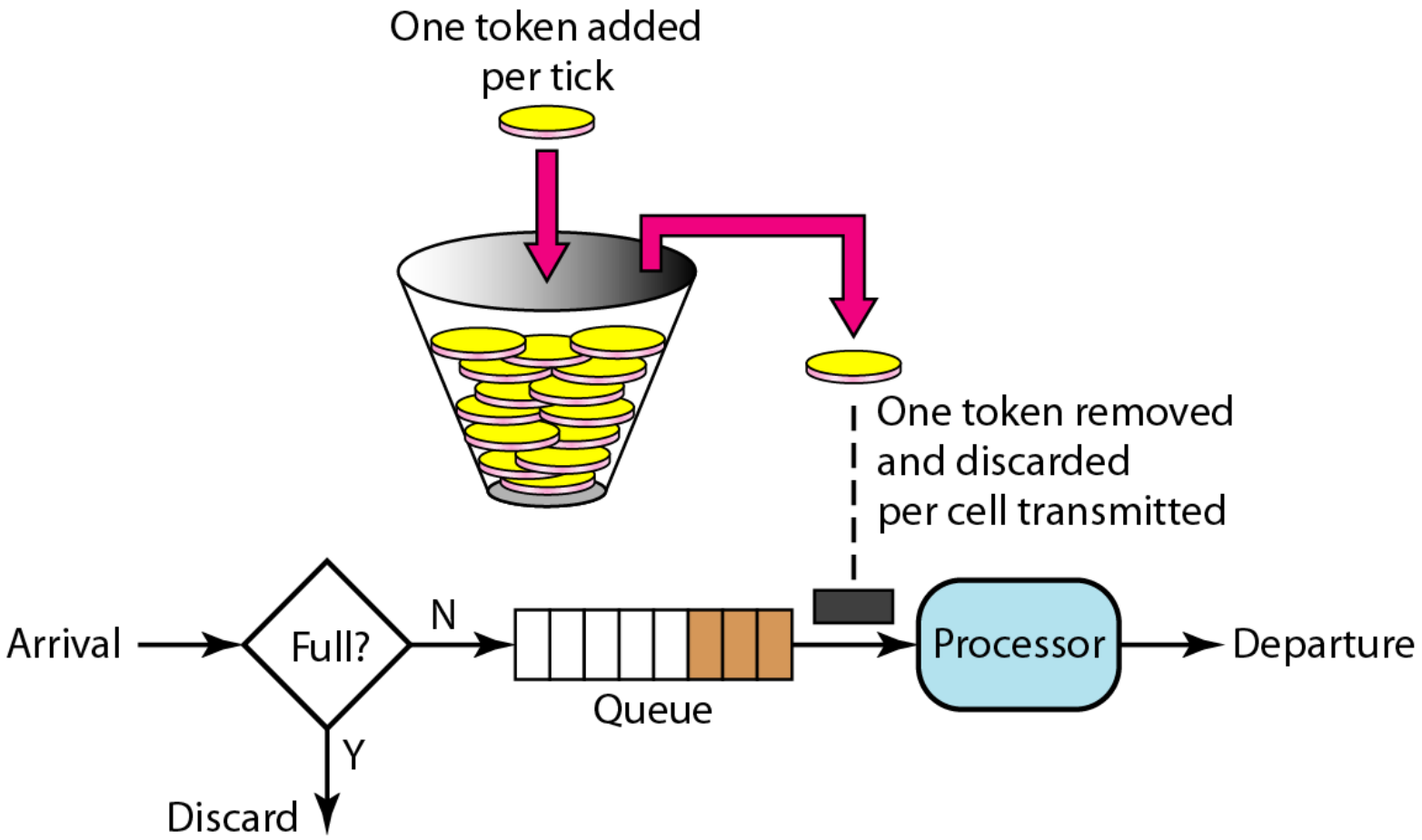
Note

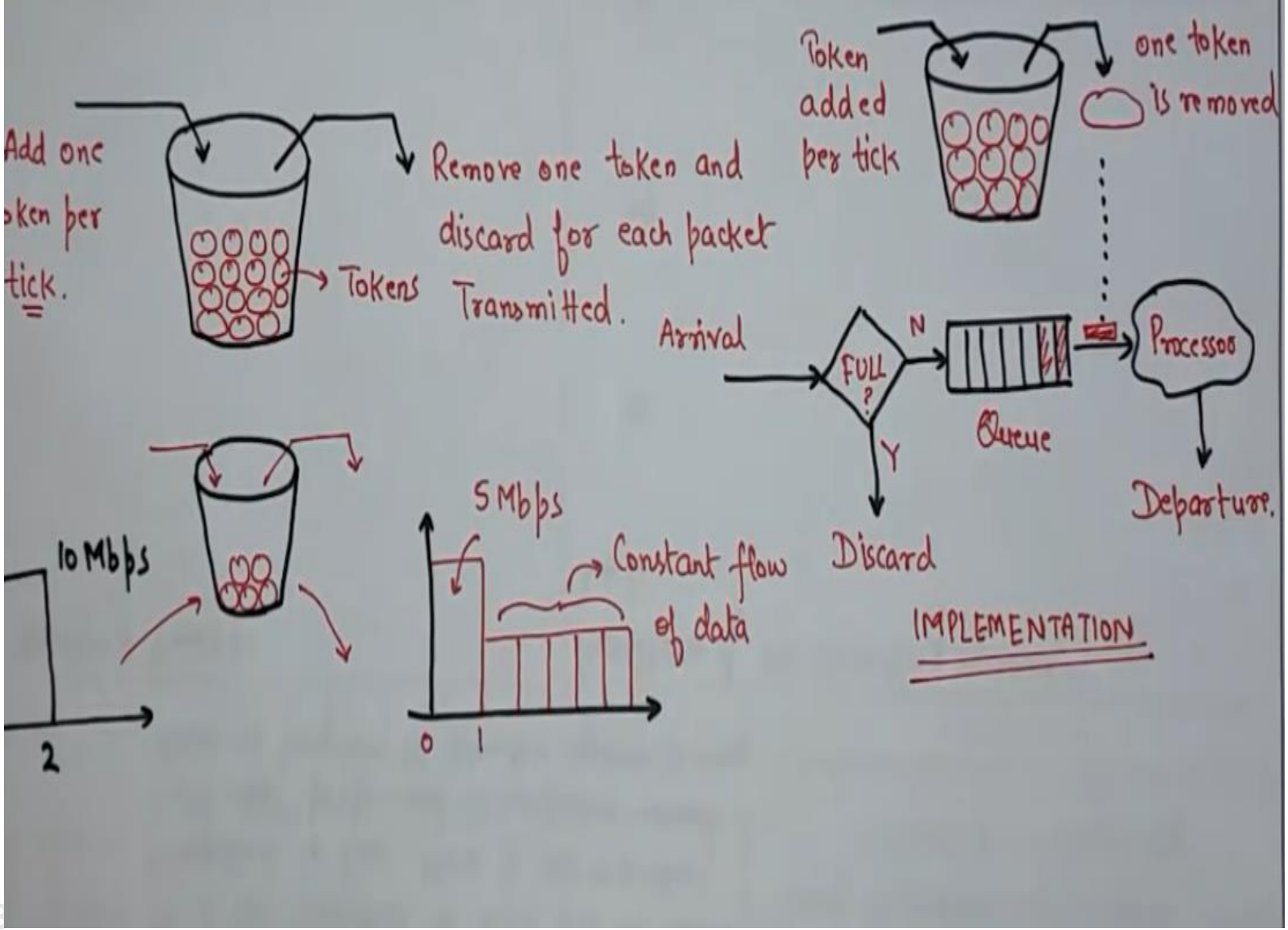
A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

Note

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

Figure 24.21 *Token bucket*




Add one token per tick.
Tokens
Remove one token and discard for each packet Transmitted.
Token added per tick
one token is removed
Arrival
FULL ?
N
Y
Queue
Processor
Departure.
Discard
10 Mbps
2
5 Mbps
Constant flow of data
0
1
IMPLEMENTATION

TOKEN BUCKET

i) Token Dependent.

ii) If Bucket is full token is discarded but not the packet.

iii) Packets can only transmit when there are enough tokens.

iv) Allows Large bursts to be send at faster rate.

v) Saves tokens to send Large bursts.

LEAKEY BUCKET

i) Token Independent.

ii) If Bucket is full, then packets are discarded.

iii) Packets are transmitted continuously.

iv) Sends the Packet at a Constant Rate.

v) No Concept of Token.

In a connection, the value of cwnd is 3000 and the value of rwnd is 5000. The host has sent 2000 bytes which has not been acknowledged. How many more bytes can be sent?