COMPUTER NETWORK

UNIT 5: TRANSPORT LAYER

PROF. ISHANI SAHA

TOPICS

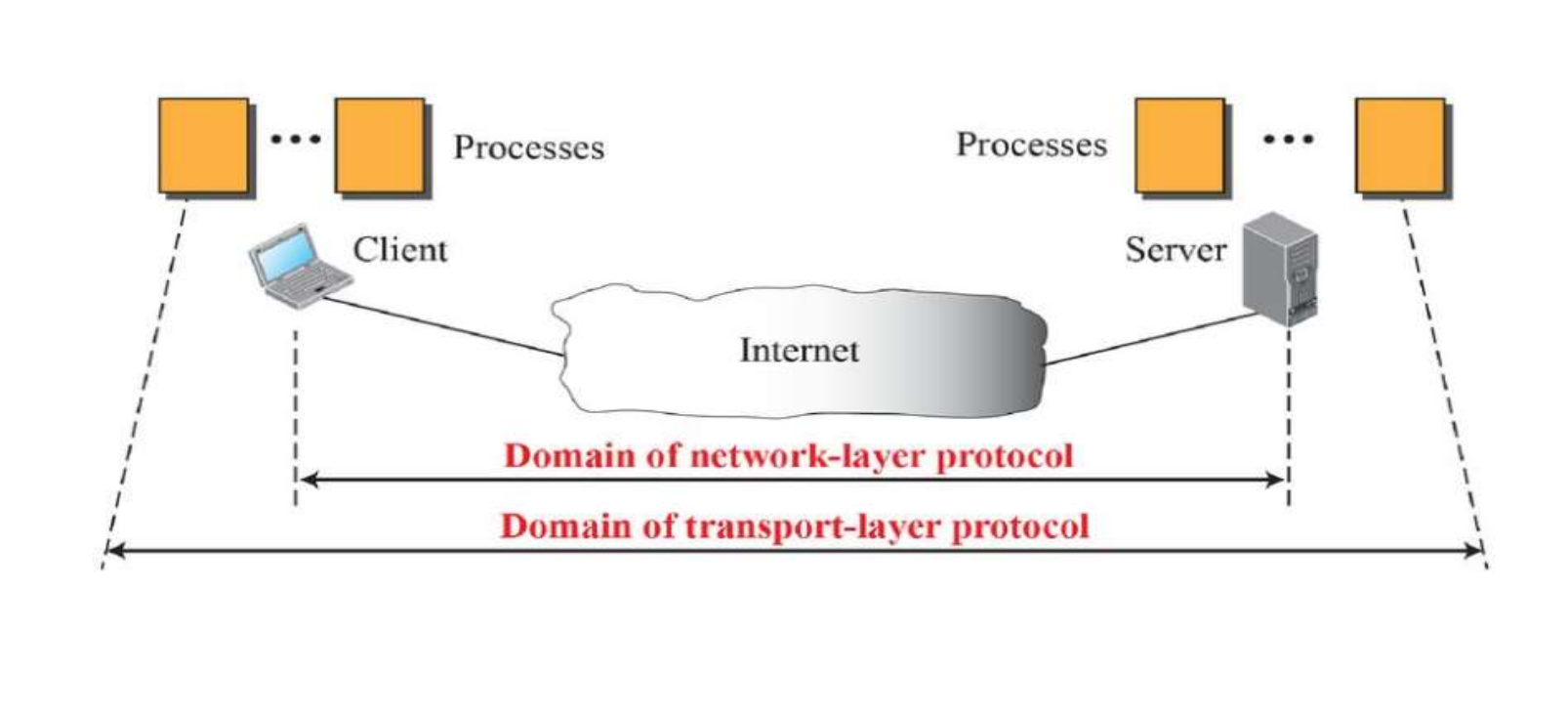
- Process to Process Communication
- User Datagram Protocol (UDP)- services, operation
- Transmission Control Protocol (TCP) - features, 3- way handshaking
- Comparison of UDP and TCP
- SCTP
- Congestion Control - open loop and close-loop;
- Quality of Service (QoS)
- QoS improving techniques - Leaky Bucket and Token Bucket algorithms.

INTRODUCTION

- The transport layer is responsible for process-to-process delivery of the entire message.
- Computers often run several programs at the same time. For this reason, source-to-destination delivery means delivery not only from one computer to the next but also from a specific process on one computer to a specific process on the other.
- The transport layer header must therefore include a type of address called a service-point address in the OSI model and port number or port addresses in the Internet and TCP/IP protocol suite.
- A transport layer protocol can be either connectionless or connection-oriented.

PROCESS TO PROCESS COMMUNICATION

- A process is a application layer entity that uses the services of the transport layer.
- A network layer protocol can deliver the message only to the destination computer. However this is an incomplete delivery.



PROCESS TO PROCESS COMMUNICATION

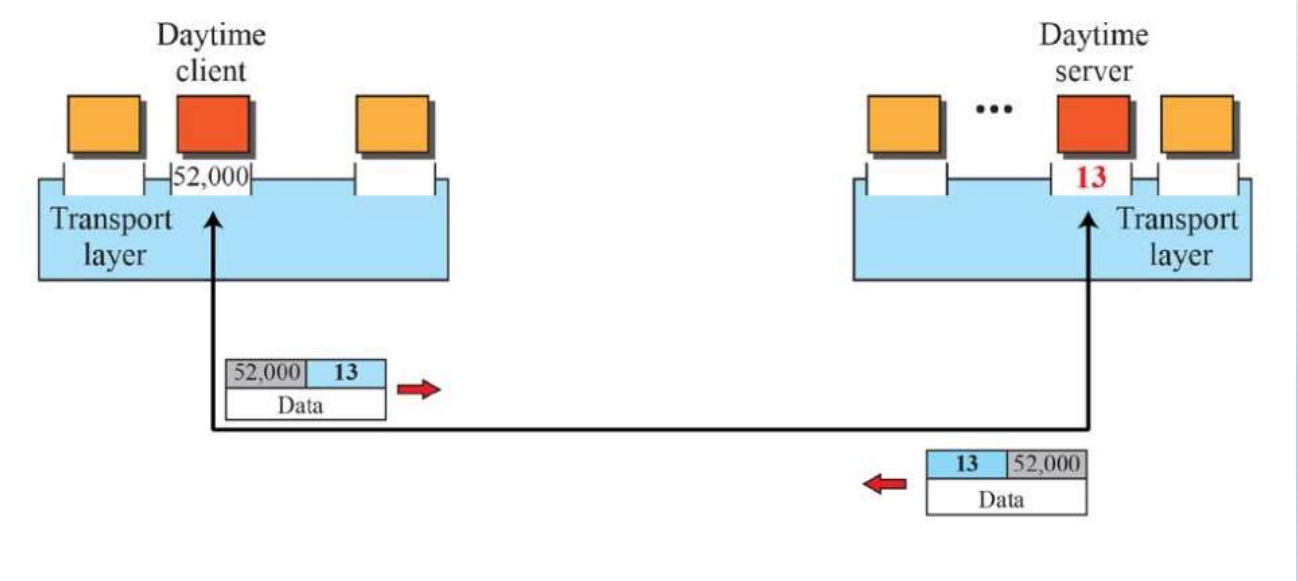
Client/Server Paradigm:

- To achieve process to process communication, the most common approach is through client/server paradigm.
- A process on the local host (client) needs service from a process on the remote host (server).
- At transport layer the addresses required are called port addresses (numbers).
- In Internet model the port numbers are 16-bit integers from 0 to 65,535.
- The client program defines itself with a port number chosen randomly.
- The server process must also define itself with a port number.
- Internet uses well-known port numbers for server processes

PROCESS TO PROCESS COMMUNICATION

Addressing: Port Numbers

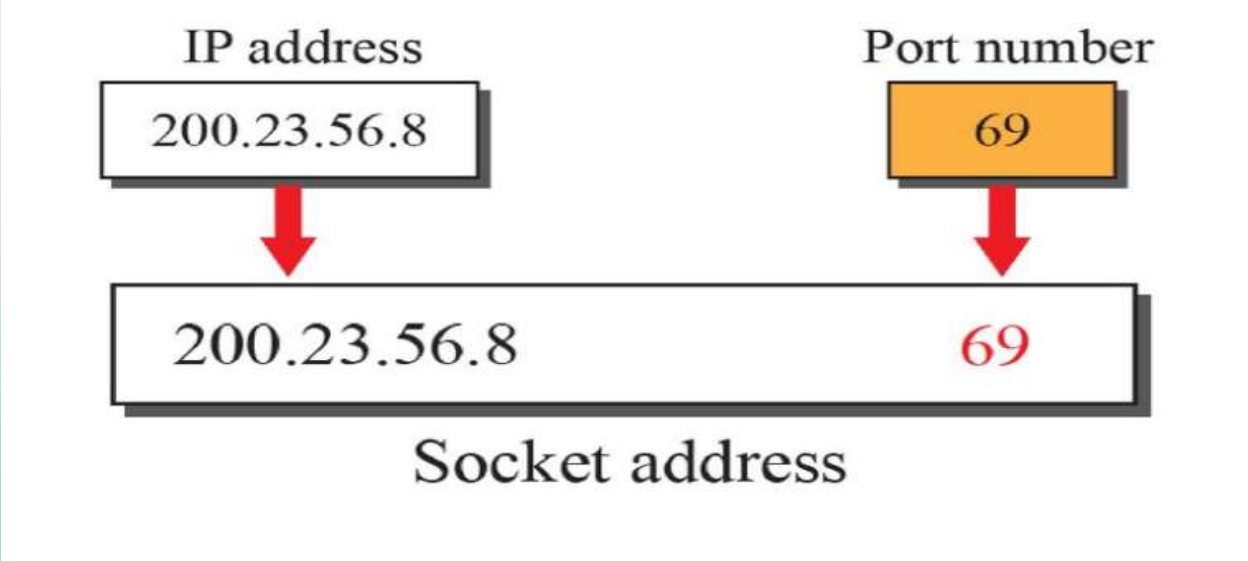
- A client program defines itself with a ephemeral port number. It is recommended to have an ephemeral port number greater than 1023 for some client/server program to work properly.
- A server process identifies itself with a well known port number.



PROCESS TO PROCESS COMMUNICATION

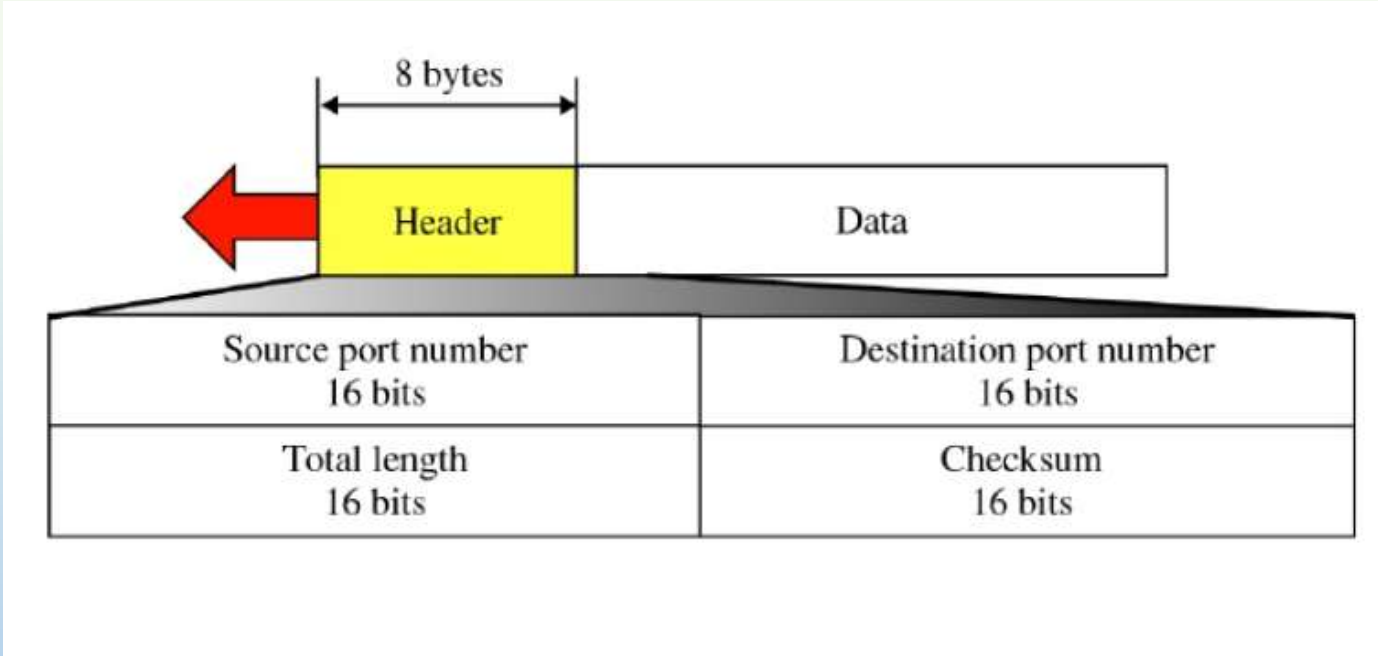
Socket Address

- A transport layer protocol in the TCP suite needs both the IP address and the port number , at each end to make a connection.
- The combination of IP address and Port number is called as Socket Address.
- The client socket address defines the client process uniquely



USER DATAGRAM PROTOCOL

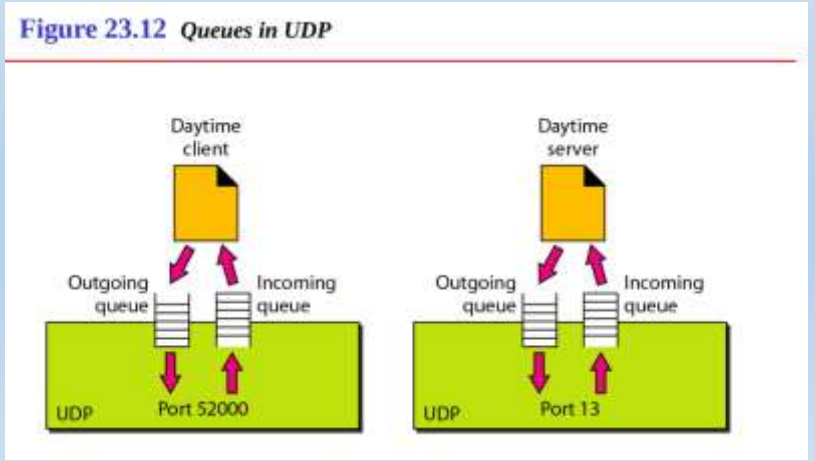
- The UDP is a connectionless, unreliable transport protocol.
- UDP is very simple protocol using a minimum of overhead.



USER DATAGRAM PROTOCOL

UDP Operations:

1. Process-to-Process Communication
2. Connectionless Service: UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
3. Checksum: The UDP checksum calculation is different from the one for IP and ICMP. Here the checksum includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.



USER DATAGRAM PROTOCOL

The following lists some uses of the UDP protocol:

1. UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
2. UDP is suitable for a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
3. UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
4. UDP is used for management processes such as SNMP.
5. UDP is used for some route updating protocols such as Routing Information Protocol

TRANSMISSION CONTROL PROTOCOL

- 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.

TRANSMISSION CONTROL PROTOCOL SERVICES

TCP offers following services to the processes at the application layer

1. Process to process Communication
2. Stream Delivery Service
3. Sending and Receiving Buffers
4. Segmentation
5. Full duplex communication
6. Connection oriented service
7. Reliable service

TRANSMISSION CONTROL PROTOCOL SERVICES

1. **Process to process Communication**

- Like UDP, TCP provides process-to-process communication using port numbers.

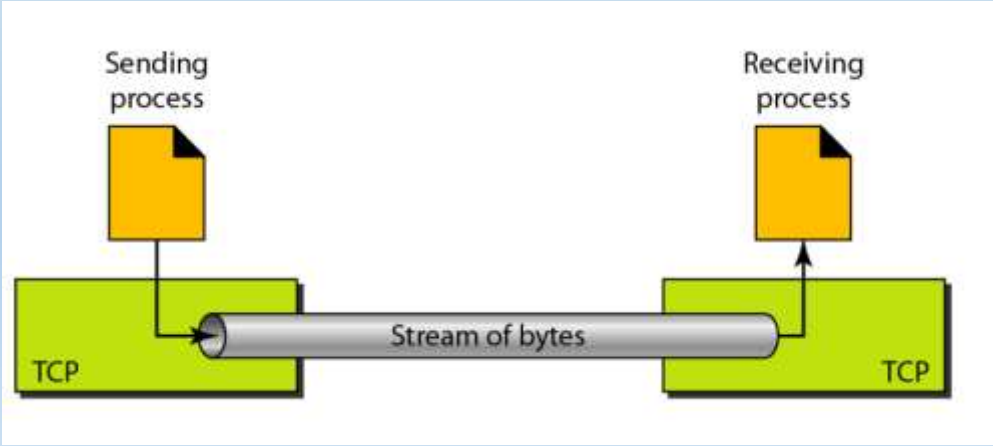
Table 23.2 *Well-known ports used by TCP*

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	FIP, Data	File Transfer Protocol (data connection)
21	FIP, Control	File Transfer Protocol (control connection)
23	TELNET	Tenninal 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

TRANSMISSION CONTROL PROTOCOL SERVICES

2. **Stream Delivery Service**

- TCP, unlike UDP, is a stream-oriented protocol.
- In UDP, a process (an application program) sends messages, with predefined boundaries, to UDP for delivery.
- UDP adds its own header to each of these messages and delivers them to IP for transmission.
- TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
- TCP creates an environment in which the two processes seem to be connected by an imaginary "tube“ that carries their data across the Internet.
- The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.



TRANSMISSION CONTROL PROTOCOL SERVICES

3. Sending and Receiving Buffers

- Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage.
- There are two buffers, the sending buffer and the receiving buffer, one for each direction.
- One way to implement a buffer is to use a circular array of 1-byte locations.

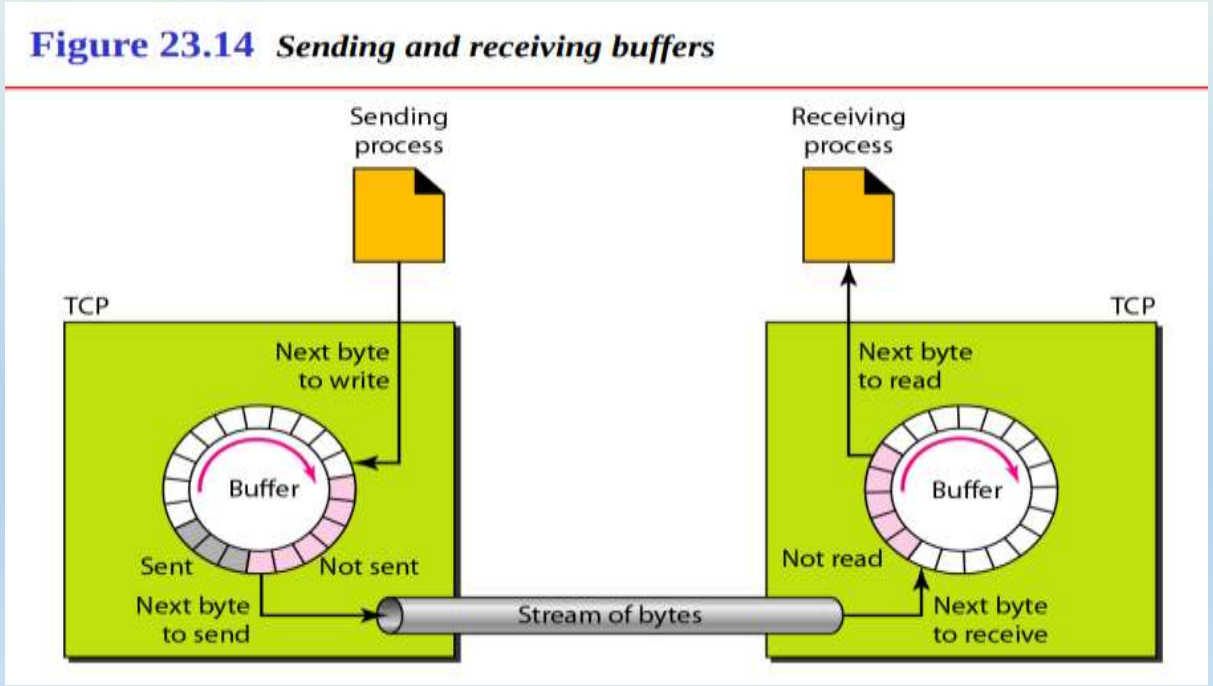
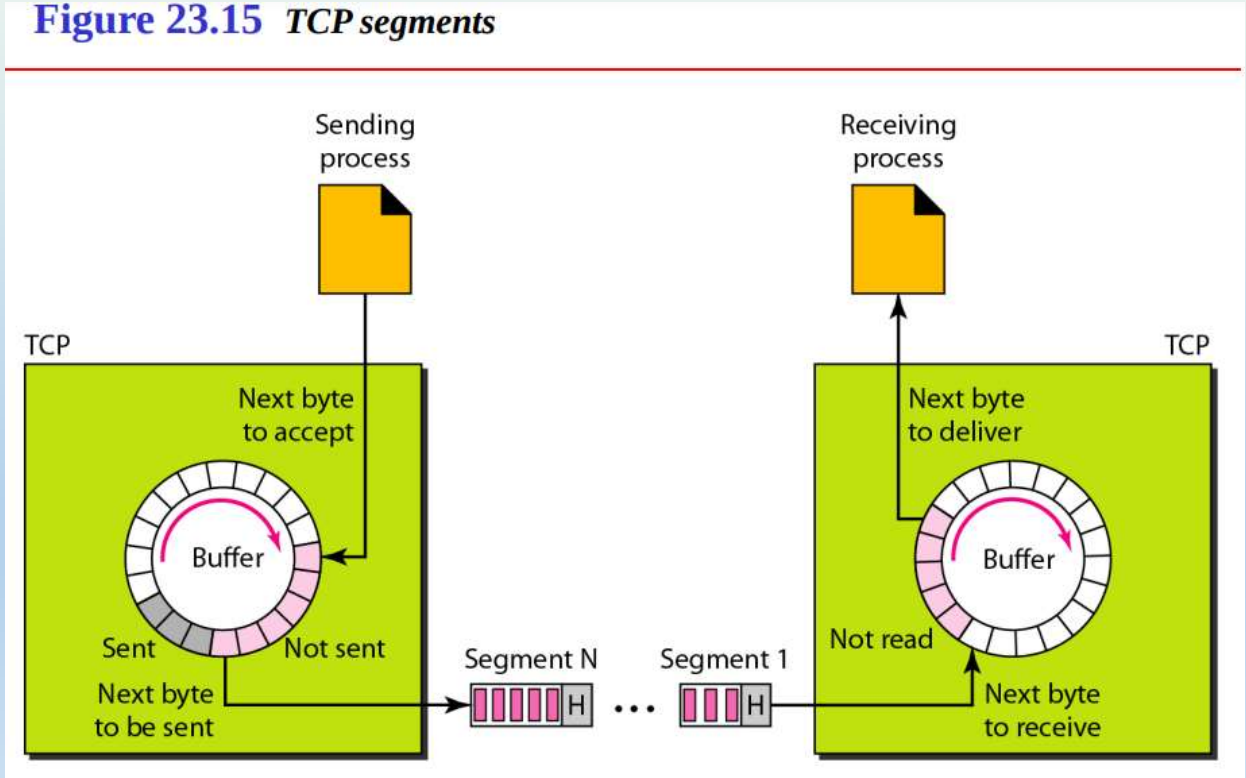


Figure 23.14 *Sending and receiving buffers*

TRANSMISSION CONTROL PROTOCOL SERVICES

4. Segmentation

- Although buffering handles the disparity between the speed of the producing and consuming processes, we need one more step before we can send data.
- The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes.
- At the transport layer, TCP groups a number of bytes together into a packet called a segment.



TRANSMISSION CONTROL PROTOCOL SERVICES

5. Full duplex communication

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

6. Connection oriented service

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

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

TRANSMISSION CONTROL PROTOCOL SERVICES

7.Reliable service

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

TRANSMISSION CONTROL PROTOCOL

TCP has several features for supporting the services provided by it

1. Numbering System
2. Flow Control
3. Error Control
4. Congestion(Traffic) Control

TRANSMISSION CONTROL PROTOCOL

1. **Numbering System**

- Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header.
- Instead, there are two fields called the sequence number and the acknowledgment number.
- These two fields refer to the byte number and not the segment number.
- **Byte Number :**TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.
- **Sequence Number:** After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.
- **Acknowledgment number:** The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

TRANSMISSION CONTROL PROTOCOL

Example:

- TCP wants to transfer a file of 5000 bytes. The first byte number is 10001. what are the sequence numbers, if data is needed to send in 5 segments?

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)**

TRANSMISSION CONTROL PROTOCOL

2. Flow Control

- TCP, unlike UDP, provides flow control. The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data.

3. Error Control

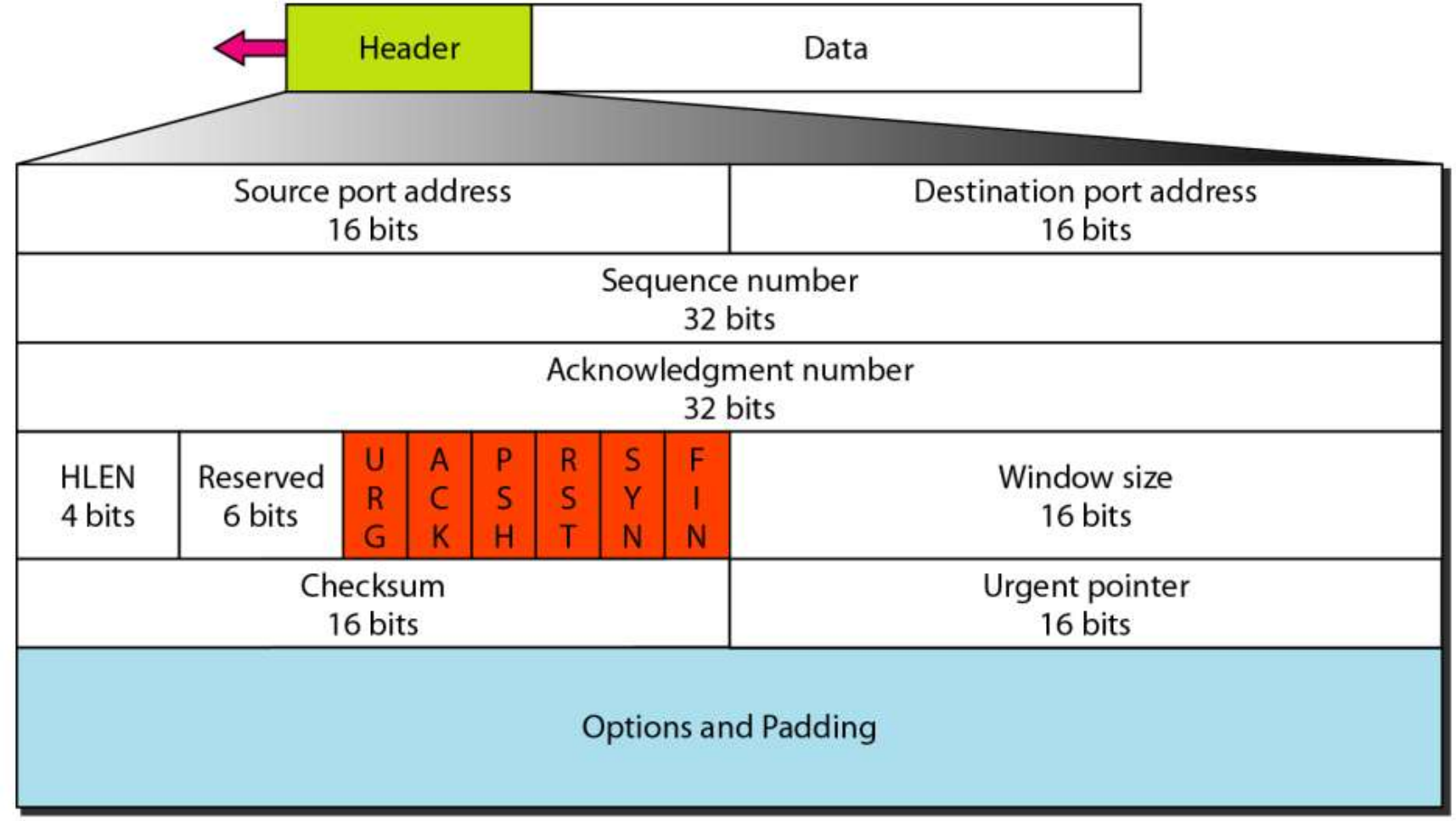
- To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

4. Congestion Control

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

TRANSMISSION CONTROL PROTOCOL SEGMENT

Figure 23.16 *TCP segment format*



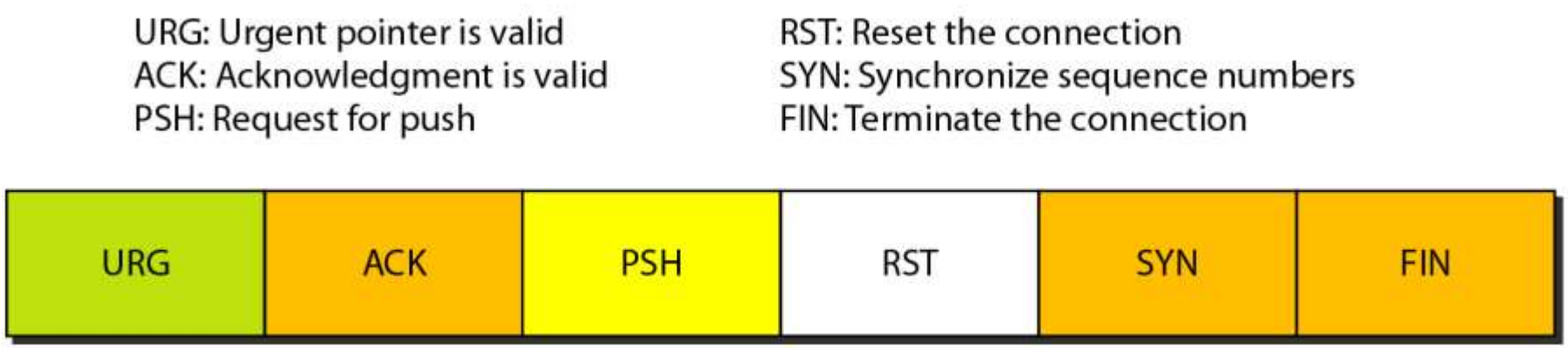
TRANSMISSION CONTROL PROTOCOL SEGMENT

- The segment consists of a 20- to 60-byte header, followed by data from the application program.
- **Source port address:** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
- **Destination port address:** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.
- **Sequence number:** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered.
- **Acknowledgment number:** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. Acknowledgment and data can be piggybacked together.
- **Header length:** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.
- **Reserved:** This is a 6-bit field reserved for future use.

TRANSMISSION CONTROL PROTOCOL SEGMENT

- **Control:** This field defines 6 different control bits or flags .One or more of these bits can be set at a time.

Figure 23.17 *Control field*



TRANSMISSION CONTROL PROTOCOL

- **Window size:** This field defines the size of the window, in bytes, that the other party must maintain.
- **Checksum:** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory.
- **Urgent pointer:** This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
- **Options:** There can be up to 40 bytes of optional information in the TCP header.

A TCP CONNECTION

- TCP is connection oriented.
- It establishes a virtual path between the source and destination
- All the segments belonging to a message are sent over this virtual path.
- This virtual path now is responsible for data transfer, the acknowledgement process as well as retransmission of damaged or lost frames
- TCP requires three phases for transmission:

1. Connection Establishment
2. Data transfer
3. Connection Release(termination)

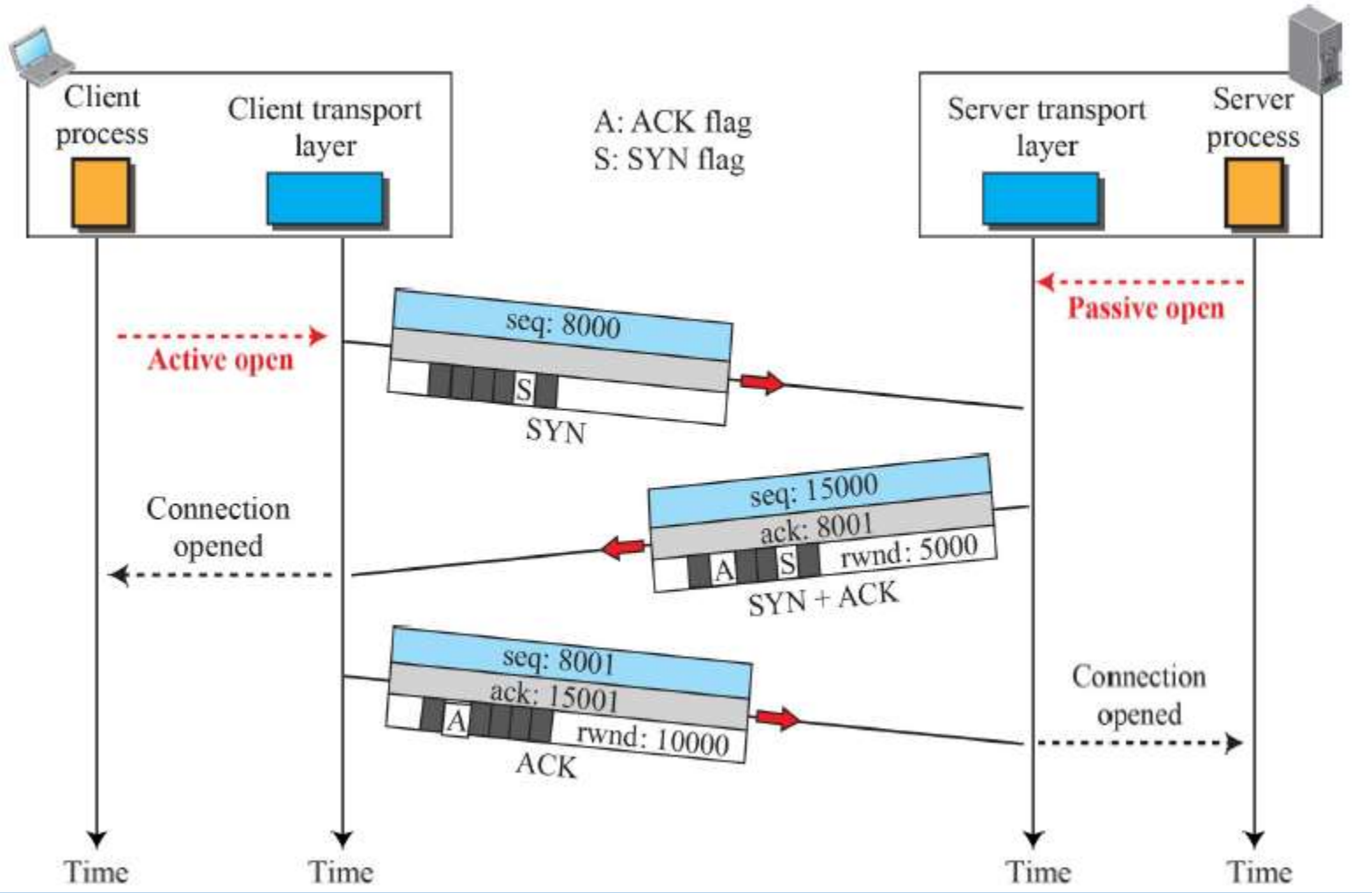
A TCP CONNECTION- CONNECTION ESTABLISHMENT

- TCP transmits data in full duplex mode.
- Before actual data transfer, each party must initialize communication.
- TCP does this by a “Three-Way handshaking”

Three-Way Handshaking

- Assume, an application program called ‘client’, wants to make a connection with another application program.
- The process starts with server, server TCP first initiates a ***passive open*** (telling TCP entity that it is ready to accept a connection).
- The client program issues a request for an ***active open.***
- TCP on the client machine starts this by three-way handshaking process (three signals SYN, ACK+SYN, ACK).
- The handshaking signals are segments shown with only important fields

A TCP CONNECTION- CONNECTION ESTABLISHMENT



A TCP CONNECTION

Simultaneous open

- When both processes, at client and server issues an active open, it is called simultaneous open.
- In this case, both TCPs transmit a SYN+ACK segment to each other and a single connection is established between them.

Resource Allocation

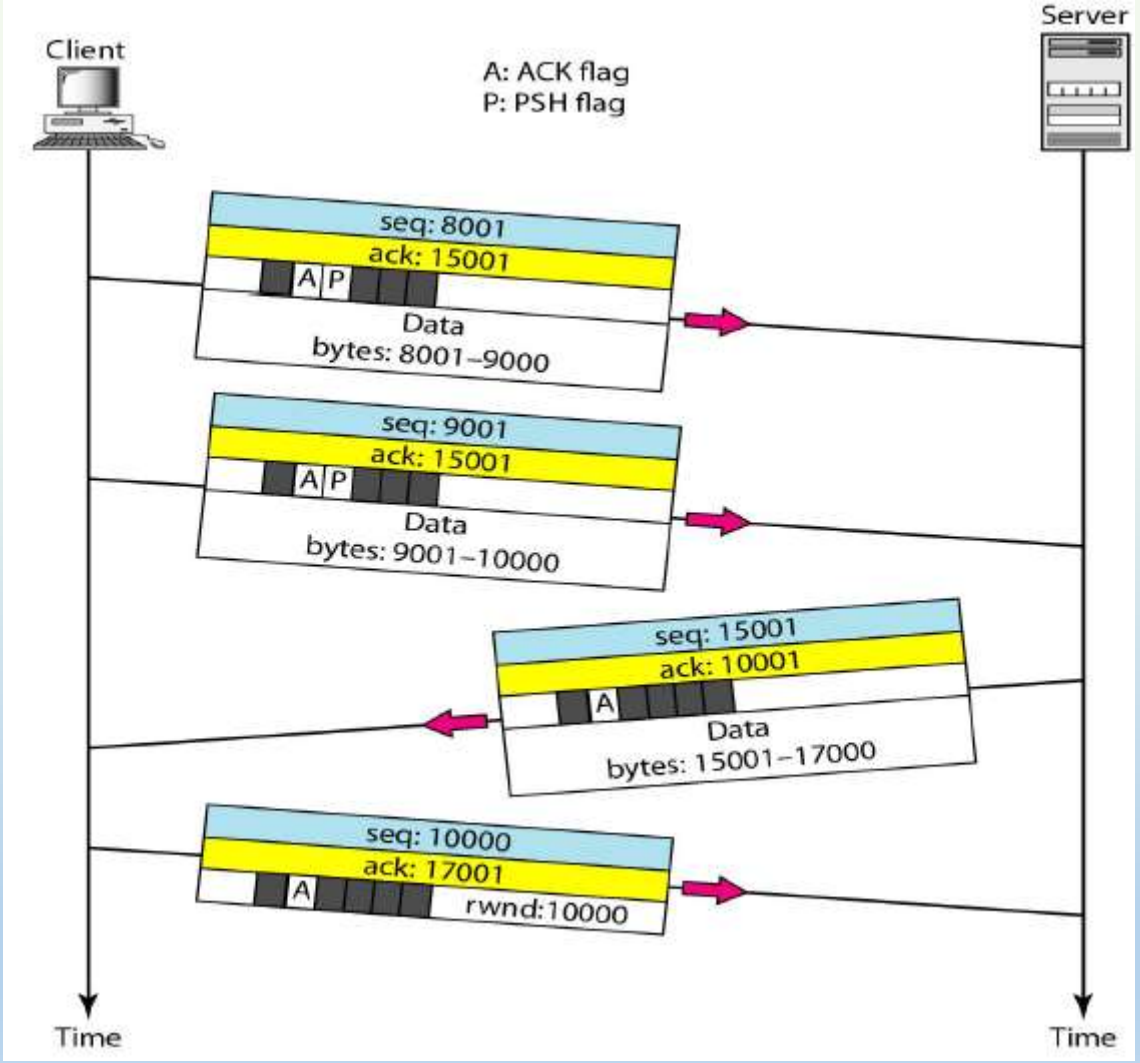
- Once three way handshake is successful, both TCP's allocate required resources to the connection

1. Port activation
2. Sending and receiving buffer allocation
3. Stream Delivery Service
4. Segmentation
5. Full duplex communication

A TCP CONNECTION- DATA TRANSFER

- After connection establishment, Data & acknowledgements can flow in both directions simultaneously.
- The acknowledgment is usually piggybacked

A TCP CONNECTION- DATA TRANSFER



A TCP CONNECTION- DATA TRANSFER

Pushing Operation

- Required in case of real time data transfer.
- The application programs require on time services
- Set PSH bit means

1. Sending TCP should not wait to make segment
2. Receiving TCP should deliver the data fast

Urgent Data

- Applications might need to send urgent data to the other party
- Set URG bit means

1. Sending TCP should place this data ahead of the any other data.
2. Receiving TCP should first deliver this data to receiving program

A TCP CONNECTION- CONNECTION TERMMINATION

- Any of the two parties (client or server) can close the connection
- The available closing options are:

1. Three way handshaking
2. Four way handshaking with a half-close option

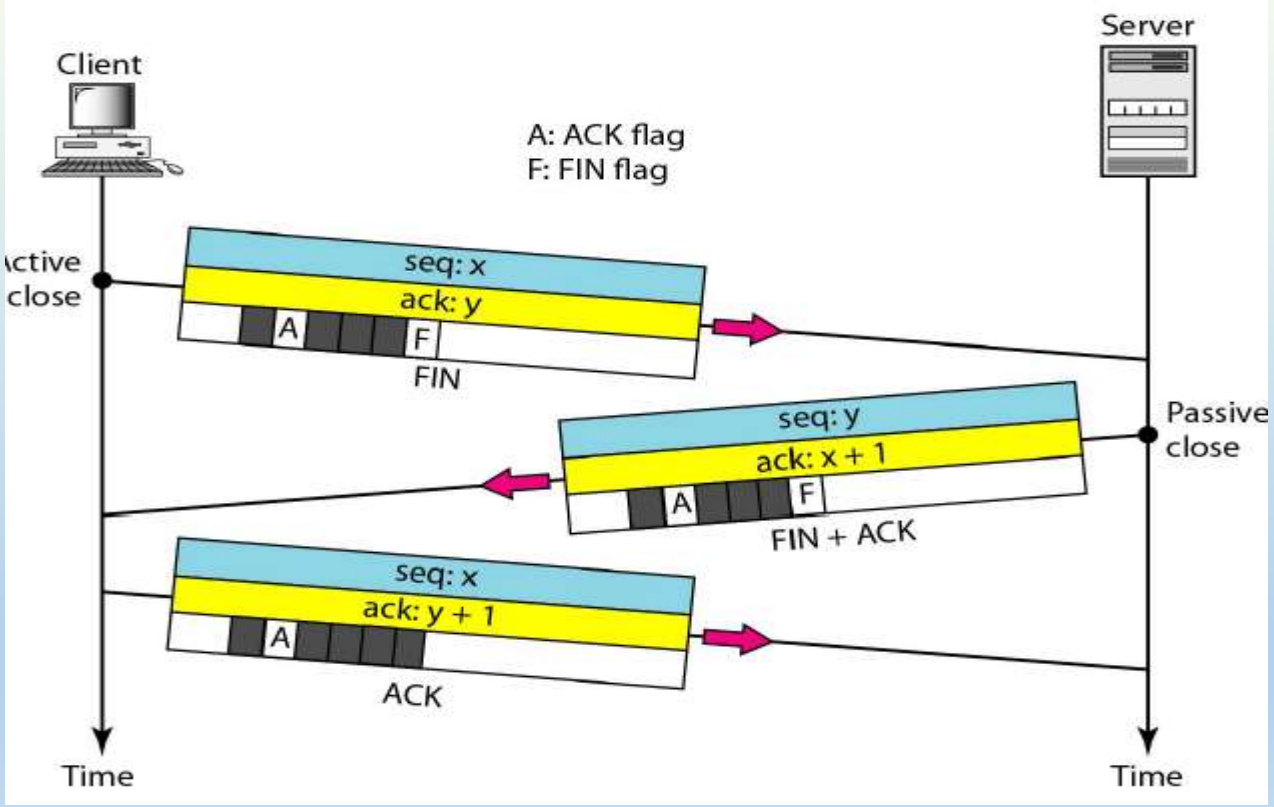
A TCP CONNECTION- CONNECTION TERMMINATION

Three way handshaking

- The client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment (with a FIN flag set).
- The FIN segment can include the last chunk of data from client OR it can be just a control segment.
- The FIN segment consumes one sequence number even if it does not carry data.
- The server now sends a FIN+ACK segment, to confirm the receipt of FIN segment as well as announcing the closing of the connection.
- This segment also can contain the last chunk of data from server
- The FIN+ACK segment also consumes one sequence number if it does not carry data.

A TCP CONNECTION- CONNECTION TERMMINATION

- The client TCP sends the last segment, an ACK segment to confirm the receipt of FIN segment from server.
- This segment contains ACK number (seq.no+1).
- This segment can not carry data and consumes no sequence number

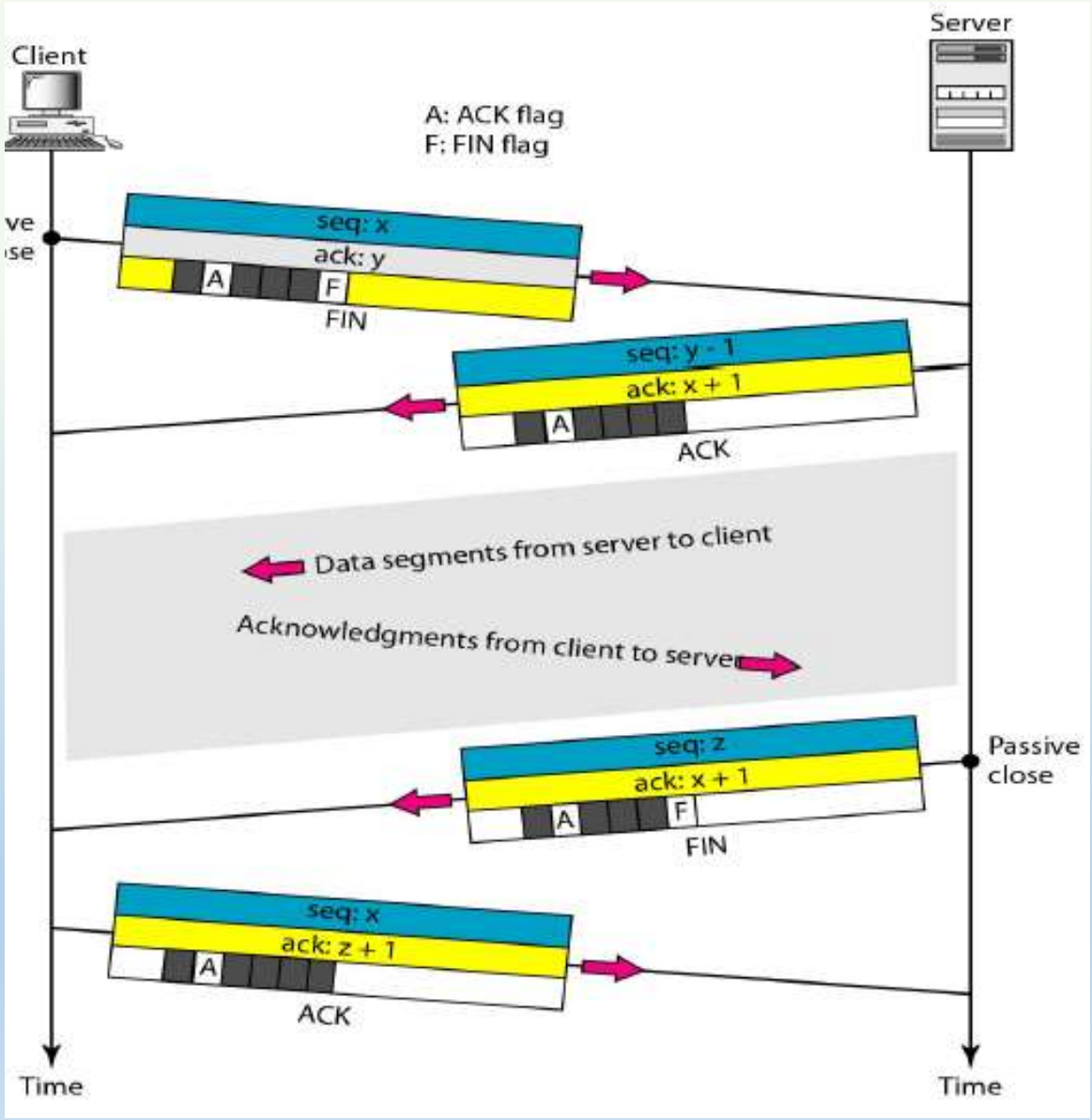


A TCP CONNECTION- CONNECTION TERMMINATION

Four way handshaking with a half-close option.

- In TCP, one end can stop sending data while still receiving data.
- The case is called Half-Close.
- It can occur when the server needs all the data before processing can begin. E.g. – sorting.
- In half-closing, client half closes the connection by sending a FIN segment.
- The server accepts the half-close by sending the ACK segment.
- Now the data transfer from client to server stops.
- The server can still send the data. E.g. result of sort.
- When server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.

A TCP CONNECTION- CONNECTION TERMMINATION



DIFFERENCE BETWEEN TCP AND UDP

Key	TCP	UDP
Definition	It is a communications protocol, using which the data is transmitted between systems over the network. In this, the data is transmitted in the form of packets. It includes error-checking, guarantees the delivery and preserves the order of the data packets.	It is same as the TCP protocol except this doesn't guarantee the error-checking and data recovery. If you use this protocol, the data will be sent continuously, irrespective of the issues in the receiving end.
Design	TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
Reliability	TCP is more reliable as it provides error checking support and also guarantees delivery of data to the destination router.	UDP, on the other hand, provides only basic error checking support using checksum. So, the delivery of data to the destination cannot be guaranteed in UDP as in case of TCP.
Data transmission	In TCP, the data is transmitted in a particular sequence which means that packets arrive in-order at the receiver.	There is no sequencing of data in UDP in order to implement ordering it has to be managed by the application layer.
Performance	TCP is slower and less efficient in performance as compared to UDP. Also TCP is heavy-weight as compared to UDP.	UDP is faster and more efficient than TCP.
Retransmission	Retransmission of data packets is possible in TCP in case packet get lost or need to resend.	Retransmission of packets is not possible in UDP.
Sequencing	The Transmission Control Protocol has a function that allows data to be sequenced (TCP). This implies that packets arrive at the recipient in the sequence they were sent.	In UDP, there is no data sequencing. The application layer must control the order if it is necessary.
Header size	TCP uses a variable-length (20-60) bytes header.	UDP has a fixed-length header of 8 bytes.
Handshake	Handshakes such as SYN, ACK, and SYNACK are used.	It's a connectionless protocol, which means it doesn't require a handshake.
Broadcasting	Broadcasting is not supported by TCP.	Broadcasting is supported by UDP.
Examples	HTTP, HTTPs, FTP, SMTP, and Telnet use TCP.	DNS, DHCP, TFTP, SNMP, RIP, and VoIP use UDP.

SCTP

- Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol.
- SCTP, however, is mostly designed for Internet applications that have recently been introduced.
- These new applications need a more sophisticated service than TCP can provide.
- SCTP is a message-oriented, reliable protocol that combines the best features of UDP and TCP.

SCTP SERVICES

1. **Process-to-Process Communication :** SCTP uses all well-known ports in the TCP space.
2. **Multiple Streams:** SCTP allows multistream service in each connection, which is called as association.
3. **Multihoming:** The sending and receiving host can define multiple IP addresses in each end for an association. In this fault tolerant approach, when one path fails, another interface can be used for data delivery without interruption.
4. **Full duplex communication:** In SCTP data can flow in both directions at the same time.
5. **Connection Oriented Service:** In SCTP a connection is called an association.
6. **Reliable Service:** It uses acknowledgement mechanism to check the safe and sound arrival of data.

SCTP

Figure 23.27 *Multiple-stream concept*

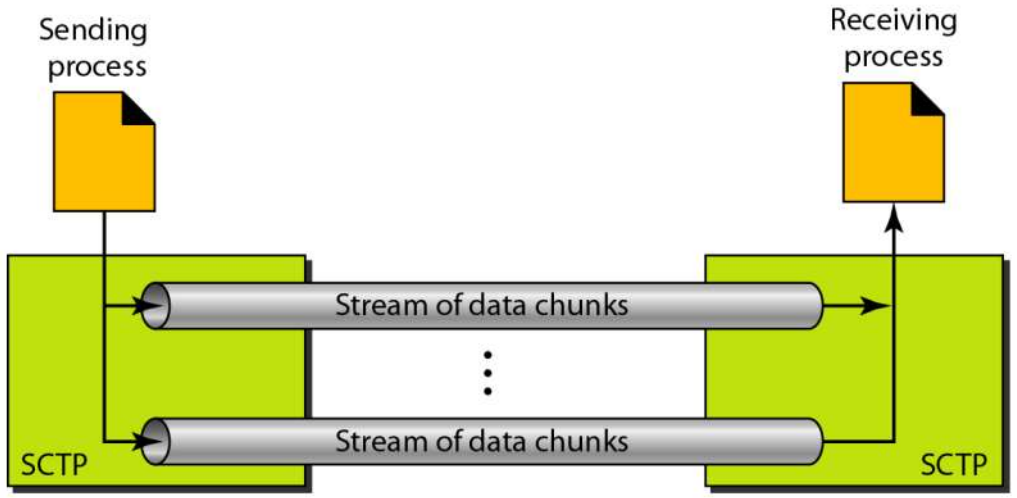
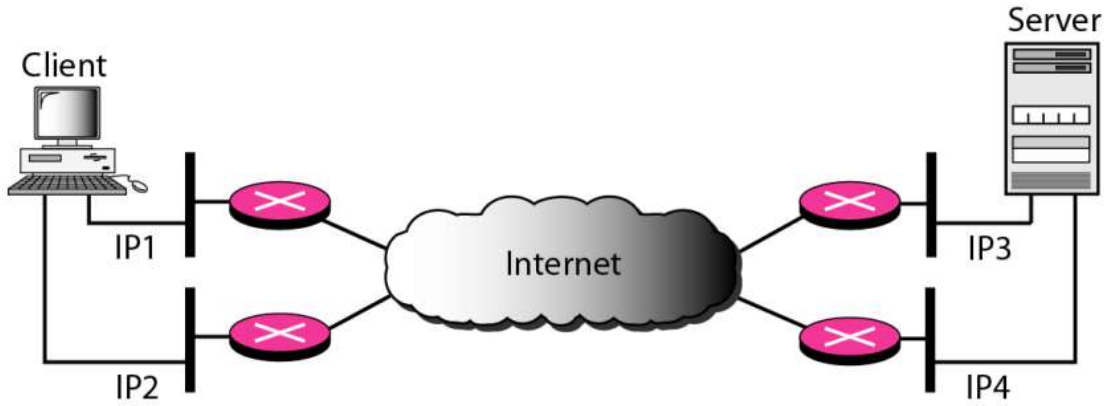


Figure 23.28 *Multihoming concept*



Feature	SCTP (Stream Control Transmission Protocol)	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Type	Connection-oriented	Connection-oriented	Connectionless
Message Boundaries	Preserves message boundaries	Does not preserve message boundaries	Preserves message boundaries
Reliability	Reliable, supports error recovery and retransmission	Reliable, ensures error recovery and retransmission	Unreliable, no guarantee of delivery or order
Connection Setup	Requires a handshake (4-way handshake)	Requires a handshake (3-way handshake)	No handshake, connectionless
Flow Control	Yes, uses flow control mechanisms	Yes, uses flow control (via sliding window)	No flow control
Congestion Control	Yes	Yes	No
Multi-Streaming	Yes, supports multi-streaming within a single connection	No	No
Multi-Homing	Yes, supports multi-homing (multiple IP addresses for redundancy)	No	No
Order of Data Delivery	Optional (can preserve or bypass order)	Guarantees ordered delivery	No guarantee of order
Error Detection	Yes, via checksum	Yes, via checksum	Yes, via checksum
Use Cases	Signaling, telephony, VoIP	Web browsing, email, file transfer	Streaming, gaming, DNS

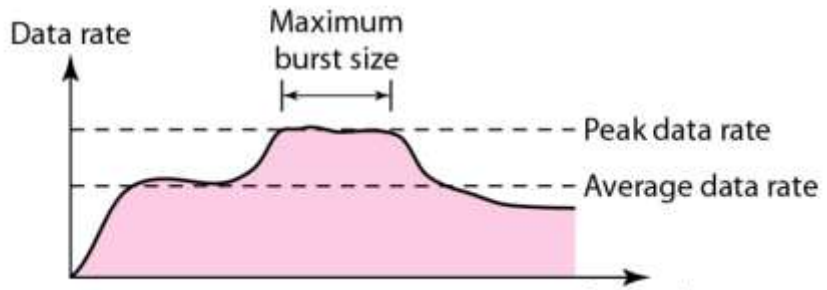
DATA TRAFFIC

- 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.

Traffic Descriptor

- Traffic descriptors are qualitative values that represent a data flow.

Figure 24.1 *Traffic descriptors*

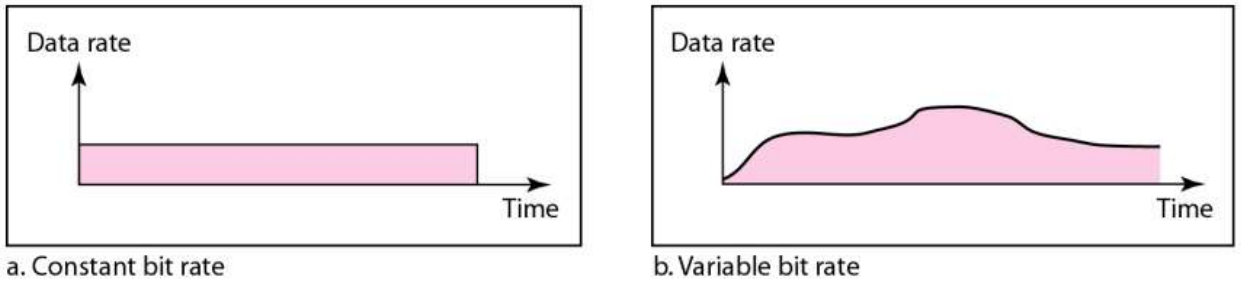


DATA TRAFFIC

Traffic Profiles

- For our purposes, a data flow can have one of the following traffic profiles

Figure 24.2 *Three traffic profiles*



a. Constant bit rate

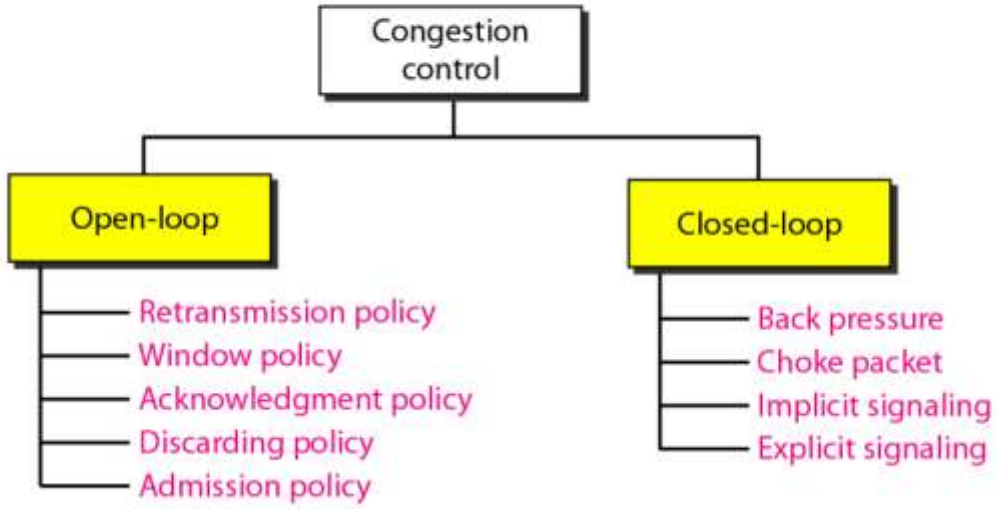
b. Variable bit rate

c. Bursty

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)

Figure 24.5 *Congestion control categories*



OPEN LOOP CONGESTION CONTROL

- In open-loop congestion control, policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination.

1. Retransmission Policy

- Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted.
- Retransmission in general may increase congestion in the network. However, a good retransmission policy can prevent congestion.
- The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.

2. **Window Policy**

- The type of window at the sender may also affect congestion.
- The Selective Repeat window is better than the Go-Back-N window for congestion control.
- The Selective Repeat window, tries to send the specific packets that have been lost or corrupted.

OPEN LOOP CONGESTION CONTROL

3. Acknowledgment Policy

- The acknowledgment policy imposed by the receiver may also affect congestion.
- If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.

4. Discarding Policy

- A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission.
- For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

OPEN LOOP CONGESTION CONTROL

5. Admission Policy

- An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks.
- Switches in a flow first check the resource requirement of a flow before admitting it to the network.
- A router can deny establishing a virtual circuit connection if there is congestion in the network or if there is a possibility of future congestion.

CLOSED LOOP CONGESTION CONTROL

Backpressure

- The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes.
- This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes.
- Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source.

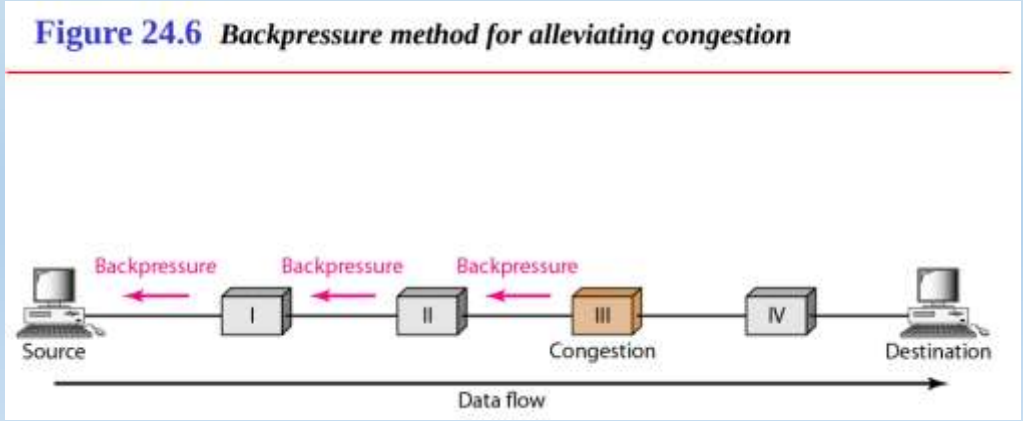
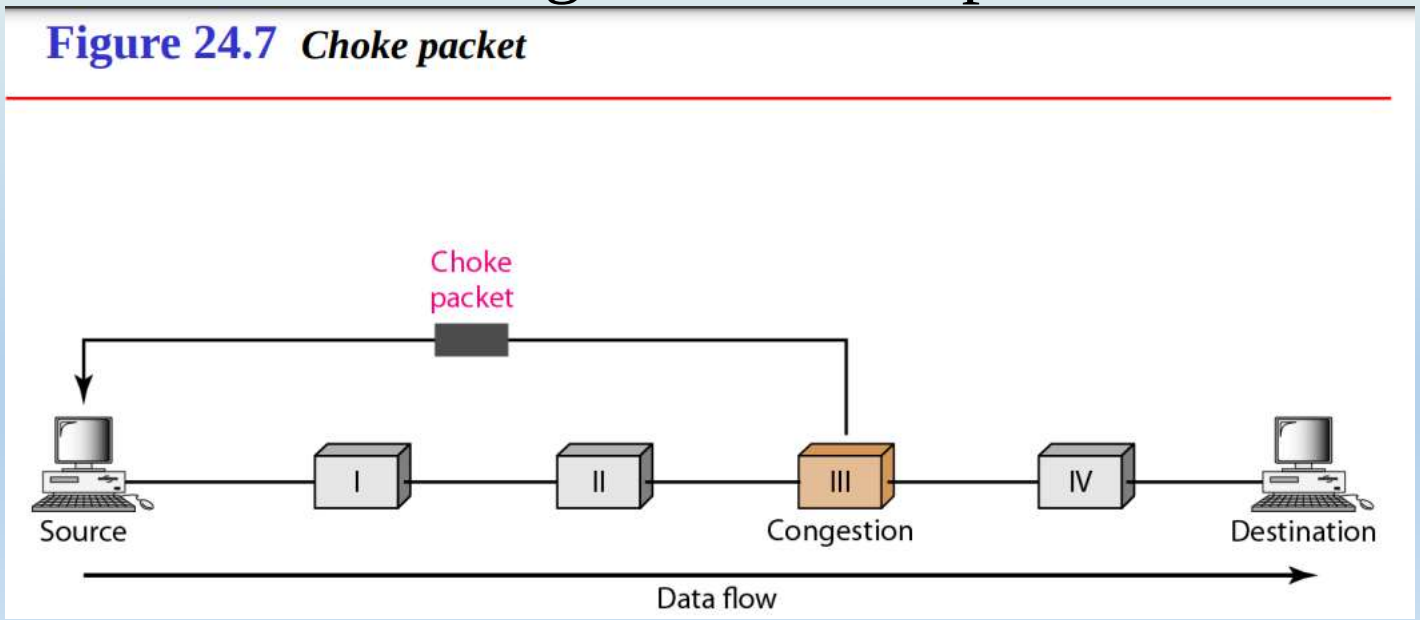


Figure 24.6 *Backpressure method for alleviating congestion*

CLOSED LOOP CONGESTION CONTROL

Choke Packet

- A choke packet is a packet sent by a node to the source to inform it of congestion.
- In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly.
- The intermediate nodes through which the packet has traveled are not warned.



CLOSED LOOP CONGESTION CONTROL

Implicit Signaling

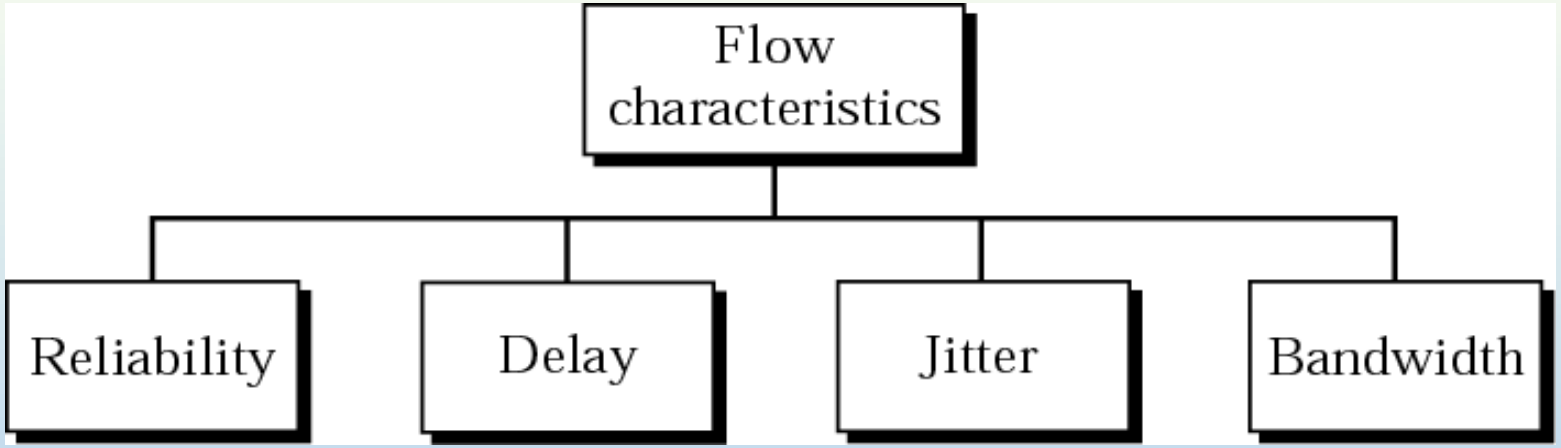
- In implicit signaling, there is no communication between the congested node or nodes and the source.
- The source guesses that there is a congestion somewhere in the network from other symptoms.

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 choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data

QUALITY OF SERVICE(QoS)

- We can define quality of service as something a flow seeks to attain.
- Traditionally, four types of characteristics are attributed to a flow.



QUALITY OF SERVICE(QoS)

Reliability

- Reliability is a characteristic that a flow needs. Lack of reliability means losing a packet or acknowledgment, which entails retransmission. However, the sensitivity of application programs to reliability is not the same.

Delay

- Source-to-destination delay is another flow characteristic. Again applications can tolerate delay in different degrees.

Jitter

- Jitter is the variation in delay for packets belonging to the same flow. For example, if four packets depart at times 0, 1, 2, 3 and arrive at 20, 21, 22, 23, all have the same delay, 20 units of time.

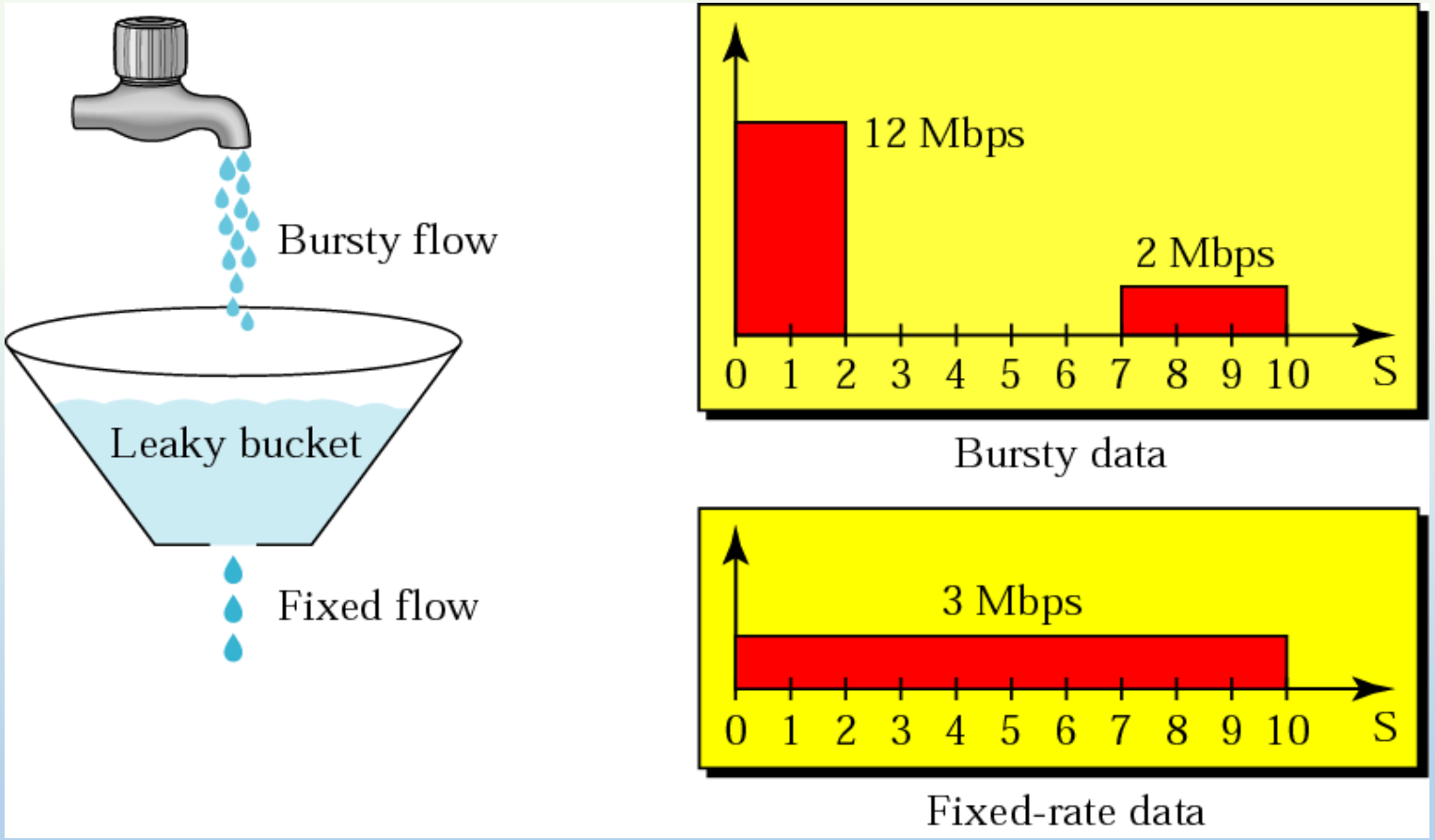
Bandwidth

- Different applications need different bandwidths. In video conferencing we need to send millions of bits per second to refresh a color screen while the total number of bits in an e-mail may not reach even a million.

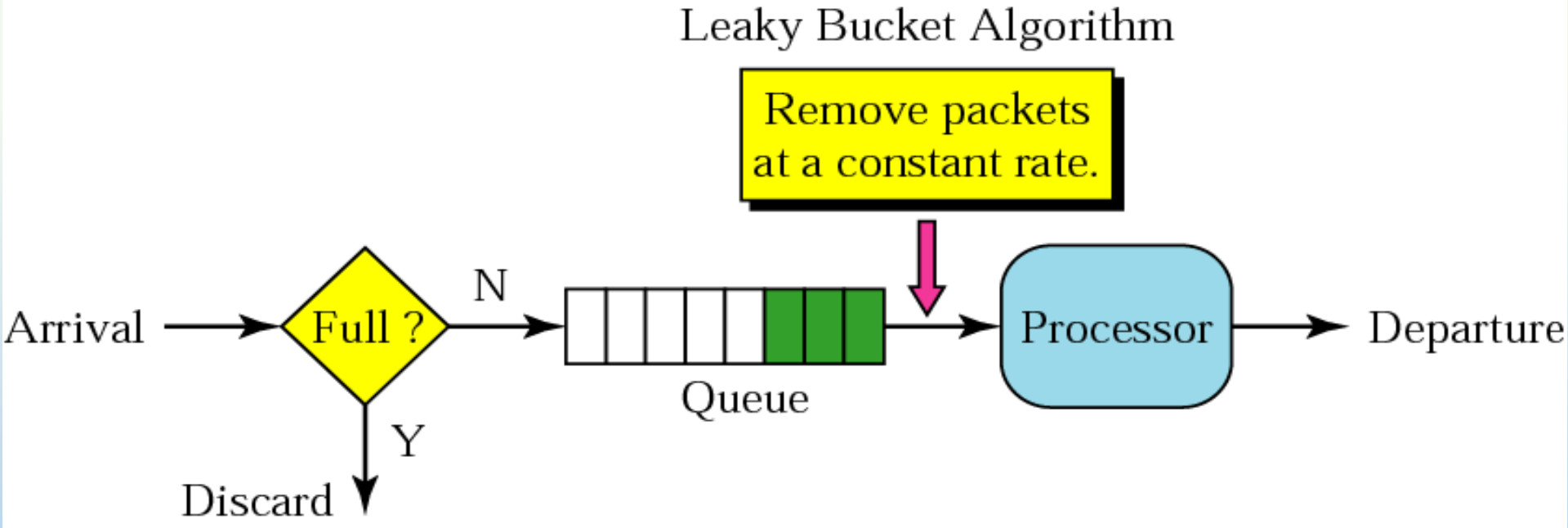
QoS IMPROVING TECHNIQUES-LEAKY BUCKET

- If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket.
- The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty.
- The input rate can vary, but the output rate remains constant.
- Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic.
- Bursty chunks are stored in the bucket and sent out at an average rate.

QoS IMPROVING TECHNIQUES-LEAKY BUCKET



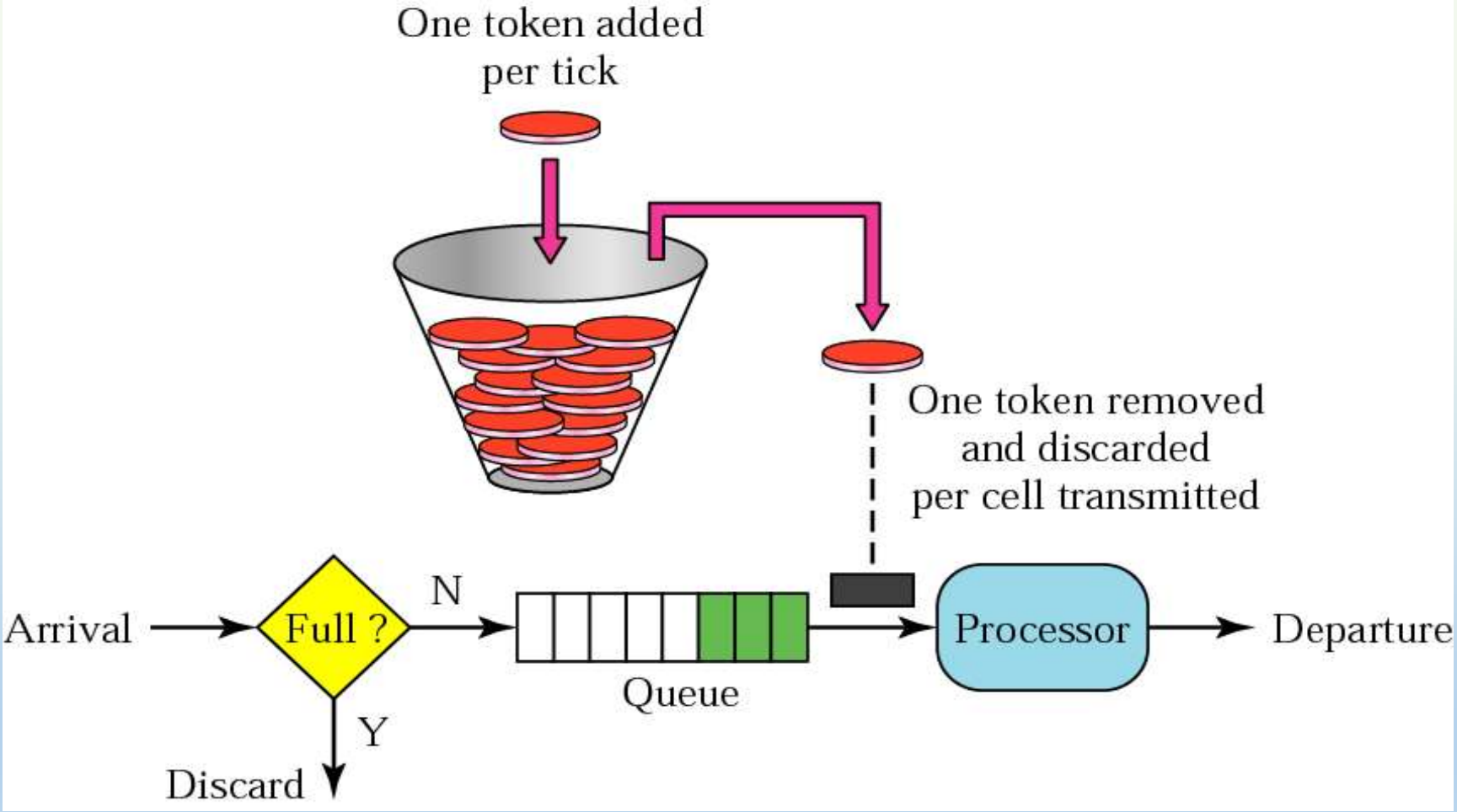
QoS IMPROVING TECHNIQUES-LEAKY BUCKET



QoS IMPROVING TECHNIQUES- TOKEN BUCKET

- The leaky bucket is very restrictive. It does not credit an idle host.
- For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate.
- The time when the host was idle is not taken into account. On the other hand, the token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens.
- For each tick of the clock, the system sends n tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if n is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens.
- Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick.
- In other words, the host can send bursty data as long as the bucket is not empty.
- The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

QoS IMPROVING TECHNIQUES- TOKEN BUCKET



THANK-YOU