UNIT IV

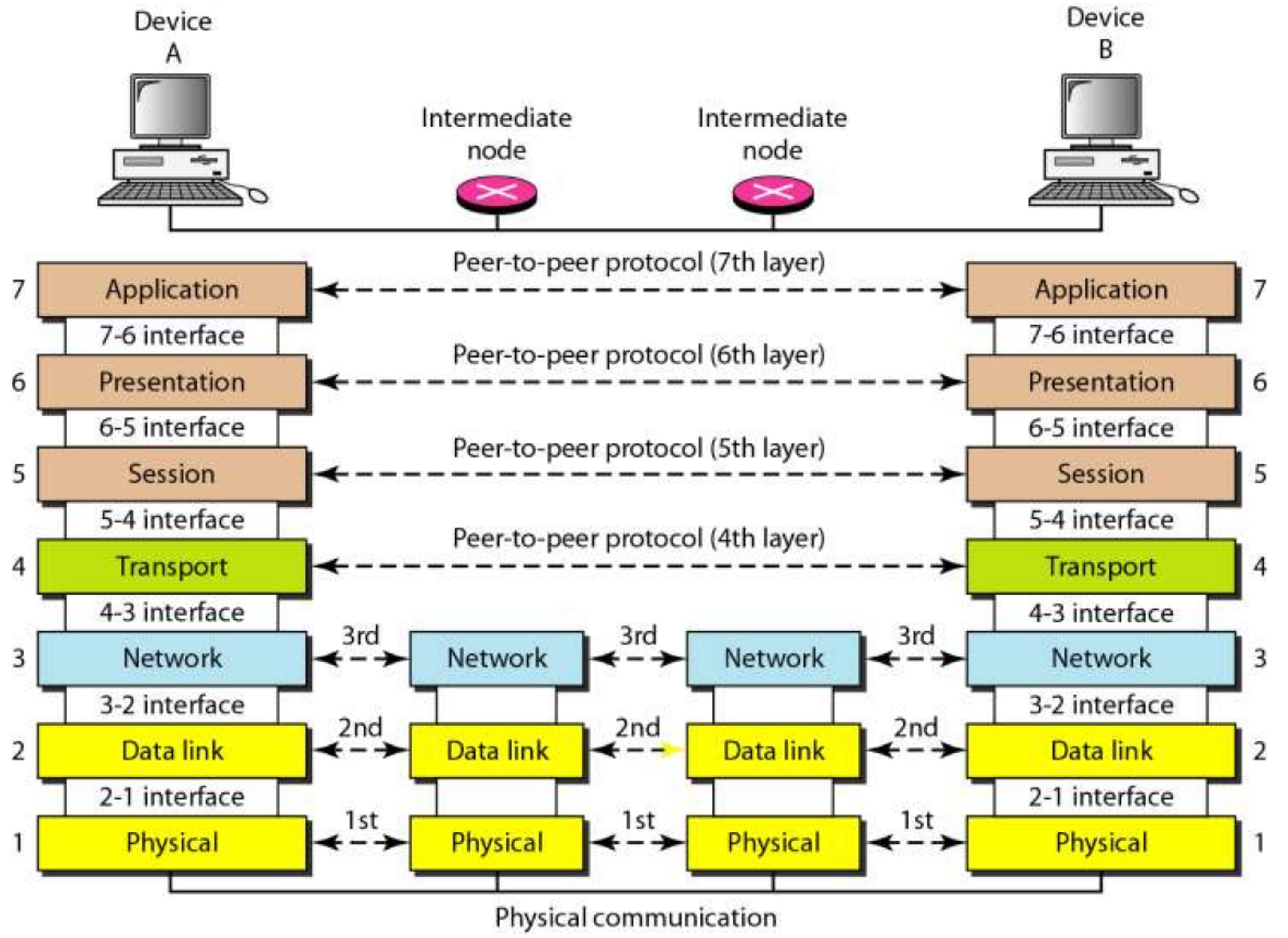
Transport Layer

Transport Layer

Process to Process Communication,
User Datagram Protocol (UDP),
Transmission Control Protocol (TCP),
Congestion control algorithms,
Quality of Service.

- Residing between the application and network layers, the transport layer is a central piece of the layered network architecture.
- It has the critical role of providing communication services(logical communication) directly to the ***application processes*** running on different hosts.
- A transport-layer protocol provides for logical communication between application processes running on different hosts.

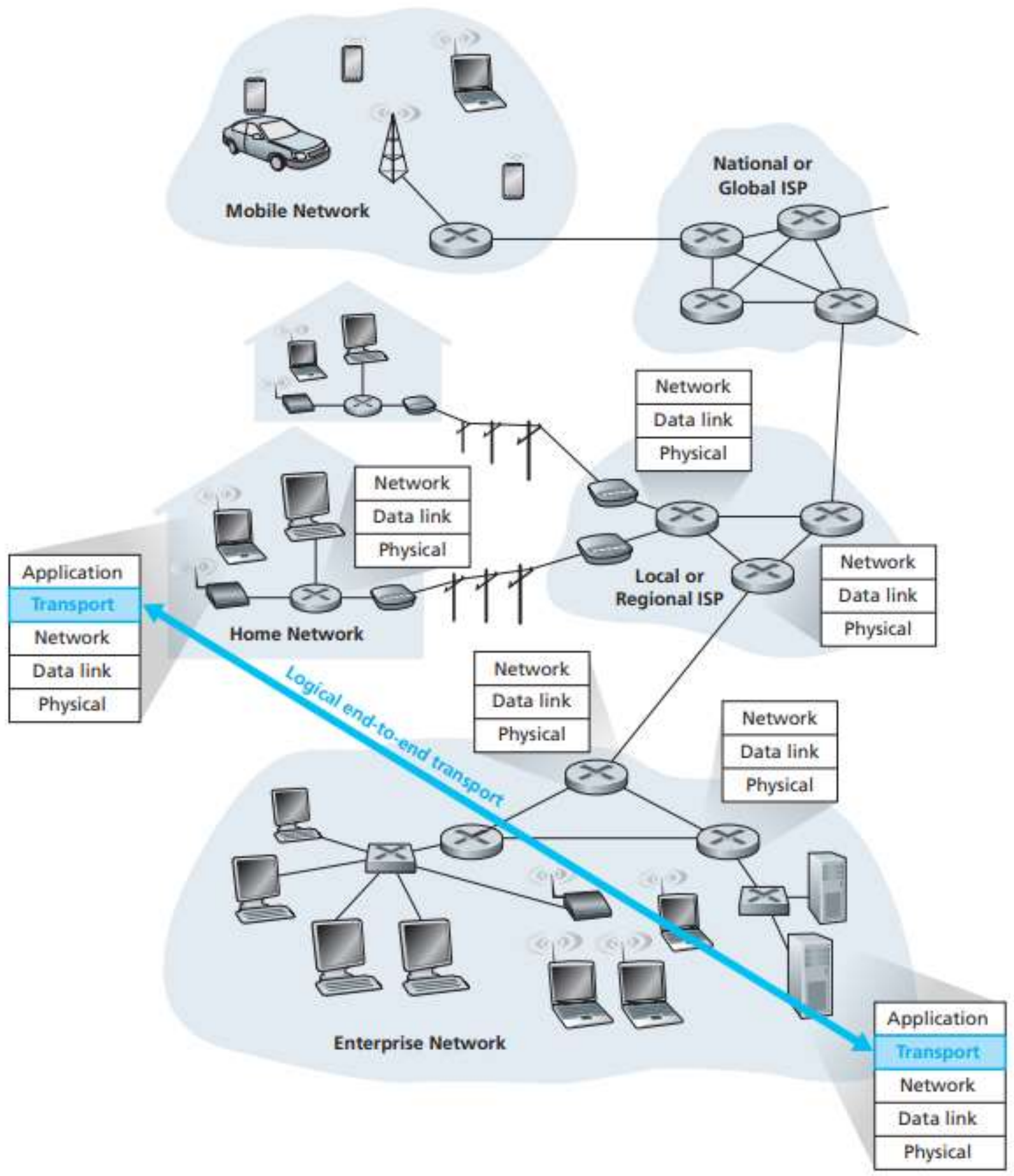
Figure *The interaction between layers in the OSI model*



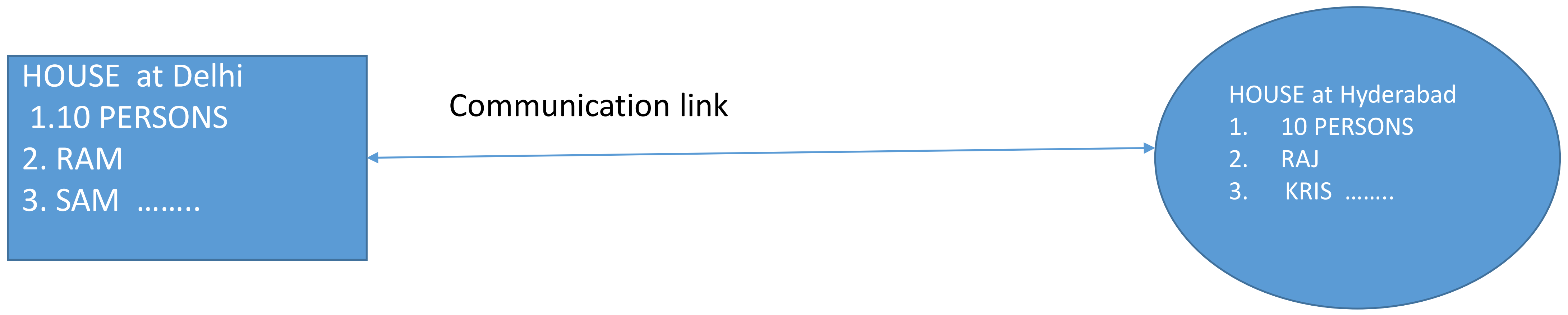
- Transport-layer protocols are implemented in the end systems but not in network routers.
- On the sending side, the transport layer converts the application-layer messages it receives from a sending application process into transport-layer packets, known as **transport-layer segments** in Internet terminology.
- This is done by (possibly) breaking the application messages into smaller chunks and adding a transport-layer header to each chunk to create the transport-layer segment.

- The transport layer then passes the segment to the network layer at the sending end system, where the segment is encapsulated within a network-layer packet (a datagram) and sent to the destination.
- It’s important to note that network routers act only on the network-layer fields of the datagram; that is, they do not examine the fields of the transport-layer segment encapsulated with the datagram.

- On the receiving side, the network layer extracts the transport-layer segment from the datagram and passes the segment up to the transport layer.
- The transport layer then processes the received segment, making the data in the segment available to the receiving application.



The transport layer provides logical rather than physical communication between application processes



application messages = letters in envelopes
processes = cousins
hosts (also called end systems) = houses
transport-layer protocol = RAM & RAJ or SAM & KRIS
network-layer protocol = postal service (including mail carriers)

a computer network may make available multiple transport protocols, with each protocol offering a different service model to applications.

- The services that a transport protocol can provide are often constrained by the service model of the underlying network-layer protocol.
- If the network-layer protocol cannot provide delay or bandwidth guarantees for transport layer segments sent between hosts, then the transport-layer protocol cannot provide delay or bandwidth guarantees for application messages sent between processes.

- Nevertheless, certain services can be offered by a transport protocol even when the underlying network protocol doesn’t offer the corresponding service at the network layer.
- For example, a transport protocol can offer reliable data transfer service to an application even when the underlying network protocol is unreliable, that is, even when the network protocol loses, garbles, or duplicates packets.

- TCP/IP network, makes two distinct transport-layer protocols available to the application layer.
- UDP (User Datagram Protocol), which provides an unreliable, connectionless service to the invoking application.
- TCP (Transmission Control Protocol), which provides a reliable, connection-oriented service to the invoking application.
- When designing a network application, the application developer must specify one of these two transport protocols.

- The Internet’s network-layer protocol has a name—IP, for Internet Protocol.
- IP provides logical communication between hosts.
- The IP service model is a best-effort delivery service.
- This means that IP makes its “best effort” to deliver segments between communicating hosts, but it makes no guarantees.
- In particular, it does not guarantee segment delivery, it does not guarantee orderly delivery of segments, and it does not guarantee the integrity of the data in the segments.
- For these reasons, IP is said to be an unreliable service.
- Every host has at least one network-layer address, so-called IP address.

- The most fundamental responsibility of UDP and TCP is to extend IP’s delivery service between two end systems to a delivery service between two processes running on the end systems.
- Extending host-to-host delivery to process-to-process delivery is called transport-layer multiplexing and de-multiplexing.
- UDP and TCP also provide integrity checking by including error detection fields in their segments’ headers.
- These two minimal transport-layer services—process-to-process data delivery and error checking—are the only two services that UDP provides.
- In particular, like IP, UDP is an unreliable service—it does not guarantee that data sent by one process will arrive intact (or at all!) to the destination process

- TCP, on the other hand, offers several additional services to applications.
- First and foremost, it provides reliable data transfer.
- Using flow control, sequence numbers, acknowledgments, and timers TCP ensures that data is delivered from sending process to receiving process, correctly and in order.
- TCP thus converts IP’s unreliable service between end systems into a reliable data transport service between processes.
- TCP also provides congestion control.
- TCP congestion control prevents any one TCP connection from swapping the links and routers between communicating hosts with an excessive amount of traffic.
- UDP traffic, on the other hand, is unregulated.
- An application using UDP transport can send at any rate it pleases, for as long as it pleases

SOCKET & PORT

- Processes running on different machines communicate with each other by sending messages into sockets.
- Each process is analogous to a house and the process’s socket is analogous to a door.
- The application resides on one side of the door in the house; the transport-layer protocol resides on the other side of the door in the outside world.
- The application developer has control of everything on the application-layer side of the socket; however, it has little control of the transport-layer side.

- A process (as part of a network application) can have one or more sockets, doors through which data passes from the network to the process and through which data passes from the process to the network.
- The transport layer in the receiving host does not actually deliver data directly to a process, but instead to an intermediary socket. Because at any given time there can be more than one socket in the receiving host, each socket has a unique identifier.
- The format of the identifier depends on whether the socket is a UDP or a TCP socket.

communicating processes that use UDP sockets

- Before the sending, process can push a packet of data out the socket door, when using UDP, it must first attach a destination address to the packet.
- After the packet passes through the sender’s socket, the Internet will use this destination address to route the packet through the Internet to the socket in the receiving process.
- When the packet arrives at the receiving socket, the receiving process will retrieve the packet through the socket, and then inspect the packet’s contents and take appropriate action.

- What goes into the destination address that is attached to the packet?
- As you might expect, the destination host’s IP address is part of the destination address.
- By including the destination IP address in the packet, the routers in the Internet will be able to route the packet through the Internet to the destination host.
- But because a host may be running many network application processes, each with one or more sockets, it is also necessary to identify the particular socket in the destination host.
- When a socket is created, an identifier, called a port number, is assigned to it.

- So, as you might expect, the packet’s destination address also includes the socket’s port number.
- In summary, the sending process attaches to the packet a destination address which consists of the destination host’s IP address and the destination socket’s port number.
- Moreover, the sender’s source address— consisting of the IP address of the source host and the port number of the source socket—are also attached to the packet.
- However, attaching the source address to the packet is typically not done by the UDP application code; instead it is automatically done by the underlying operating system

USER DATAGRAM PRPTOL (UDP)

User Datagram Protocol (UDP)

- **User Datagram Protocol (UDP)** is a Transport Layer protocol.
- UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite.
- Unlike TCP, it is an **unreliable and connectionless protocol.**
- So, there is no need to establish a connection prior to data transfer.

- 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.
- Also, it performs very limited error checking.

- UDP is so powerless, why would a process want to use it? With the disadvantages come some advantages.
- UDP is a very simple protocol using a minimum of overhead.
- If a process wants to send a small message and does not care much about reliability, it can use UDP.
- Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP

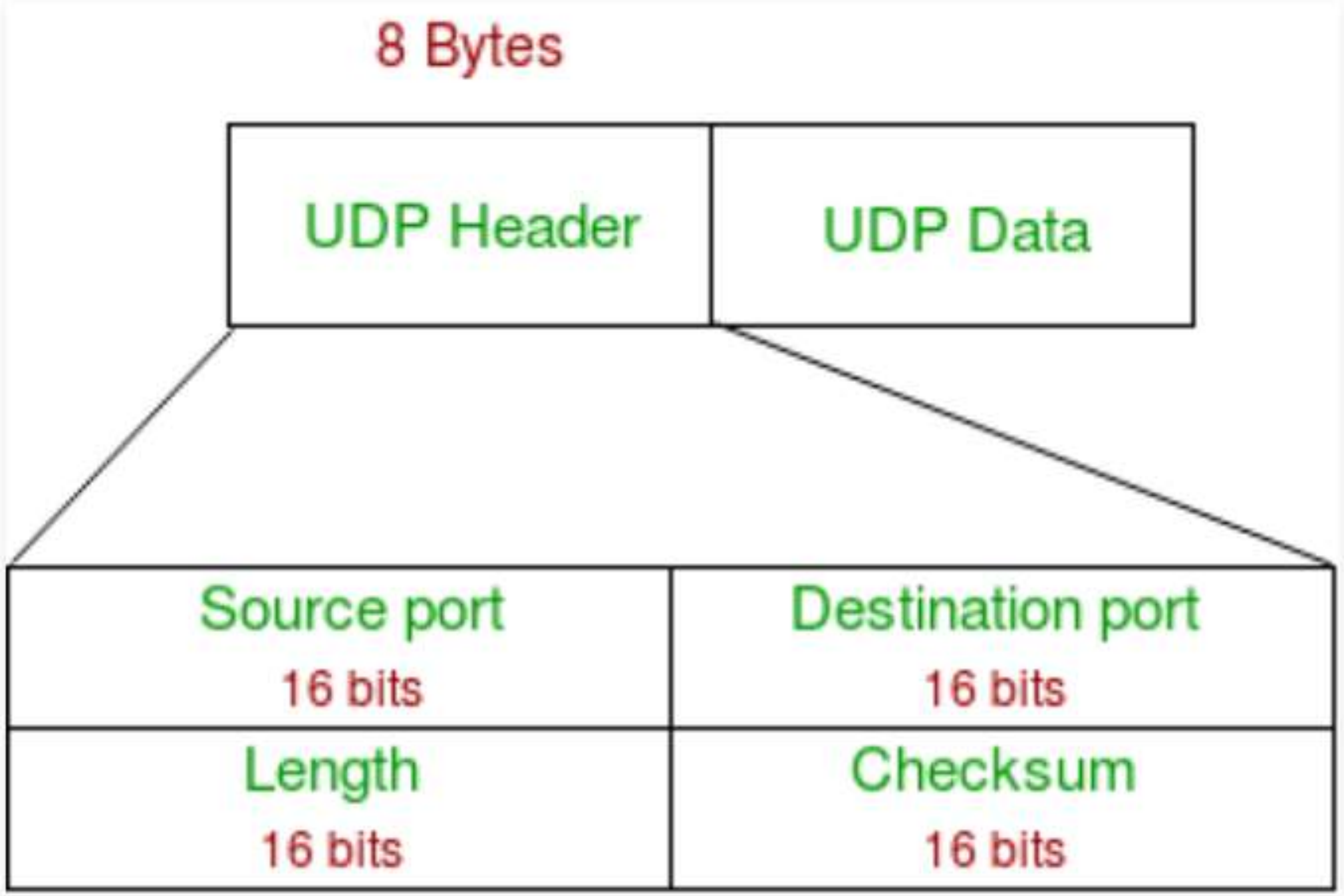
- Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services;

--- provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency.

- Here, UDP comes into the picture.
- For real-time services like computer gaming, voice or video communication, live conferences; UDP is used.

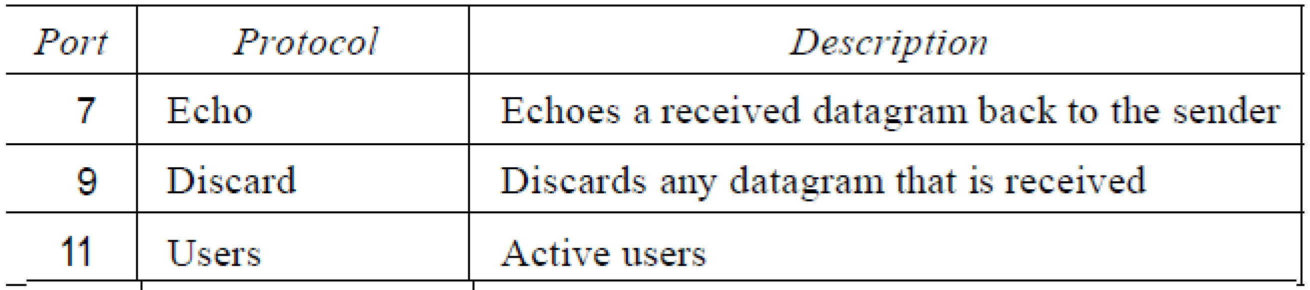
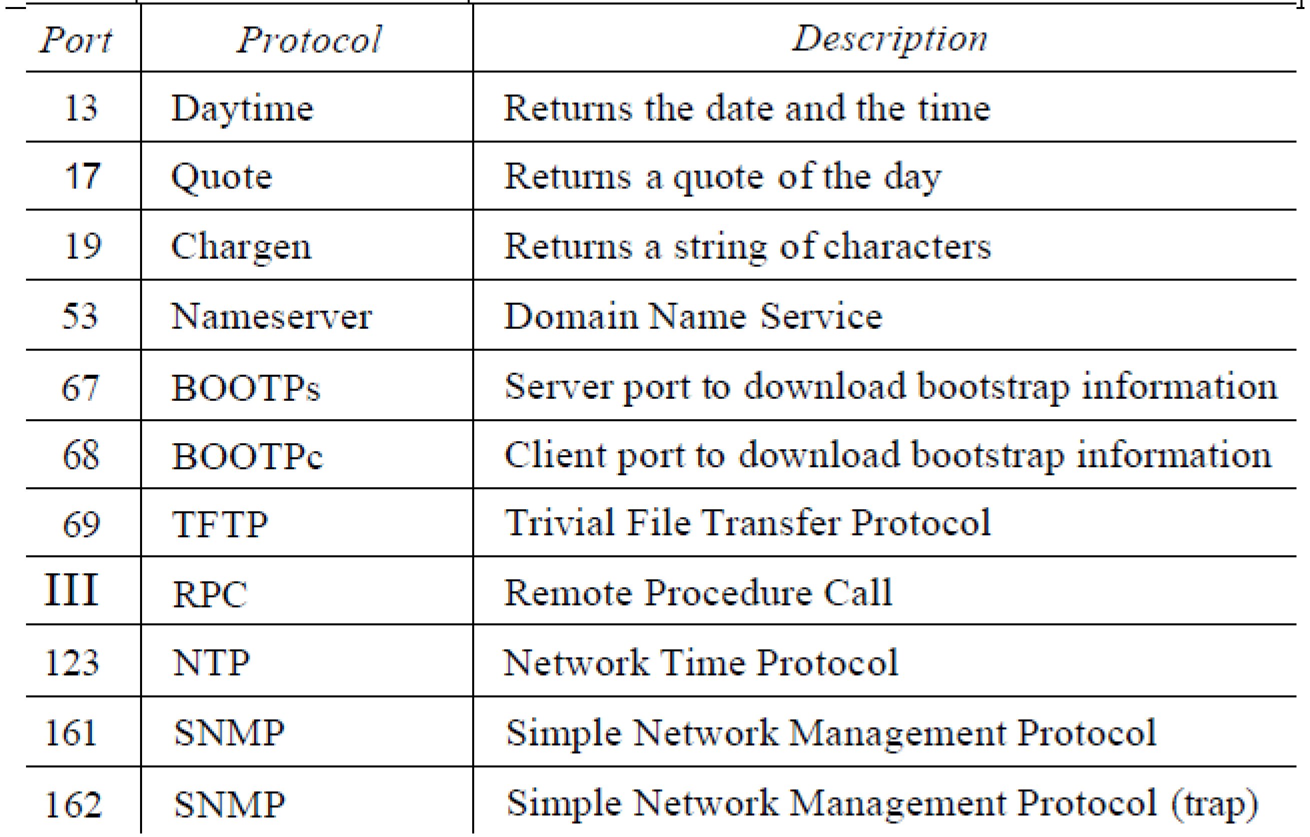
- Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets.
- There is no error checking in UDP, so it also saves bandwidth.
- User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

- 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.
- Each message from the process is called a user datagram and becomes, eventually, one IP datagram.


8 Bytes
UDP Header
UDP Data
Source port
16 bits
Destination port
16 bits
Length
16 bits
Checksum
16 bits

- UDP header is an **8-bytes** fixed and simple header.
- The first 8 Bytes contains all necessary header information and the remaining part consist of data.
- UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved.
- Port numbers help to distinguish different user requests or processes.

Well-Known Ports for 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
III	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Source Port:

- Source Port is a 2 Byte long field used to identify the port number of the source, 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

- It is a 2 Byte long field, used to identify the port of the destined packet.
- This is the port number used by the process running on the destination host.
- 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.

Length

- Length is the length of UDP including the header and the data. It is a 16-bits field.
- 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.
- However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.

- The length field in a UDP user datagram is actually not necessary.
- A user datagram is encapsulated in an IP datagram.
- There is a field in the IP datagram that defines the total length.
- There is another field in the IP datagram that defines the length of the header.

- So if we subtract the value of the second field from the first, we can deduce the length of a UDP datagram that is encapsulated in an IP datagram.

UDP length = IP length - IP header's length

Checksum

- The checksum includes three sections:

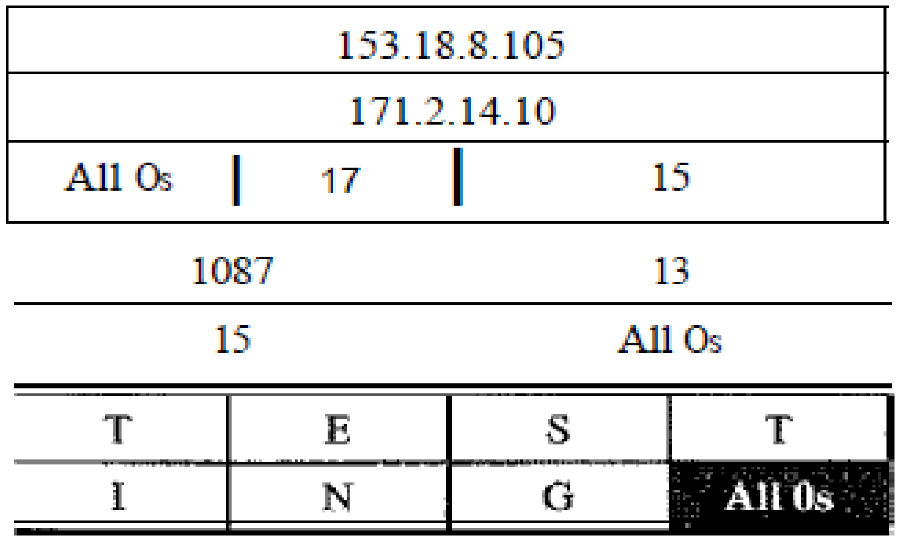
  a pseudo header, the UDP header, and the data coming from the application layer.

- Checksum is 2 Bytes long field.

- The pseudo header is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with Os.
- If the checksum does not include the pseudo header, a user datagram may arrive safe and sound.
- However, if the IP header is corrupted, it may be delivered to the wrong host.
- The protocol field is added to ensure that the packet belongs to UDP, and not to other transport-layer protocols.

- 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.
- Checksum is 2 Bytes long field.
- It is the 16-bit one’s complement of the one’s complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Checksum calculation ofa simple UDP user datagram



10011001	00010010	153.18
00001000	01101001	8.105
10101011	00000001 0	171.2
00001110	00001010	14.10
00000000	00010001	Oand 17
00000000	00001111	15
00000100	00111111	1087
00000000	00001101	13
00000000	00001111	15
00000000	00000000	0 (checksum)
01010100	01000101	T and E
01010011	01010100	Sand T
01001001	01001110	I and N
010001 II	00000000	G and 0 (padding)
1O01O110	11101011	Sum
HUHIOOl	00010100	Checksum

UDP Operation

Connectionless Services:

- UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram.
- There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.

- The user datagrams are not numbered.
- Also, there is no connection establishment and no connection termination means that each user datagram can travel on a different path.
- Only those processes sending short messages should use UDP.

Flow and Error Control

- UDP is a very simple, unreliable transport protocol.
- There is no flow control and hence no window mechanism.
- The receiver may overflow with incoming messages.

- There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated.
- When the receiver detects an error through the checksum, the user datagram is silently discarded.
- The lack of flow control and error control means that the process using UDP should provide these mechanisms.

Encapsulation and Decapsulation

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

Queuing

Client site

- In UDP, queues are associated with ports.
- At the client site, when a process starts, it requests a port number from the operating system.
- Some implementations create both an incoming and an outgoing queue associated with each process.
- Other implementations create only an incoming queue associated with each process.

- 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, in most cases, identified by ephemeral port numbers.
- The queues function as long as the process is running.
- When the process terminates, the queues are destroyed.

- 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, in most cases, 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, after adding 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.

- When a 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.

At server side

- At the server site, the mechanism of creating queues is different.
- A 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.
- All the incoming messages for one particular server, whether coming from the same or a different client, are sent to the same queue.

- If this happens, UDP drops the user datagram and asks for a port unreachable message to be sent 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.
- An outgoing queue can overflow.
- If this happens, the operating system asks the server to wait before sending

  any more messages.

Applications of UDP:

•Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.

•It is a suitable protocol for multicasting as UDP supports packet switching.

•UDP is used for some routing update protocols like RIP(Routing Information Protocol).

•Normally used for real-time applications which can not tolerate uneven delays between sections of a received message.

•Following implementations uses UDP as a transport layer protocol:

NTP (Network Time Protocol)

DNS (Domain Name Service)

BOOTP, DHCP.

NNP (Network News Protocol)

Quote of the day protocol

- UDP takes a datagram from Network Layer, attaches its header, and sends it to the user. So, it works fast.
- Actually, UDP is a null protocol if you remove the checksum field.
    1. Reduce the requirement of computer resources.
    2. When using the Multicast or Broadcast to transfer.
    3. The transmission of Real-time packets, mainly in multimedia applications.

- The transport layer protocols used for real time multimedia, file transfer, DNS and email, respectively are:
  (A) TCP, UDP, UDP and TCP
  (B) UDP, TCP, TCP and UDP
  (C) UDP, TCP, UDP and TCP
  (D) TCP, UDP, TCP and UDP

- TCP is connection oriented and UDP is connectionless, this makes TCP more reliable than UDP. But UDP is stateless (less overhead), that makes UDP is suitable for purposes where error checking and correction is less important than timely delivery.
- For real time multimedia, timely delivery is more important than correctness. –> UDP
- For file transfer, correctness is necessary. –> TCP
- DNS, timely delivery is more important –> UDP
- Email again same as file transfer –> TCP

Answer: (C)

- Which of the following transport layer protocols is used to support electronic mail?
  (A) SMTP
  (B) IP
  (C) TCP
  (D) UDP

  Answer: (C)

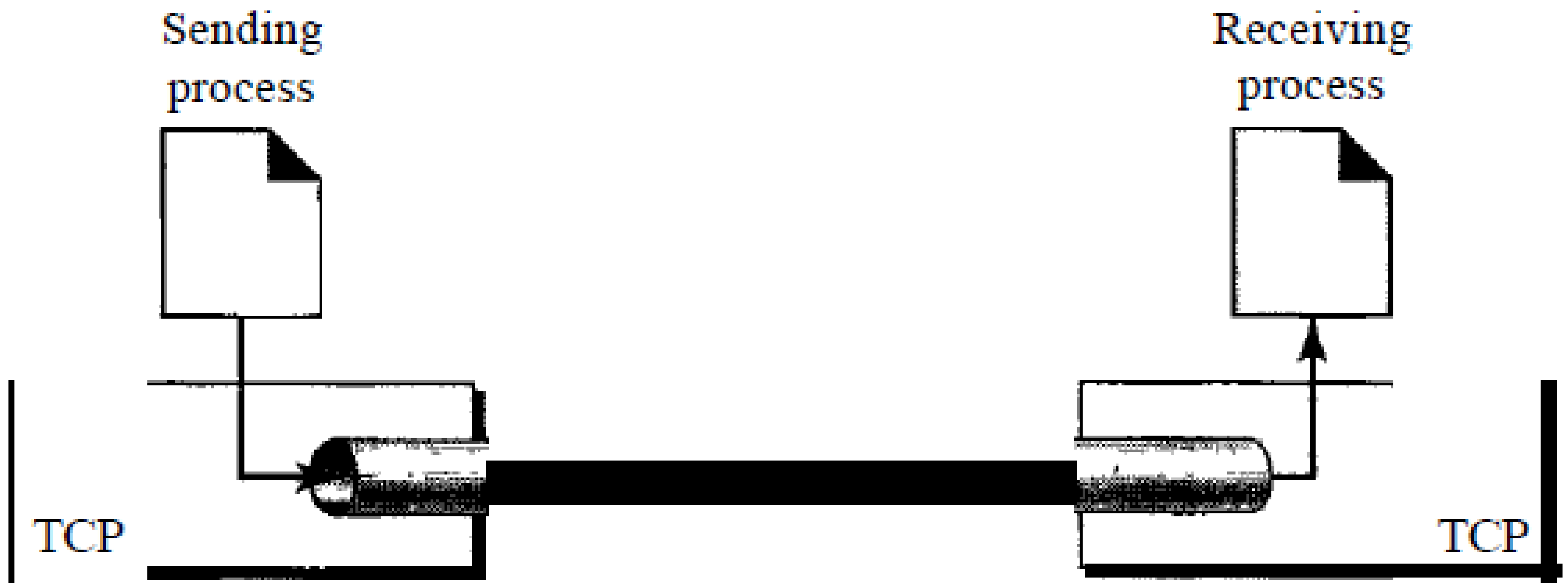
Transmission Control Protocol (TCP)

- TCP, like UDP, is a process-to-process (program-to-program) protocol.
- TCP, therefore, like UDP, uses port numbers.
- TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data.

- In addition, TCP uses flow and error control mechanisms at the transport level.
- In brief, TCP is called a *connection-oriented, reliable* transport protocol.
- It adds connection-oriented and reliability features to the services of IP.

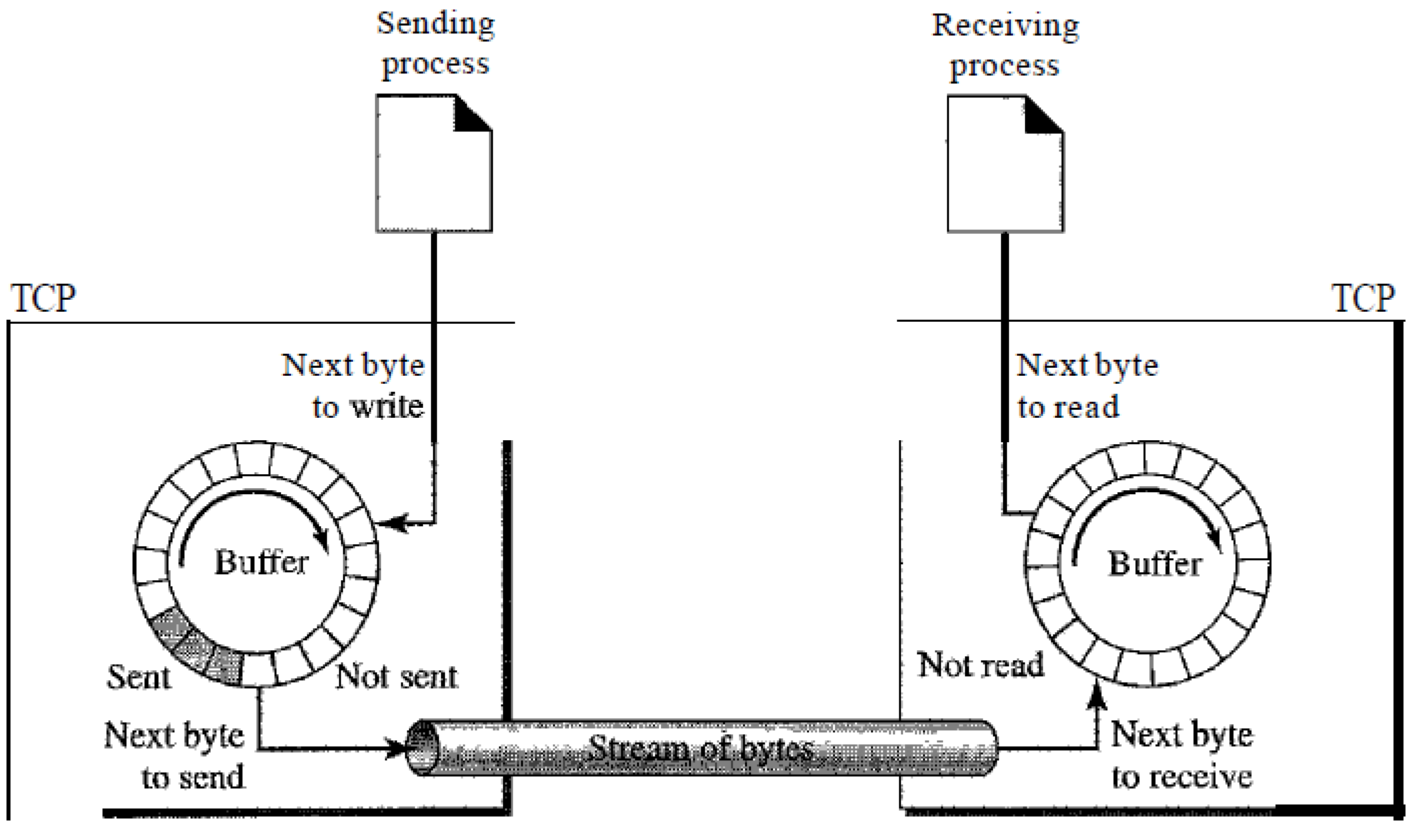
- TCP, unlike UDP, is a stream-oriented protocol.
- TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
- TCP creates an environment in which the two processes seem to be connected by an imaginary "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.

Stream delivery



- 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.
- normally the buffers are hundreds or thousands of bytes, depending on the implementation.

Sending and receiving buffers



At the sending site,

- the buffer has three types of chambers.

-- empty chambers that can be filled by the sending process (producer).

-- The area holds bytes that have been sent but not yet acknowledged.

- TCP keeps these bytes in the buffer until it receives an acknowledgment.

-- bytes to be sent by the sending TCP.

- TCP may be able to send only part of the bytes due to the slowness of the receiving process or perhaps to congestion in the network.
- after the bytes sent are acknowledged, the chambers are recycled and available for use by the sending process.

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

- 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.
- TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission.
- The segments are encapsulated in IP datagrams and transmitted.
- Note that the segments are not necessarily the same size.

Full-Duplex Communication

- TCP offers full-duplex service, in which data can flow in both directions at the same time.

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.

Note that this is a virtual connection, not a physical connection.

- The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost, or corrupted, and then resent.
- Each may use a different path to reach the destination

Reliable Service

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

TCP Features

Numbering System

- 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.

- When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them.
- The numbering does not necessarily start from 0.
- Instead, TCP generates a random number between 0 and 232 - 1 for the number of the first byte.
- For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056.

Sequence Number

- After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.

Acknowledgment Number

- communication in TCP is full duplex;
- when a connection is established, both parties can send and receive data at the same time.
- The sequence number in each direction shows the number of the first byte carried by the segment

- Each party also uses an acknowledgment number to confirm the bytes it has received.
- However, the acknowledgment number defines the number of the next byte that the party expects to receive.
- The party takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number.

- if a party uses 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642.
- Note that this does not mean that the party has received 5642 bytes because the first byte number does not have to start from O.

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.
- The numbering system allows TCP to use a byte-oriented flow control.

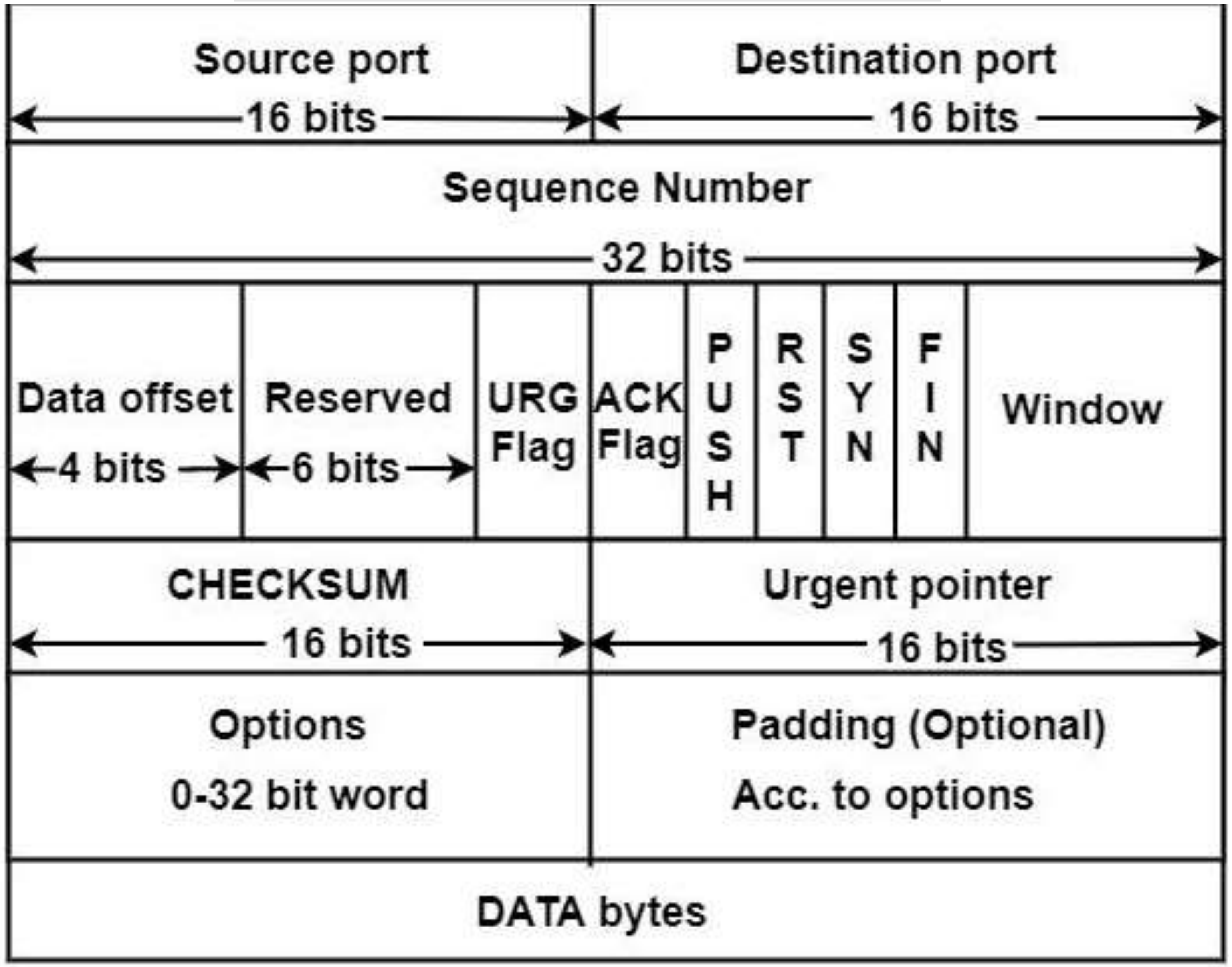
Error Control

- 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.

Congestion Control

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

TCP Segment Header



- Every TCP segment consists of a 20 byte fixed format header.
- Header options may follow the fixed header.
- With a header so that it can tag up to 65535 data bytes.

Source Port

- It is a 16-bit source port number used by the receiver to reply.

Destination Port

- It is a 16-bit destination port number.

Sequence Number

- The sequence number of the first data byte in this segment.
- During the SYN Control bit is set, and the sequence number is n, and the first data byte is n + 1.

Acknowledgement Number

- If the ACK control bit is set, this field contains the next number that the receiver expects to receive.

Data Offset

- The several 32-bit words in the TCP header shows from where the user data begins.

Reserved (6 bit)

- It is reserved for future use.

Control.

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

URG

ACK

PSH

RST

SYN

FIN

URG

- It indicates an urgent pointer field that data type is urgent or not.

ACK

- It indicates that the acknowledgement field in a segment is significant.

PUSH

- The PUSH flag is set or reset according to a data type that is sent immediately or not.
- **RST**

  It Resets the connection.

SYN

- It synchronizes the sequence number.

FIN

- This indicates no more data from the sender.

Window

- It is used in Acknowledgement segment.
- It specifies the number of data bytes, beginning with the one indicated in the acknowledgement number field that the receiver is ready to accept.

Checksum

- It is used for error detection.

Options

- The IP datagram options provide additional punctuality. It can use several optional parameters between a TCP sender and receiver. It depends on the options used. The length of the field may vary in size, but it can't be larger than 40 bytes due to the header field's size, which is 4 bit.

Padding

- Options in each may vary in size, and it may be necessary to "pad" the TCP header with zeros so that the segment ends on a 32-bit word boundary as per the standard.

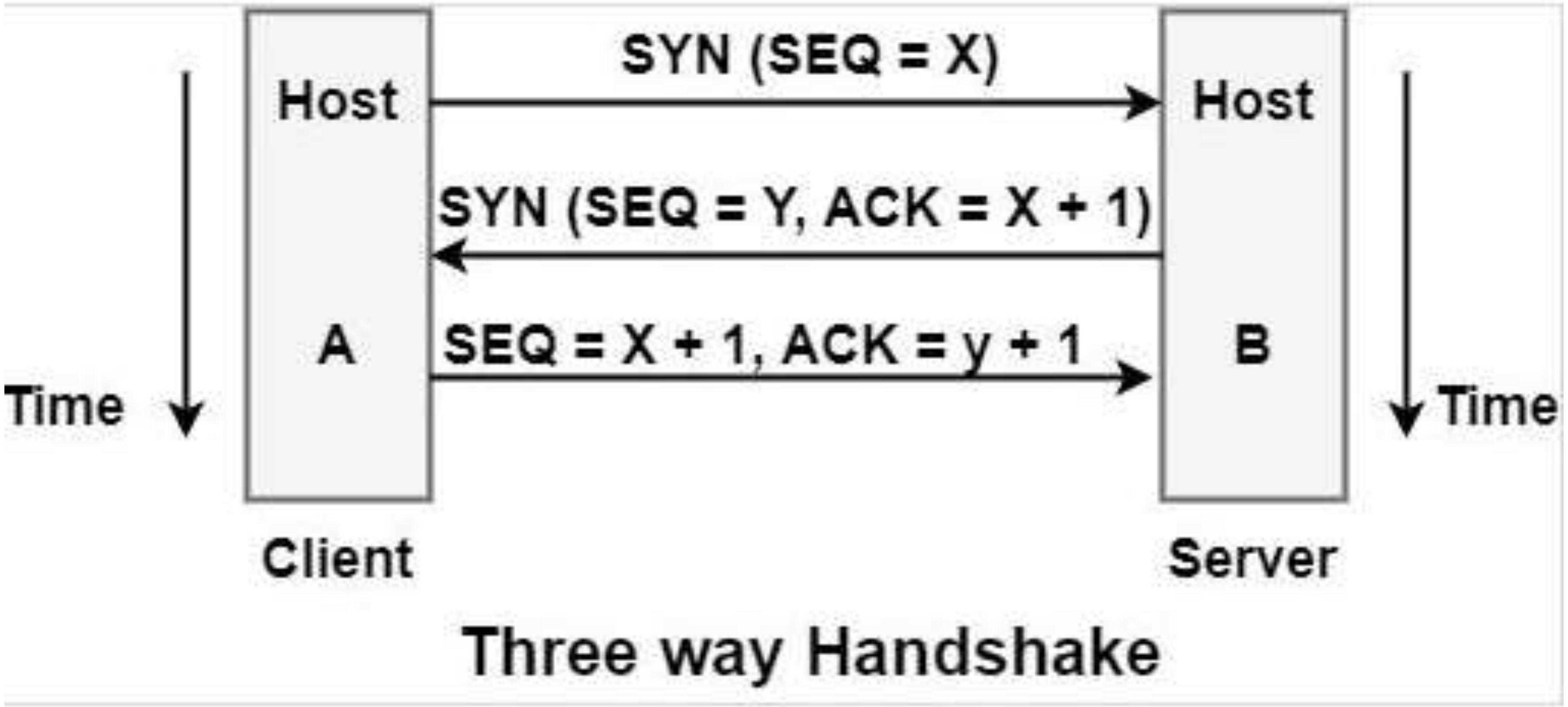
Data

- Although in some cases like acknowledgement segments with no data in the reverse direction, the variable-length field carries the application data from sender to receiver. This field, connected with the TCP header fields, constitute a TCP segment.

Connection Managment

- To make the transport services reliable, TCP hosts must establish a connection-oriented session with one another.
- Connection establishment is performed by using the three-way handshake mechanism.
- A three-way handshake synchronizes both ends of a network by enabling both sides to agree upon original sequence numbers.

- This mechanism also provides that both sides are ready to transmit data and learn that the other side is available to communicate.
- This is essential so that packets are not shared or retransmitted during session establishment or after session termination.
- Each host randomly selects a sequence number used to track bytes within the stream it is sending and receiving.

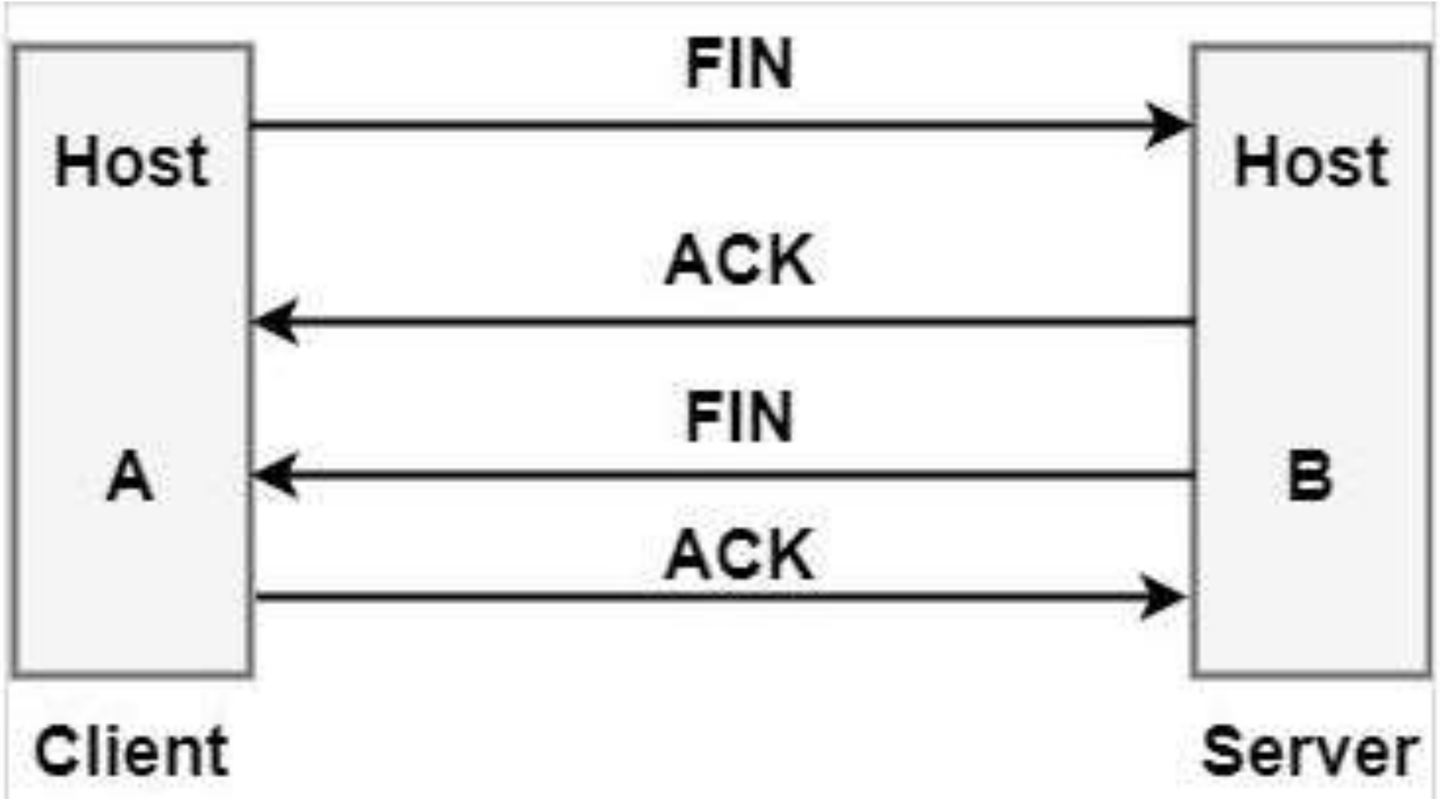
SYN (SEQ = X)
Host
SYN (SEQ = Y, ACK = X + 1)
Host
A
SEQ = X + 1, ACK = y + 1
B
Time
Time
Client
Server
Three way Handshake

- The requesting end (Host A) sends an SYN segment determining the server's port number that the client needs to connect to and its initial sequence number (x).
- The server (Host B) acknowledges its own SYN segment, including the servers initial sequence number (y).
- The server also responds to the client SYN by accepting the sender's SYN plus one (X + 1).

- An SYN consumes one sequence number. The client should acknowledge this SYN from the server by accepting the server's SEQ plus one (SEQ = x + 1, ACK = y + 1). This is how a TCP connection is settled.

Connection Termination Protocol (Connection Release)

- While it creates three segments to establish a connection, it takes four segments to terminate a connection.
- During a TCP connection, is full-duplex (that is, data flows in each direction independently of the other direction), each direction should be shut down alone.
- The termination procedure for each host is shown in the figure.
- The rule is that either end can share a FIN when it has finished sending data.



TCP Termination

- The receipt of a FIN only means that there will be no more data flowing in that direction.
- A TCP can send data after receiving a FIN.
- The end that first issues the close (example, send the first FIN) executes the active close.
- The other end (that receives this FIN) manages the passive close.

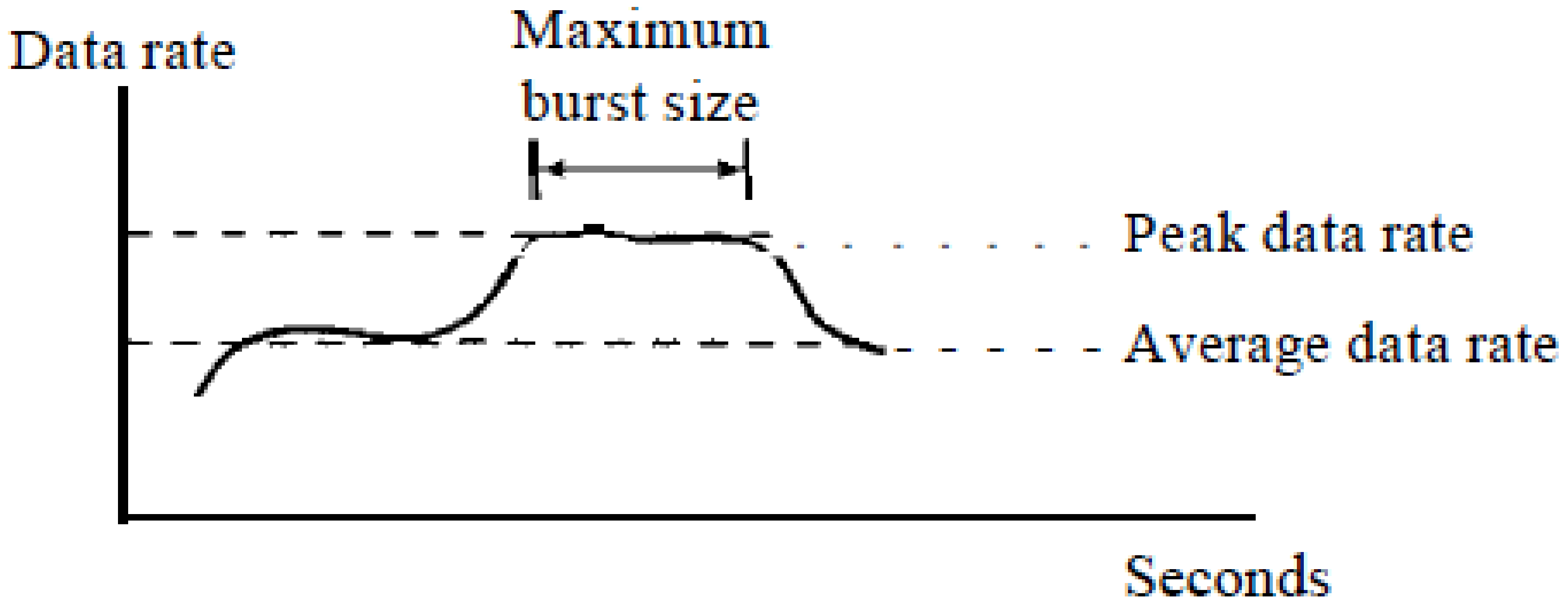
- https://www.tutorialspoint.com/what-is-the-tcp-connection-establishment

DATA TRAFFIC

- Depends on two factors
  1. Congestion and
  2. Quality of Services.

- 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



Average Data Rate

- The average data rate is the number of bits sent during a period of time, divided by the number of seconds in that period.
- We use the following equation:

$$\text{Average data rate} = \frac{\text{amount of data}}{\text{time}}$$

- The average data rate is a very useful characteristic of traffic because it indicates the average bandwidth needed by the traffic.

Peak Data Rate

- The peak data rate defines the maximum data rate of the traffic.
- The peak data rate is a very important measurement because it indicates the peak bandwidth that the network needs for traffic to pass through without changing its data flow.

Maximum Burst Size

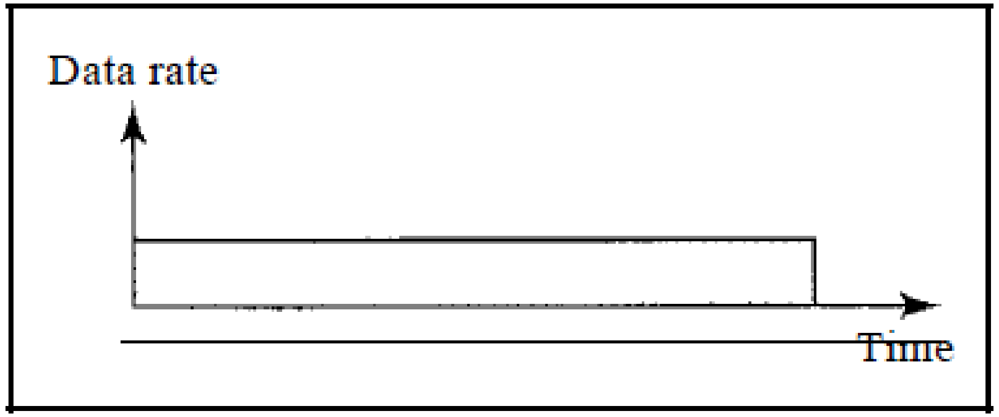
- Although the peak data rate is a critical value for the network, it can usually be ignored if the duration of the peak value is very short.
- The maximum burst size normally refers to the maximum length of time the traffic is generated at the peak rate.

Effective Bandwidth

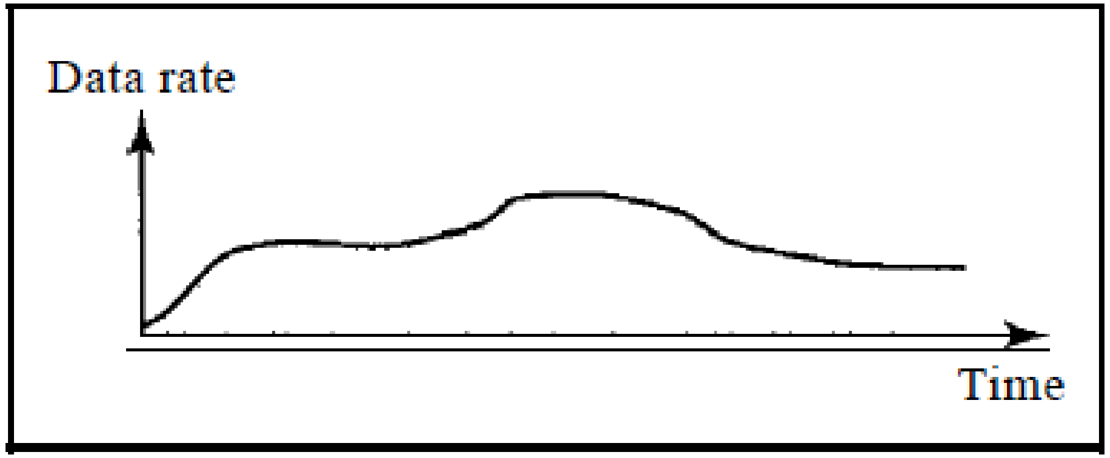
- The effective bandwidth is the bandwidth that the network needs to allocate for the flow of traffic.
- The effective bandwidth is a function of three values: average data rate, peak data rate, and maximum burst size.
- The calculation of this value is very complex.

Traffic Profiles

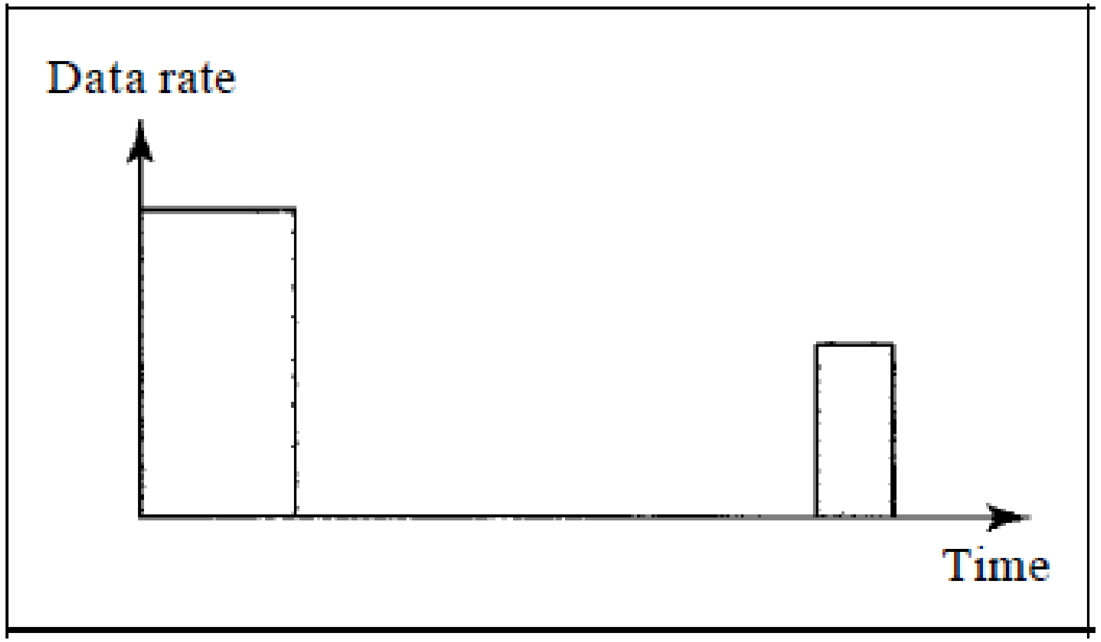
- constant bit rate,
- variable bit rate,
- bursty



a. Constant bit rate



b. Variable bit rate



c. Bursty

CONGESTION

- Congestion causes choking of the communication channel.
- When too many packets are displayed in a part of the subnet, the subnet's performance degrades.
- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.

- It is known as heavily congested when the packets never reach the destination, denoting the delay method infinity.
- When the input traffic rate exceeds the output lines capacity, the subnet's input part gets choked and generates congestion.
- It also happens when the routers are too slow to execute queuing buffers, refreshing tables, etc.

- The loss of capacity of the routers' buffer is also one of the many factors for congestion.
- However, enhancing the memory of the router may be helpful up to a certain point.

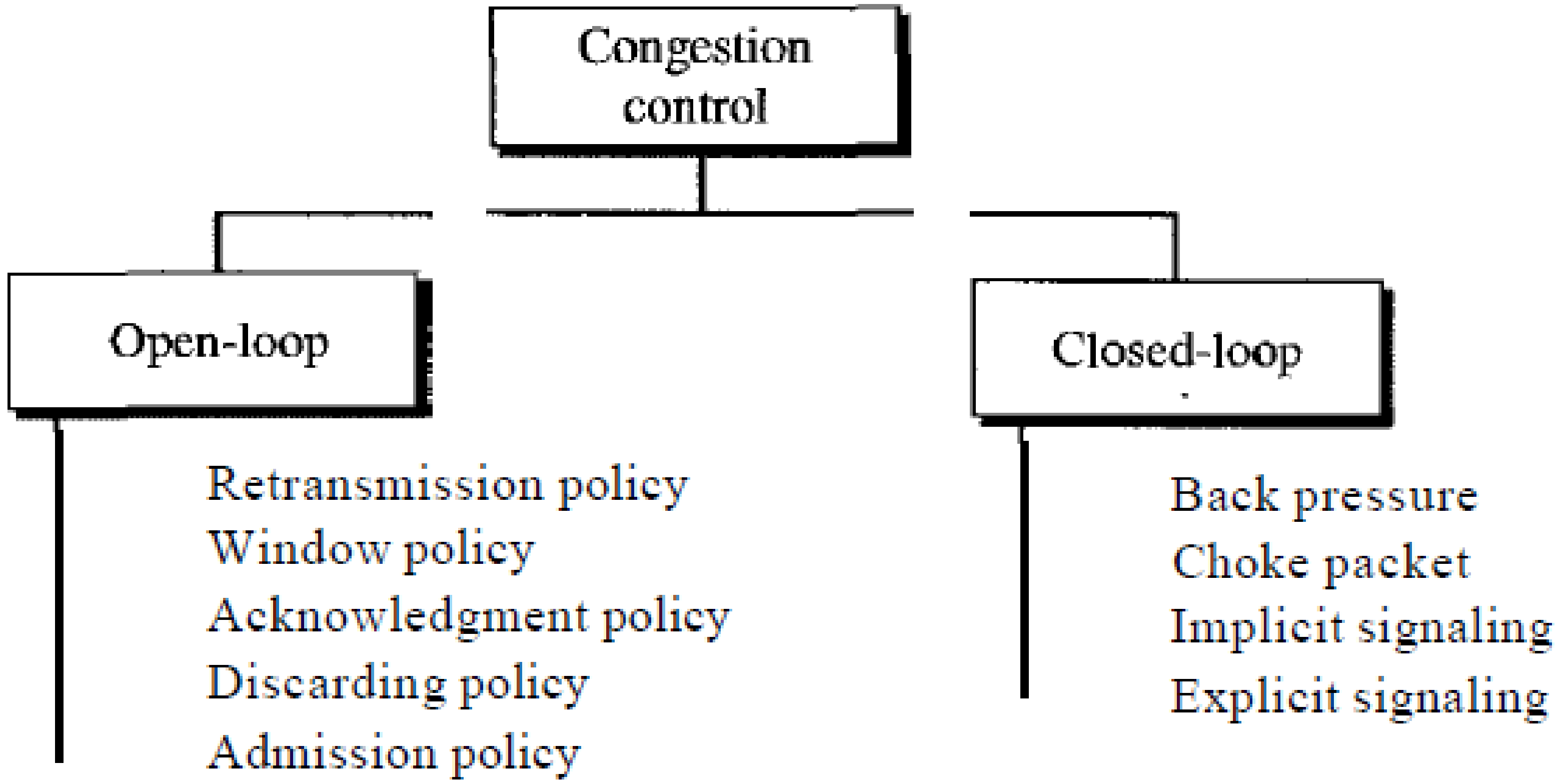
Principles of Congestion Control

The open-loop solutions :

- It provides an excellent design to ensure that the problem does not occur in the first place.
- The designing tools include deciding to accept new traffic, discarding Packets and scheduling the packets at various network points.
- Policies are applied to prevent congestion before it happens.
- In these mechanisms, congestion control is handled by either the source or the destination.

Closed-loop solutions

- It makes the decision based on the concept of a feedback loop.
- The feedback Notes loop enables the closed-loop system to monitor the procedure to detect when and where congestion occurs.
- After that, it passes the information to the places where they can take actions.


Congestion control
Open-loop
Closed-loop
Retransmission policy
Window policy
Acknowledgment policy
Discarding policy
Admission policy
Back pressure
Choke packet
Implicit signaling
Explicit signaling

Retransmission Policy

- 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

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.
- In the *Go-Back-N* window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver.
- This duplication may make the congestion worse.
- The Selective Repeat window, on the other hand, tries to send the

specific packets that have been lost or corrupted.

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.
- The acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.

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.

Ex Audio packets.

Admission Policy

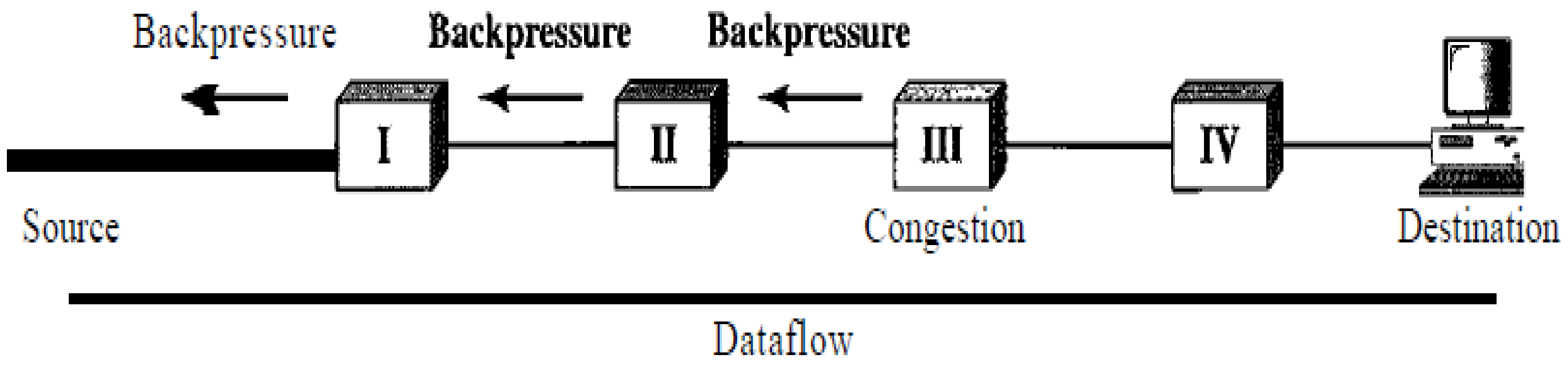
- A router can deny establishing a virtual circuit connection if there is congestion in the network or if there is a possibility of future congestion.
- An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks.

Closed-Loop Congestion Control

- Closed-loop congestion control mechanisms try to alleviate congestion after it happens.

  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.
- And so on.

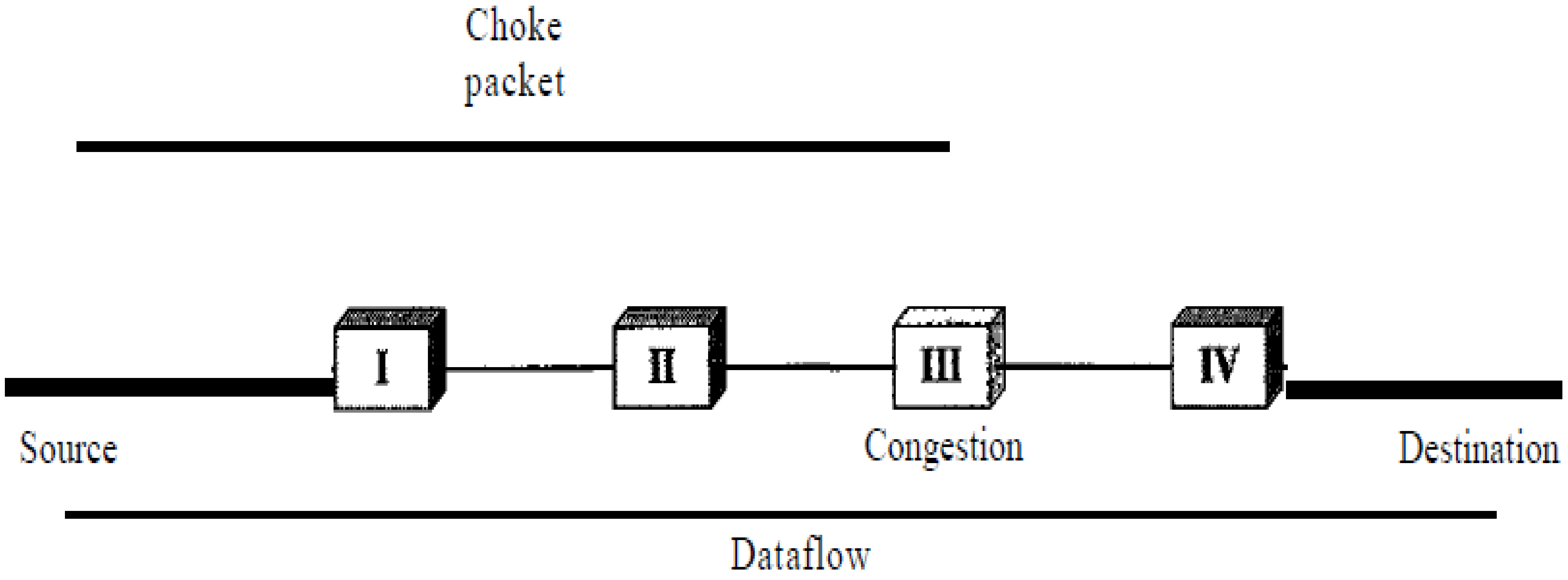


The *pressure* on node III is moved backward to the source to remove the congestion.

Choke Packet

A choke packet is a packet sent by a node to the source to inform it of congestion.

- In backpressure,the warning is from one node to its upstream node, although the warning may eventually reach the source station.
- In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly


Choke packet
I
II
III
IV
Source
Congestion
Destination
Dataflow

Implicit Signalling

- 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.
- For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested.
- The delay in receiving an acknowledgment is interpreted as congestion in the network.

Explicit Signalling

The node that experiences congestion can explicitly send a signal to the source or destination.

- 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.

Backward Signaling

- A bit can be set in a packet moving in the direction opposite to the congestion.
- This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

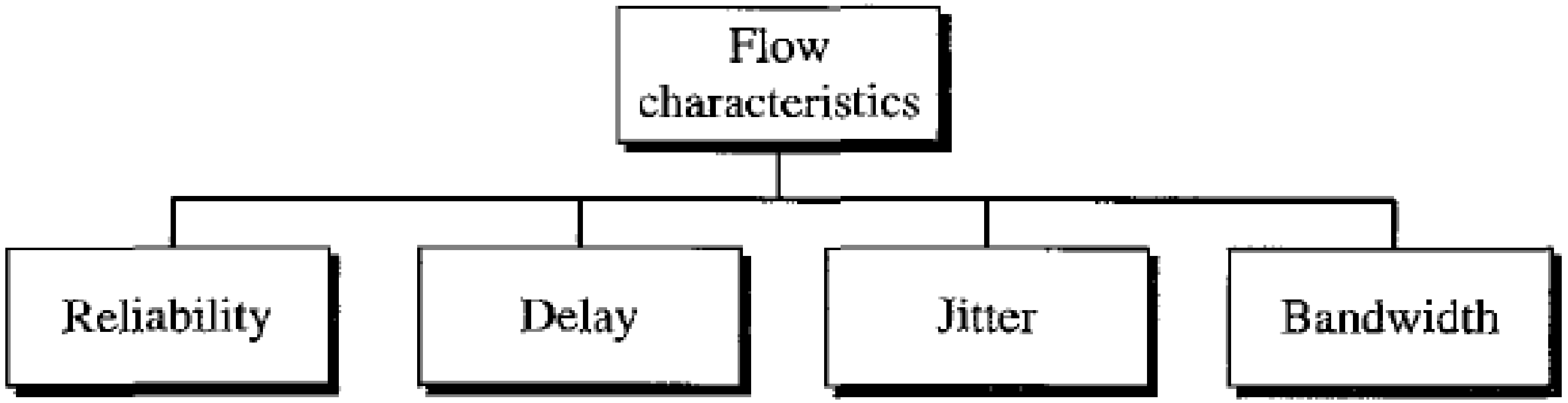
Forward Signaling

- A bit can be set in a packet moving in the direction of the congestion.
- This bit can warn the destination that there is congestion.
- The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion

QUALITY OF SERVICE

- The quality of service as something a flow seeks to attain.

Flow Characteristics



Reliability

- Reliability is a characteristic that a flow needs.
- Lack of reliability means losing a packet or acknowledgment, which entails retransmission.
- For example, it is more important that electronic mail, file transfer, and Internet access have reliable transmissions than telephony or audio conferencing.

Delay

- Source-to-destination delay is another flow characteristic.
- Again applications can tolerate delay in different degrees.
- In this case, telephony, audio conferencing, video conferencing, and remote log-in need minimum delay, while delay in file transfer or e-mail is less important.

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.
- On the other hand, if the above four packets arrive at 21, 23, 21, and 28, they will have different delays: 21,22, 19, and 24.

- For applications such as audio and video, the first case is completely acceptable; the second case is not.
- For these applications, it does not matter if the packets arrive with a short or long delay as long as the delay is the same for all packets.
- For this application, the second case is not acceptable.
- Jitter is defined as the variation in the packet delay. High jitter means the difference between delays is large; low jitter means the variation is small.

Bandwidth

- Different applications need different bandwidths.
- 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.

Flow Classes

- Based on the flow characteristics, we can classify flows into groups, with each group having similar levels of characteristics.
- This categorization is not formal or universal

Techniques TO IMPROVE QoS

1. Scheduling
2. Traffic shaping,
3. Admission control, and
4. Resource reservation.

Scheduling:

- Packets from different flows arrive at a switch or router for processing.
- A good scheduling technique treats the different flows in a fair and appropriate manner.

Types

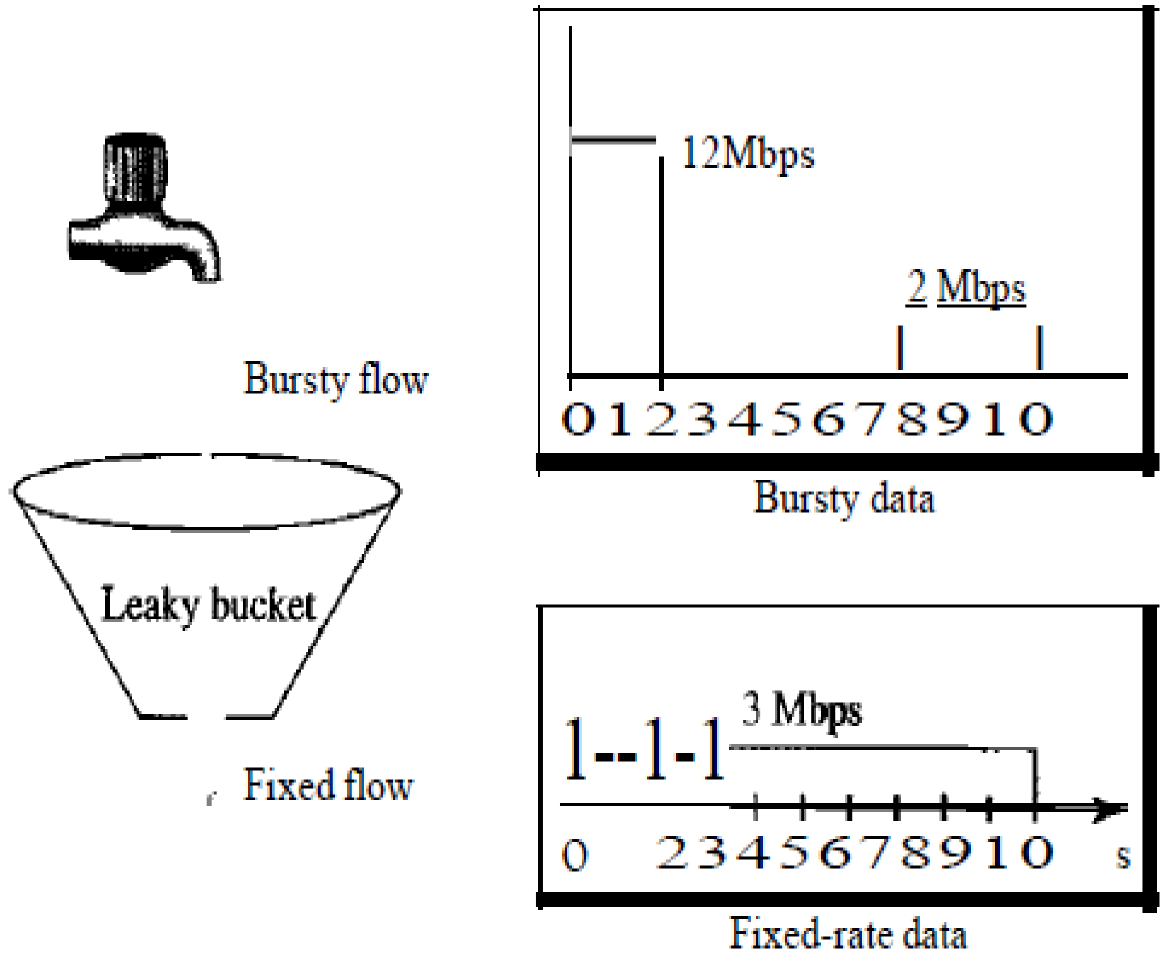
FIFO queuing, priority queuing, and weighted fair queuing.

Traffic Shaping

- Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network.
- Two techniques can shape traffic:

1. Leaky bucket and
2. Token bucket.

Leaky Bucket



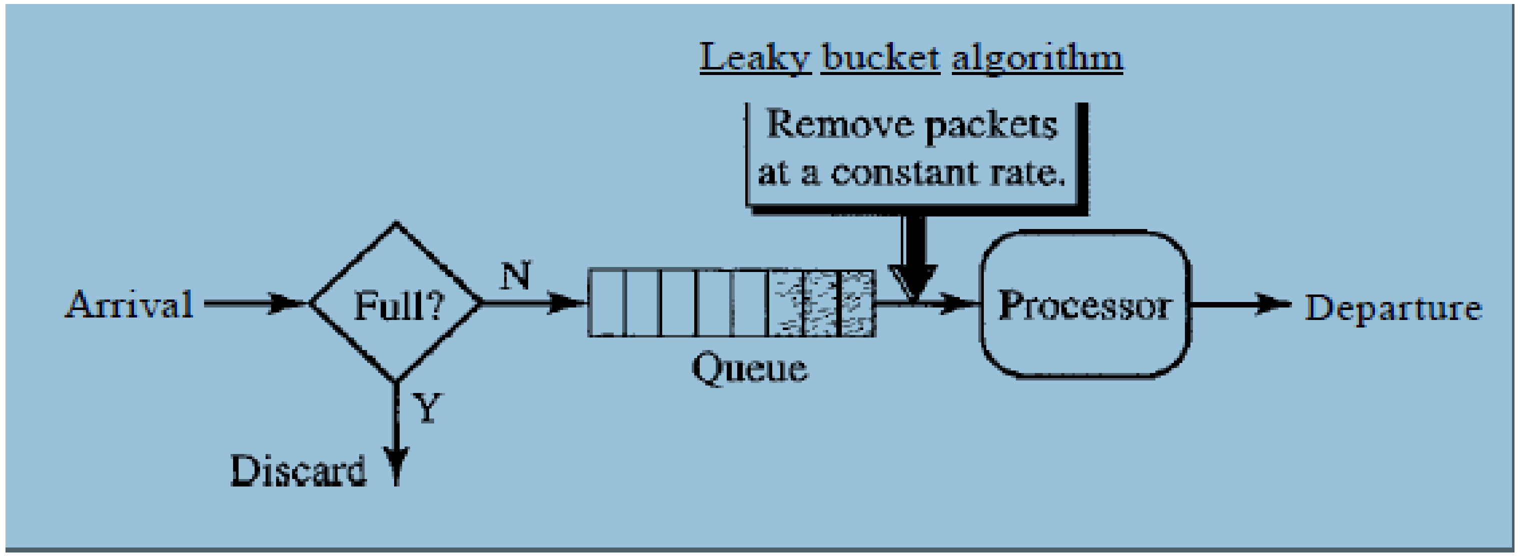
- If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rateas long as there is water in the bucket.
- The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty
- The input rate can vary, but 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.

- Assume that the network has committed a bandwidth of 3 Mbps for a host.
- The use of the leaky bucket shapes the input traffic to make it conform to this commitment.
- Let the host sends a burst of data at a rate of 12 Mbps for 2 s, a total of 24 M bits of data.
- The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 M bits of data.
- In all, the host has sent 30 M bits of data in 10 s.
- The leaky bucket smooth the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

- A simple leaky bucket implementation is shown in Figure.
- A FIFO queue holds the packets.
- If the traffic consists of fixed-size packets the process removes a fixed number of packets from the queue at each tick of the clock.
- If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

Leaky bucket implementation



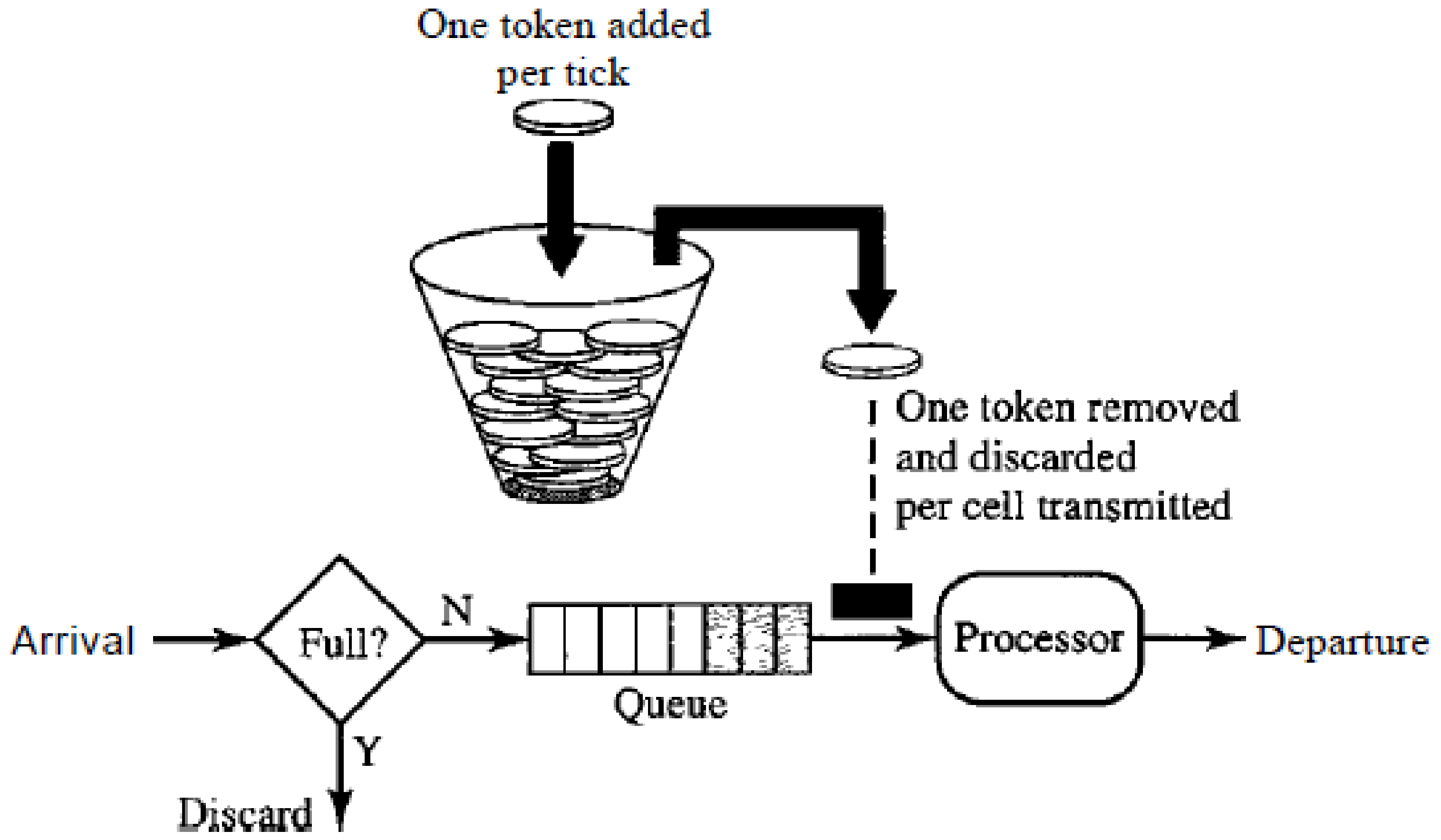
The following is an algorithm for variable-length packets:

1. Initialize a counter to *n* at the tick of the clock.
2. If *n* is greater than the size of the packet, send the packet and decrement the
counter by the packet size. Repeat this step until *n* is smaller than the packet size.
3. Reset the counter and go to step 1.

- 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.

Token Bucket

- The leaky bucket is very restrictive.
- It does not credit an idle host.
- If a host is not sending for a while, its bucket becomes empty.
- 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.


One token added per tick
One token removed and discarded per cell transmitted
Arrival
Full?
N
Y
Discard
Queue
Processor
Departure

- 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 algorithm can be conceptually understood as follows:
- A token is added to the bucket every 1/r seconds.
- The bucket can hold at the most b tokens.
- If a token arrives when the bucket is full, it is discarded.

- 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.

- When a packet of n bytes arrives,

-- if at least n tokens are in the bucket, n tokens are removed from the bucket and the packet is sent to the network.

-- if fewer than n tokens are available., no tokens are removed and the packet is considered to the non- conformant.

Admission Control

- Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications.
- Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

- It is a validation process in communication systems where a check is performed before a connection is established to see if current resources are sufficient or not.
- Routers or switches put restrictions on the admission of packets from host.
- Before the router accepts the flow it checks the flow for specifications in terms of bandwidth, buffer size, cpu speed etc..

Resource Reservation

- A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on.
- The quality of service is improved if these resources are reserved beforehand.

QoS model called Integrated Services, which depends heavily on resource reservation to improve the quality of service.