TRANSPORT LAYER UNIT-5 01/04/2020

Topics To be Covered

1) Transport layer Services
2) Elements of Transport layer Protocols
3) UDP (User Datagram packet)
4) TCP (Transmission Control Protocol)
5) Quality of Service
6) Leaky Bucket
7) Token Bucket Algorithm

Process-To Process Delivery :-

* The Data link layer is Responsible for delivery of frames b/w two neighboring nodes over a link. This is called Node to Node delivery

* The network layer is Responsible for delivery of Datagram b/w two hosts. This is called Host to Host delivery.

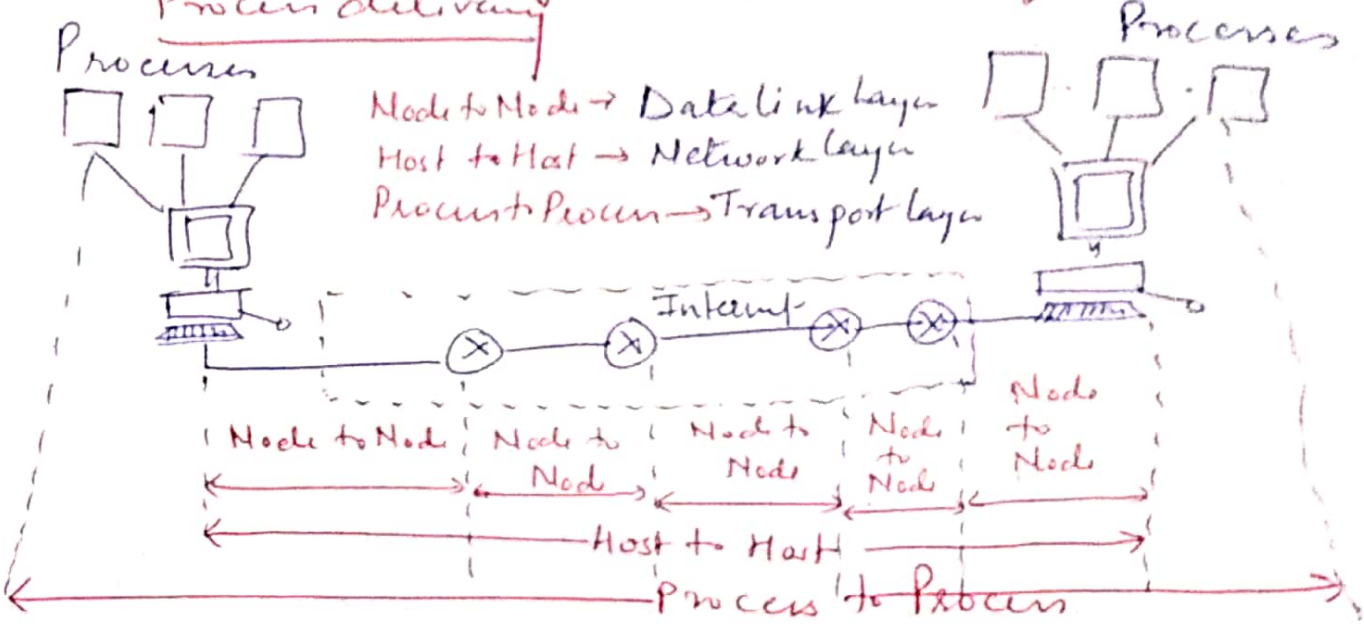
Communication on the Internet is not defined as the Exchange of Data b/w two nodes or b/w two hosts. Real Com^n takes place b/w two Process (Application program).

* Several process may be running on the source host and several on the destination host. To complete the delivery we need a Mechanism to deliver data from one of these process running on the source host to the corresponding process running on the Destination host.

* The Transport layer is Responsible for Process to Process delivery - The delivery of a packet, part of a message, from one process to another. Two process communicate in client/Server Relationship

Shows three types of deliveries and their Domain

The transport layer is Responsible for Process to Process delivery



Types of Data deliveries

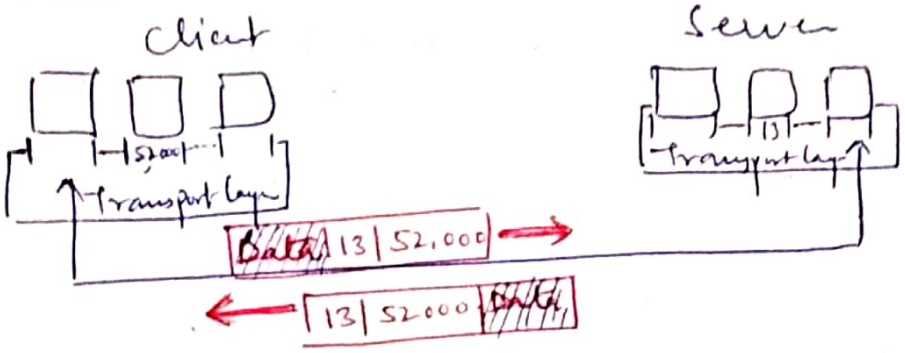
Addressing -

1) At data link layer we need a MAC address / Physical address to choose one node among several nodes if the connection is not point to point. A frame in the DLL needs a destination MAC address for delivery and a source address for the next node Reply.

2) At Network layer, we need an IP address to choose one host among millions. A datagram in the NW layer needs a destination IP address for delivery and a source IP address for the Destination's Reply.

3) At transport layer we need a Transport Layer address called Port address to choose among Multiple processes running on the destination host. The Destination port number is needed for delivery; the source port number is needed for the Reply.

Port Number



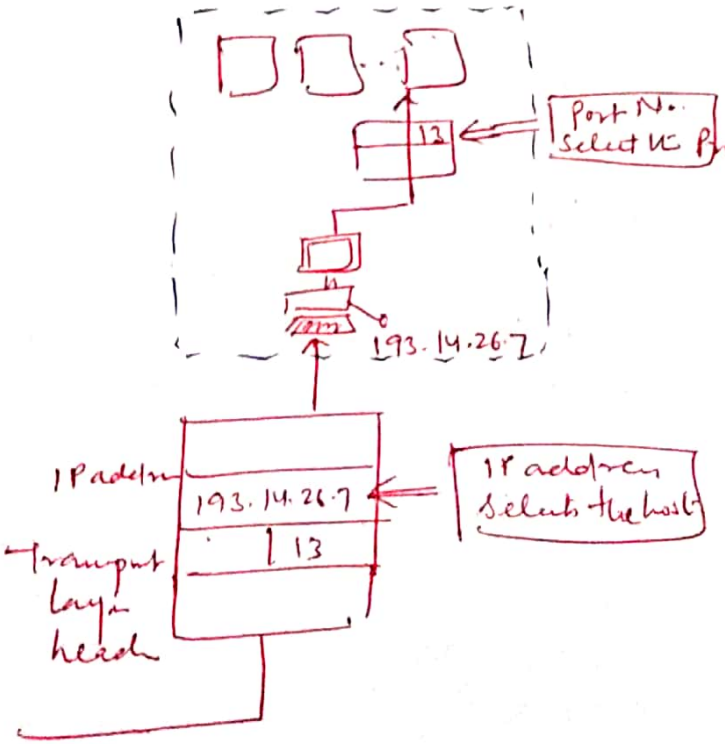
In Unix model, the port number are 16 bit Integer between 0 and 65535. The client program defines itself with a Port Number, choosen randomly by the Transport layer S/W running on the client host. This is Ephemeral port number.

The unix has decided to use universal Port numbers for servers, these are called well known port Numbers.

Every client process knows the well known port No. of the corresponding server process.

client process → ephemeral (Temporary) port No → 52000 → Volatility
server process → well known (permanent) port No → 13.

IP address Versus port Number

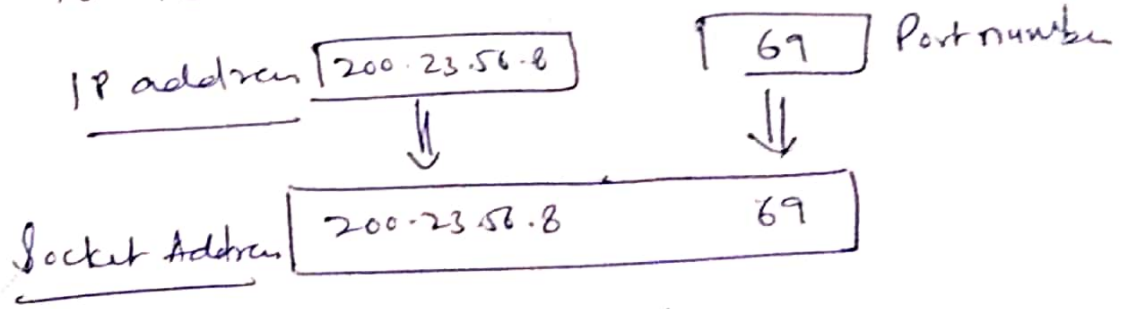


Socket Address - To process delivery needs two identifiers, IP address and port Number at each end to make a connection. The combination of IP address and a port Number is called **Socket address**

- The client Socket address defines the client process uniquely. Just as server socket address defines the server process uniquely.

- The Transport layer protocol needs a pair of Socket address: the client address
  the Server socket Address.

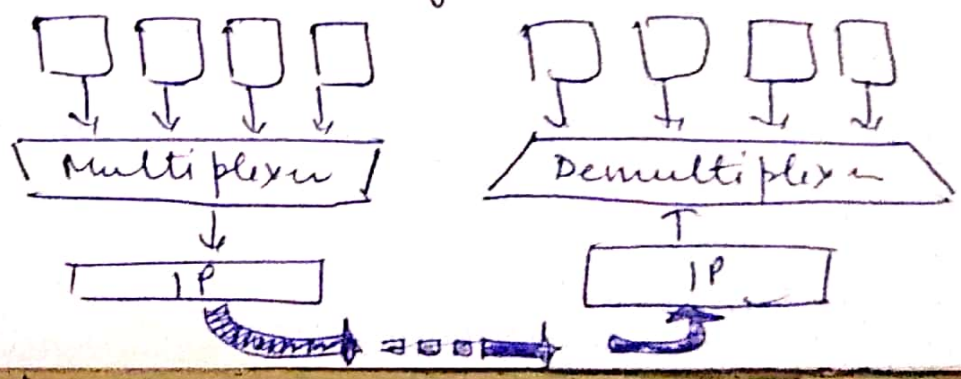
- These four pieces of information are part of IP header and TLP header. IP header contain the IP address, the UDP & TCP header contain the Port Number.



Socket Address

Multiplexing and Demultiplexing :-

- The addressing mechanism allows Multiplexing and Demultiplexing by IP & TL.



Multiplexing :- At the sender site, there may be several process that need to send packet. There is only one TLP at any time. This is a many to one relationship and require Multiplexing. The protocol accepts messages from different process differentiated by their assigned port Number. After adding header, the TL passes the packet to the NL.

Demultiplexing :- At Receiver site, the Relationship is one to many and required Demultiplexing. The TL receive datagram from the NLW layer. After error checking and dropping of the header, the transport layer deliver each message to the appropriate process based on the port Number.

Connection Vs Connection Oriented Service :-

A TLP can either be Connectionless or Connection Oriented.

Connection less Services : In Connectionless Service the packet are sent from one party to another with no need for Connection establishment or Connection release. The packet are not numbered they may delayed, or lost or may arrive out of sequence. There is No. Acknowledgment, we will see in Next lecture one of TLP is the Packet marked. **UDP is connectionless**

Connection Oriented Service :- In Connection Oriented Service, a connection is first established between Sender and the Receiver. Data are transferred. At the end the Connection is released. We will see Next lecture **TCP and SCTP are Connection oriented Protocol.**

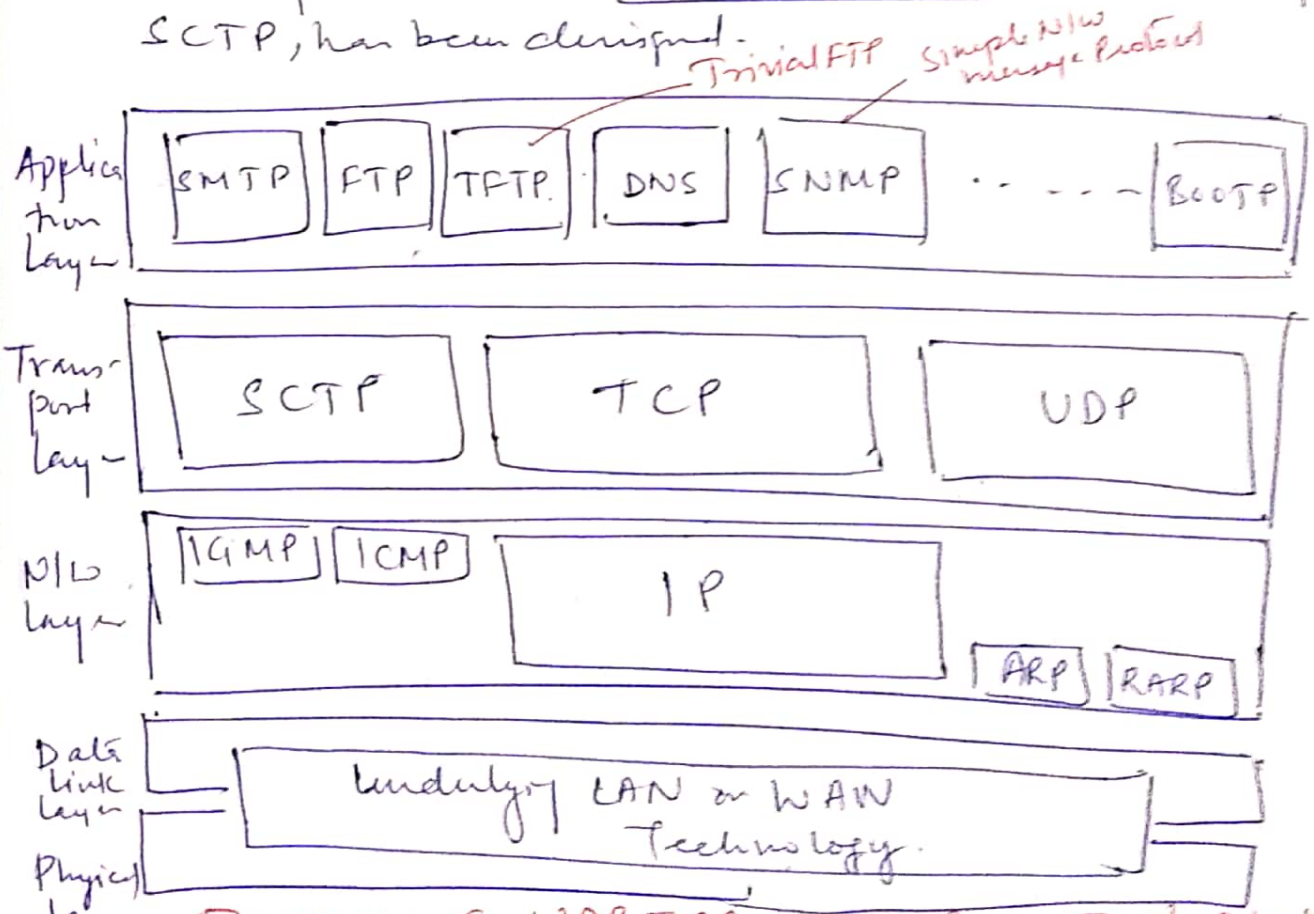
Reliable Vs Unreliable

The TL service can be Reliable or Unreliable
If the AL program need reliability, we use Reliable TLP by implementing flow and error control at a TL. This means a slower and more complex service.

On other hand, If an Application Program does not need reliability because it uses the own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control. Then an Unreliable protocol can be used.

UDP → Connectionless Services / Unreliable
SCTP & TCP → Connection Oriented Services / Reliable

Three Protocol TCP/IP protocol suite specifies two Protocol for the TL → UDP and TCP. Another TL protocol SCTP, has been designed.



POSITION OF UDP, TCP and SCTP in TCP/IP Suite.

TCP - Transmission Control Protocol

The second Transport layer protocol is called Transmission Control Protocol (TCP). TCP, like UDP, is process to process (program to program) Protocol

TCP therefore, like UDP, uses port numbers. Unlike UDP, TCP is a connection oriented protocol; it creats a virtual connection bet^n Two TCP's to send data. In addition, TCP uses flow and error Control Mechanisms at the Transport level.

TCP is called a Connection-Oriented, Reliable Transport protocol. It adds Connection-oriented and Reliability features to the services of IP.

TCP Services :-

Process to Process Communication :-

Like UDP, TCP provides process-to-process communication using ports numbers. Lists of some well known port number used by TCP.

Well known ports used by TCP

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the Sender
9	Discard	Discards any datagram that is received.
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day.
19	Chargen	Returns a string of characters.
20	FTP, Data	File Transfer Protocol (data Connection)
21	FTP, Control	File Transfer Protocol (Control Connection)

23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Stream Delivery Service :-

TCP, unlike UDP, is a stream-oriented protocol. In

* UDP, a process (an Application program) sends message, with predefined boundaries, to UDP for delivery. UDP adds its own header to each of these message and delivers them to IP for transmission.

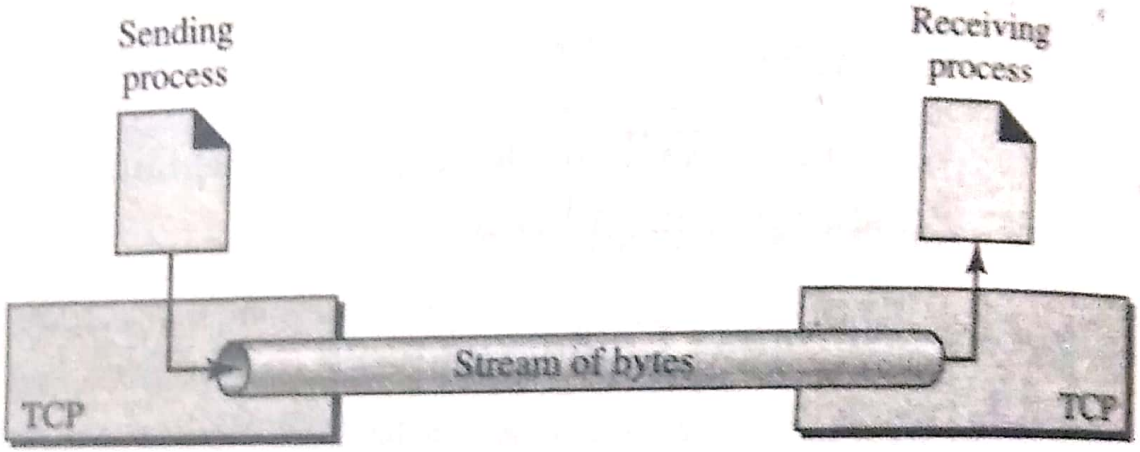
* Each message from the process is called a User datagram and becomes, one IP datagram.

* TCP, on other hand, allows 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 process seem to be connected by an imaginary "Tube" that carries their data across the Internet.

* This imaginary environment is shown in diagram. The sending process produces (writes to) the stream of bytes and the receiving process consumes (Reads from) them.

Figure 23.13 *Stream delivery*



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. (We will see later that these buffers are also necessary for flow and error control mechanisms used by TCP.) One way to implement a buffer is to use a circular array of 1-byte locations as shown in Figure 23.14. For simplicity, we have shown two buffers of 20 bytes each; normally the buffers are hundreds or thousands of bytes, depending on the implementation. We also show the buffers as the same size, which is not always the case.

Figure 23.14 *Sending and receiving buffers*

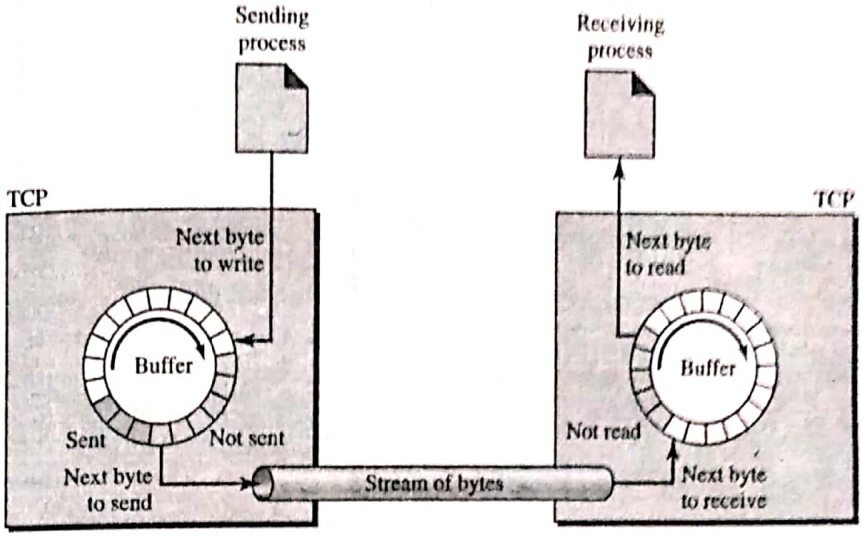


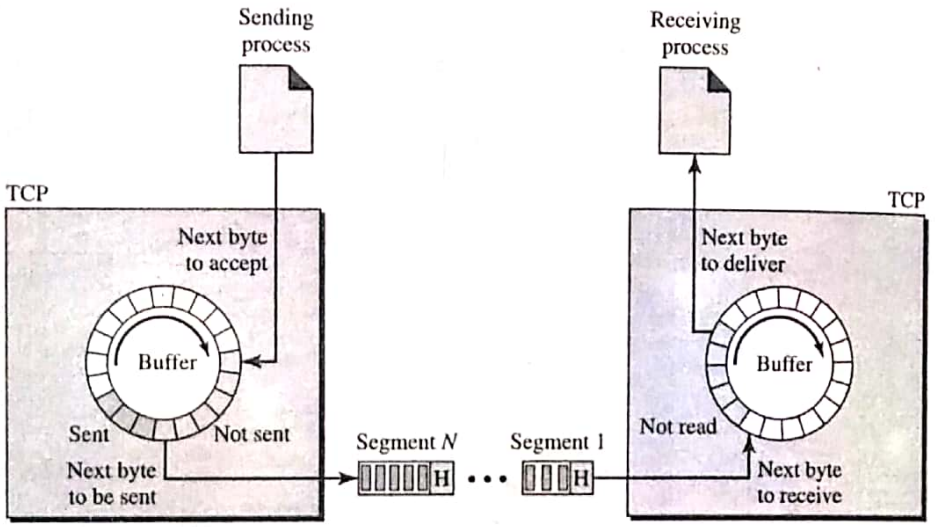
Figure 23.14 shows the movement of the data in one direction. At the sending site, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer). The gray area holds bytes that have been sent but not yet acknowledged. TCP keeps these bytes in the buffer until it receives an acknowledgment. The colored area contains bytes to be sent by the sending TCP. However, as we will see later in this chapter, TCP may be able to send only part of this colored section. This could be due to the slowness of the receiving process or perhaps to congestion in the network. Also note that after the bytes in the gray chambers are acknowledged, the chambers are recycled and available for use by the sending process. This is why we show a circular buffer.

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

Segments 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.** TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission. The segments are encapsulated in IP datagrams and transmitted. This entire operation is transparent to the receiving process. Later we will see that segments may be received out of order, lost, or corrupted and resent. All these are handled by TCP with the receiving process unaware of any activities. Figure 23.15 shows how segments are created from the bytes in the buffers.

Figure 23.15 *TCP segments*



Note that the segments are not necessarily the same size. In Figure 23.15, for simplicity, we show one segment carrying 3 bytes and the other carrying 5 bytes. In reality, segments carry hundreds, if not thousands, of bytes.

Full-Duplex Communication

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

Connection-Oriented Service

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. There is no physical connection. TCP creates a stream-oriented environment in which it accepts the responsibility of

delivering the bytes in order to the other site. The situation is similar to creating a bridge that spans multiple islands and passing all the bytes from one island to another in one single connection. We will discuss this feature later in the chapter.

Reliable Service

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

TCP Features

To provide the services mentioned in the previous section, TCP has several features that are briefly summarized in this section and discussed later in detail.

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. 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 $2^{32} - 1$ 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. We will see that byte numbering is used for flow and error control.

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.

Example 23.3

Suppose a TCP connection is transferring a file of 5000 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 1000 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)

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

When a segment carries a combination of data and control information (piggybacking), it uses a sequence number. If a segment does not carry user data, it does not logically define a sequence number. The field is there, but the value is not valid. However, some segments, when carrying only control information, need a sequence number to allow an acknowledgment from the receiver. These segments are used for connection establishment, termination, or abortion. Each of these segments consumes one sequence number as though it carried 1 byte, but there are no actual data. If the randomly generated sequence number is x, the first data byte is numbered $x + 1$. The byte x is considered a phony byte that is used for a control segment to open a connection, as we will see shortly.

Acknowledgment Number As we discussed previously, communication in TCP is full duplex; when a connection is established, both parties can send and receive data at the same time. Each party numbers the bytes, usually with a different starting byte number. 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. In addition, the acknowledgment number is cumulative, which means that 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. The term *cumulative* here means that 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 0.

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.

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, as we will see later.

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.

TCP (Transmission Control Protocol) TCP is most widely used for data transmission in Com^n NW such as Internet. TCP provides Process-to-Process Communication PORT Number

Features of TCP :-

1) Process to Process Com^n.
2) Port Numbers are used for Communication
3) Numbering system in TCP - It is used to keep a track of segments that are transmitted or Recieved. There is no field for a segment Number value in the segment header.
There are two fields called Sequence number and Acknowledgment Number.

a) Sequence Number :- After the bytes have be numbered TCP assigns a sequence No. of each segment that is being sent. The sequence No. of for each segment is the Number of the first byte carried in that segments.

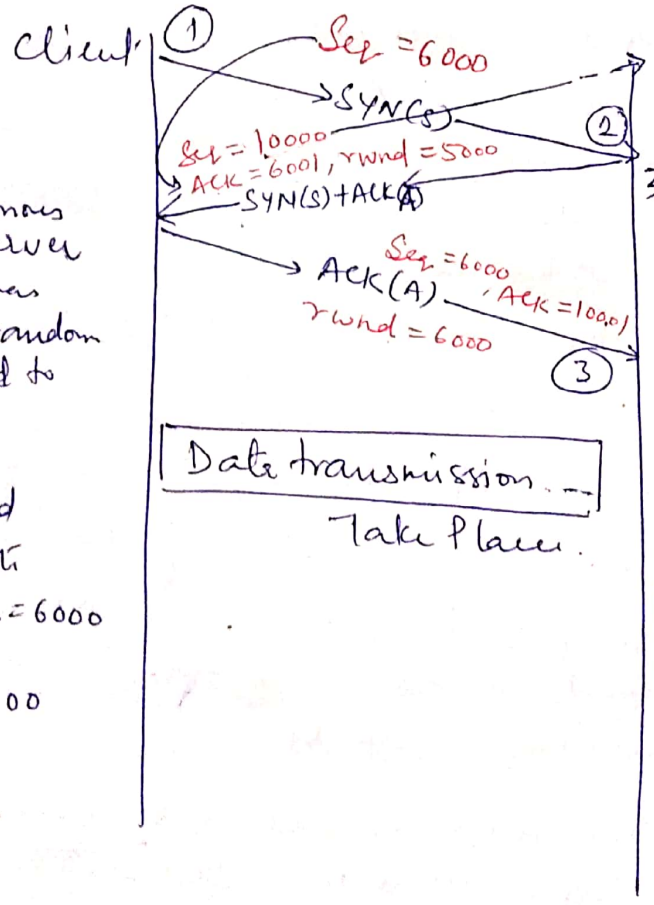
b) Byte Number - TCP numbers all data ~~types~~ bytes that are transmitted in a Connection.

c) Acknowledgment Number :- Communication in TCP is full duplex : When a connection is established

Both parties can send and receive Data at the same time. Each party numbers the bytes, with a different starting byte number. The Sequence number in each direction shows the Number of first byte carried by the segment. Each party also uses an Acknowledgment No. to confirm the bytes it has received. However the acknowledgment No. defines the No. of the Next bytes that party expect to receive.

Three-Way Handshaking:-

TCP is connection oriented Protocol so that we use Three way Handshaking method. It is the Connection establishment in TCP. TCP is work on FULLDUPLEX mode.



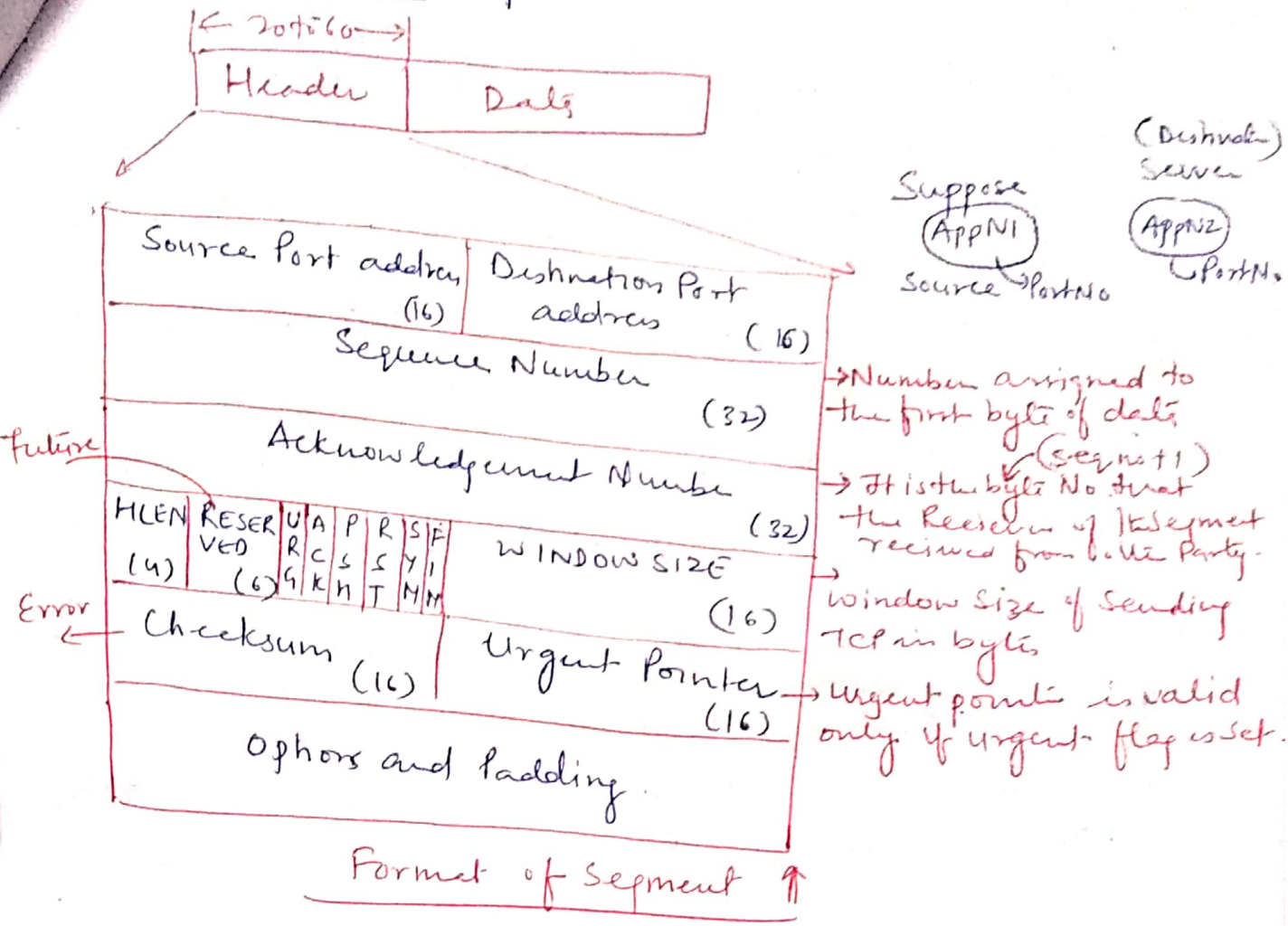
1) Client send SYN(S) Synchronous flag (S) to server In this flag has Seq = 6000 random No. is send to Server.

2) Server Send SYN(S) + ACK(S) Recieved from the SYN(S). Server seq No = 10000 ACK = 6001, Reciving window size = 5000.

3) Client send ACK(A) with Seq no same = 6000 Ack = 10001, rwnd = 6000

TCP Segment

A packet in TCP is called Segment



Format of Segment

Segment consist of 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.

1) Source port address:- This 16 bit field that defines the port number of the application Program in the host that is sending the segment. This serves the same purpose of source port address in UDP header

2) Destination Port address:- This 16 bit field that defines the Port Number of the application program in the host that Receiving the segment. This serves the same purpose as the destination port address in the UDP header

3) Sequence Number:- This 32 bit field defines the Number assigned to the first byte of data contained in this segment.
TCP is a stream transport protocol to ensure connectivity each byte to be transmitted is numbered. The Seq No. tells the destination which byte is the Sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial Sequence Number (ISN). which is different in each direction.

4) Acknowledgment No:- This 32 bit field defines the byte Number that the Receiver of the Segment is expecting to receive from the other party. If the receiver of the Segment has successfully received byte Number x from the other party. It defines x+1 as the Ack Number.

5) Header length - This is 4 bit field indicating the Number of 4 byte words in the TCP header. The length of header can be between 20 and 60 bytes therefore, the value of this field can be between 5 (5×4=20) and 15 (15×4=60).

6) Reserved :- This is a 6 bit field reserved for future

7) Control - This field defines ~~6 reserved bits~~ 6 different control bits or flags as.

URG – Urgent pointer is valid
ACK – Acknowledgment is valid
PSH – Request for Push
RST – Reset the Connection
SYN – Synchronize Seq No
FIN – Terminate the Connection

URG	Ack	PSH	RST	SYN	FIN

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

Description of flags in the control field

Flag	Description
URG	The value of the Urgent pointer filed is valid
Ack	The value of the Ack field is valid
PSH	Push the Data
RST	Reset the Connection.
SYN	Synchronize Seq Number during Connection.
FIN	Terminate the Connection

8) **Window Size** – This field defines the size of the window, in bytes, that the other party maintain. Note that the length of field is 16 bits, which mean that the Maximum Size of the window is 65535 bytes. The value is referred to as Receiving window (rwnd) and is determined by the Receiver.

9) Checksum - This 16 bit field contains the checksum. The calculation of the Checksum for TCP follow as the same procedure as in UDP. This is used for Error Technique.

10) Urgent Pointer - This is 16 bit field which is valid only if the Urgent field flag is set. it used when the segment contain Urgent data.

11) options - There can be up to 40 bytes of optional information in the TCP header.

Important Question from Unit-5 (Transport layer)

Q1. Write any five difference bet UDP and TCP.

Q2. Differentiate bet Connection less and Connection Oriented Service.

Q3. Explain the Service provided by the TCP.

Q4. Explain the Leaky bucket Algo with the help of suitable diagram.

Q5. Differentiate leaky Bucket & Token bucket Algo.

Q6. Draw and Explain TCP header segment Structure.

Q7. Explain TCP Service model in detail.

Q8. Discuss the TCP Connection establishment and Release

Q9. Write a Technical note on TCP Congestion Control.

UDP (User Datagram Protocol):-

The User Datagram Protocol UDP is called a Connection-less, unreliable transport protocol.

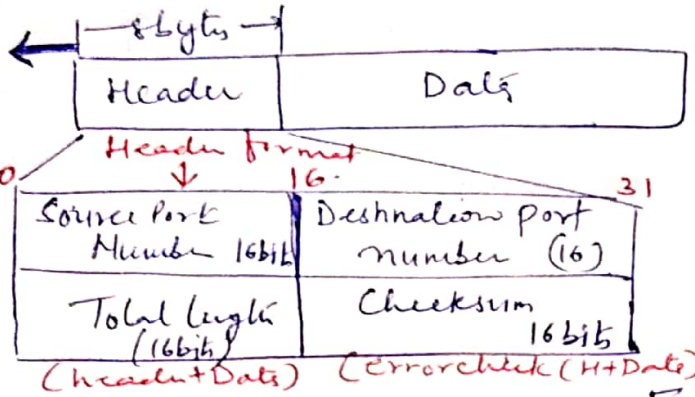
UDP is very simple Protocol using a minimum of overhead. If a process want to send a small message and does not care much about Reliability, It can use UDP. Sending a small message by using UDP takes much less interaction between the sender and Receiver than using TCP or SCTP.

Well Known Ports for UDP: Some well known port Numbers used by UDP. Some ports number can be used by both UDP and TCP.

Port	Protocol	Description
7	Echo	Echoes a received back to the sender
9	Discard	Discards any Datagram that is Received
11	User	Active User
13	Daytime	Returns the date + the time
17	Quoti	Return a quote of the day
19	chargen	Domain Name Service
67	BOOTPs	Server port to download Bootstrap file
68	BOOTPc	Client port to Download Bootstrap file
53	Name Server	Domain Name Service
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol

User Datagram Format :-

UDP, called User Datagrams have a fixed size header of 8 bytes.



User Datagram Format

1) Source Port Number :- This is the port Number used by the the Process running on the source host. It is 16 bit long which mean that the port No. can be Range from 0 to 65535.

If the source host is client (a client sending a Request) the port No. in most cases is an Ephemeral port Number requested by the process and chosen by the UDP sfw running on the Source host.

If the source host is the Server (a server sending a Response), the Port Number, is most cases is well known port Number

2) Destination Port Number :- This a port number used by the Process running on the destination host. It is also 16 bit long. If the Destination host is the Server (a client Sending a Request) the port number, in most cases is a well known port Number

If the Destination host is Client (a server sending a response). the port number, in most cases is an ephemeral port number. In this cases, the Server copies the ephemeral port Number it has recieved in the Request packet.

3) **Total Length** :- This is 16 bit field that defines the total length of the user datagram, header plus Data. The 16 bit can define a total length of 0 to 65,535 bytes.

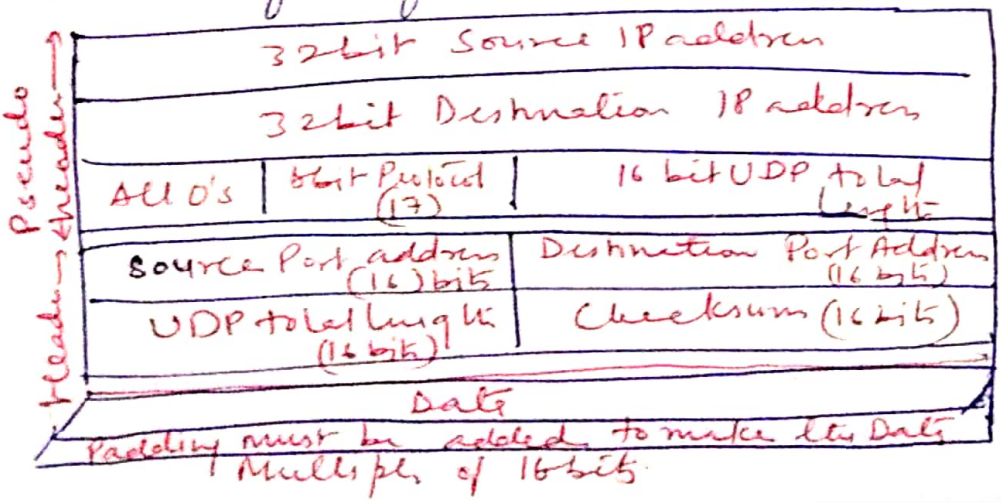
UDP length = IP length - IP header's length

The length field in UDP user datagram is not necessary. A user datagram is encapsulated in a IP datagram. There is field in IP datagram that defines the total length. There are another field in IP Datagram that defines the length of the header. So if we substract the value of the second field from the first, we can obtain the length of UDP datagram that is encapsulated in an IP datagram.

4) **Checksum** - This field is used to detect errors over the entire User Datagram. (header plus Data)

UDP checksum calculation is different from one for IP and ICMP. Here the checksum includes three sections: a pseudoheader, UDP header and data comming from the Application layer.

The pseudoheader is the part of the header of the IP packet in which the User Datagram is Encapsulated with some field filled with 0's.



UDP operation

1) Connectionless Service - UDP is provide a connectionless service. This means that each user Datagram sent by UDP is an Independent datagram.

2) Flow and Error Control - UDP is very simple, unreliable Transport protocol. There is no flow Control and error control.

3) Encapsulation and Decapsulation
To send a message from one process to another the UDP protocol encapsulates and decapsulates message in an IP datagram.

Uses of UDP :-

Some uses of UDP protocol :-

1) UDP is used for some route updating protocol such as Routing Information Protocol (RIP)
2) UDP is used for Management process such as SNMP
3) UDP is a suitable Transport protocol for Multicasting, Multicasting capability is embedded in UDP sftw but not in TCP sftw.
4) UDP is suitable for a Process with internal flow and error control Mechanisms. For Example the TFTP (Trivial FTP) process includes flow and Error Control. It can easily use UDP.

Features of UDP :-

1) Process-to Process Communication
2) Port number are used for Communication
3) Connection less Service
4) Un-Reliable service provided by UDP.
5) Flow Control → No
6) Error Control → No
7) Congestion Control → No

IMP Difference b/w TCP and UDP

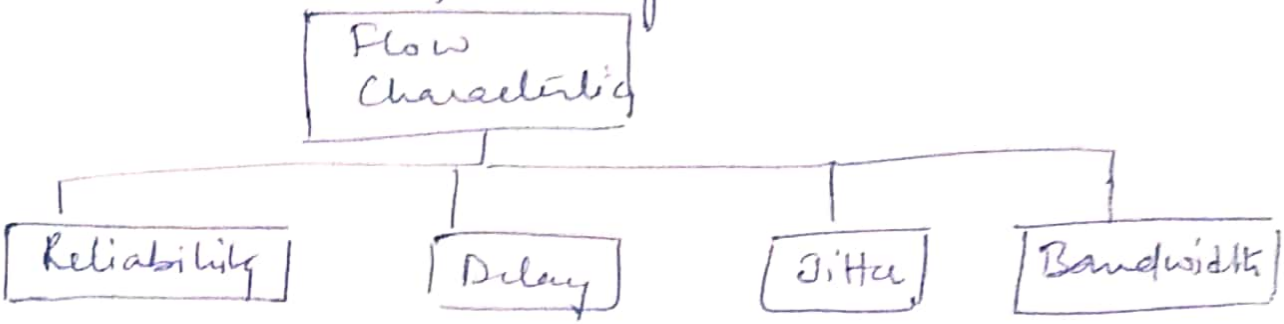
Characteristic / Description	UDP	TCP
1) General	1) Simple, High speed, with low functionality protocol	1) Full feature, Reliable Data transfer.
2) Connection setup	2) Connection less	2) Connection Oriented
3) Data Interface to Application	3) Message based data is sent	3) Continuous stream based Data is sent.
4) Reliability	4) Unreliable (ACK X)	4) Reliability (ACK V)
5) Retransmission	5) Not Performed	5) If Data is loss ⇒ Retransmission Done
6) Flow control	6) No	6) Sliding window ⇒ Flow control
7) Error Control	7) No	7) Yes
8) Overhead (Extra Feature)	8) Very low	8) Low, But Higher than UDP
9) Transmission Speed	9) Very High speed	9) High, lower than UDP.
10) Application that uses Protocol	10) DNS, BOOTP, TFTP DHCP, SNMP (Multimedia Application)	10) FTP, TELNET, SMTP, DNS, HTTP.

Quality of Service :-

In QoS we try to create a appropriate Environment for the traffic.

Flow characteristics :- Traffic is flow of data.

Four types of characteristics are attributes to a flow : Reliability, Delay, Jitter and Bandwidth



a) Reliability -

* Reliability is a characteristics that a flow needs.
* Lack of Reliability means loosing a packet or acknowledgement.
* For Example, it is more important that Electronic mail, file transfer and Internet access have Reliable transmission than Telephony or audio Conferencing.

b) Delay - Source to destination delay is another flow characteristic. Again application can tolerate delay.

* Delay is nothing but the time required to reach the source to destination.
* In this case, telephony, audio conferencing, vedio conferencing and Remote login need minimum delay. while delay in file transfer or email is less important.

Source to destination delay

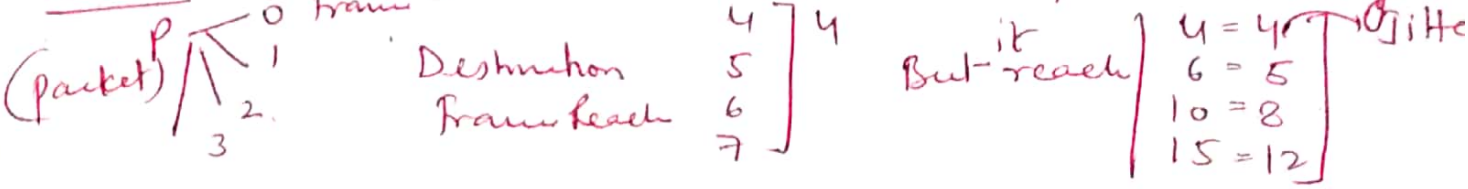
Source —packet-0 time→ Destination [4 packets 4 time]

3) **Jitter** - Jitter is the variation in delay for packet belonging to the same flow. For Example Four Packet depart at time 0, 1, 2, 3 and arrive 20, 21, 22, 23, all have the same delay, 20 units of time.

On other hand if above four packet Arrive at 21, 23, 21, and 28, they will have different delay: 21, 22, 19 and 24

Jitter is defined as the variation in packet delay.

For Ex



In case of Audio and Vedio Jitter is much required.

4) **Bandwidth** -

Different application need different Bandwidth In veido conferencing we need to send millions of bits per second to refresh a color screen while total number of bits in an email may not reach even a million.

Techniques To Improve QOS :-

Four Common Methods : scheduling, traffic shaping, admission control, And Resource Reservation.

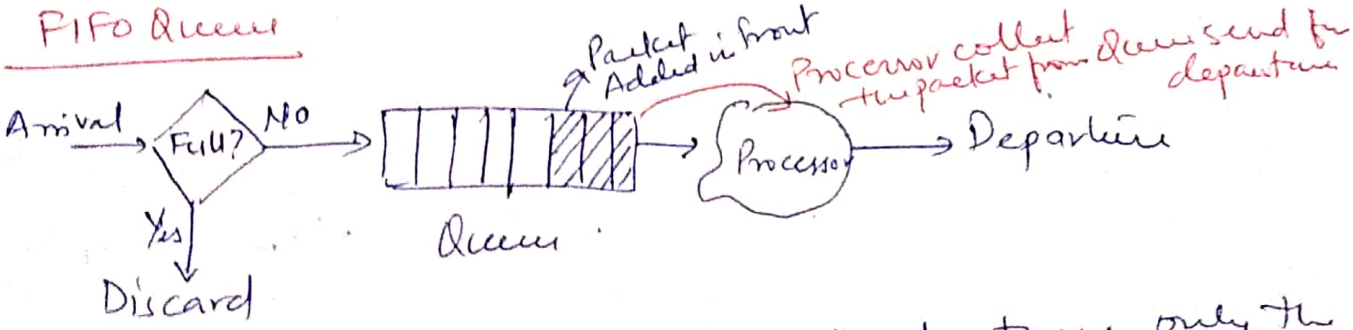
Scheduling -

Packets from different flows arrive at a switch or Router for Processing. Several scheduling techniques are designed to improve the quality of service. Three types are : FIFO queuing, priority queuing and weighted fair queuing.

FIFO Queuing

In first in-first out (FIFO) queuing, packet wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up & new packet will be discarded. A FIFO queue is familiar to those who have had to wait for a bus at a bus stop. Diagram show the concept of fifo queue.

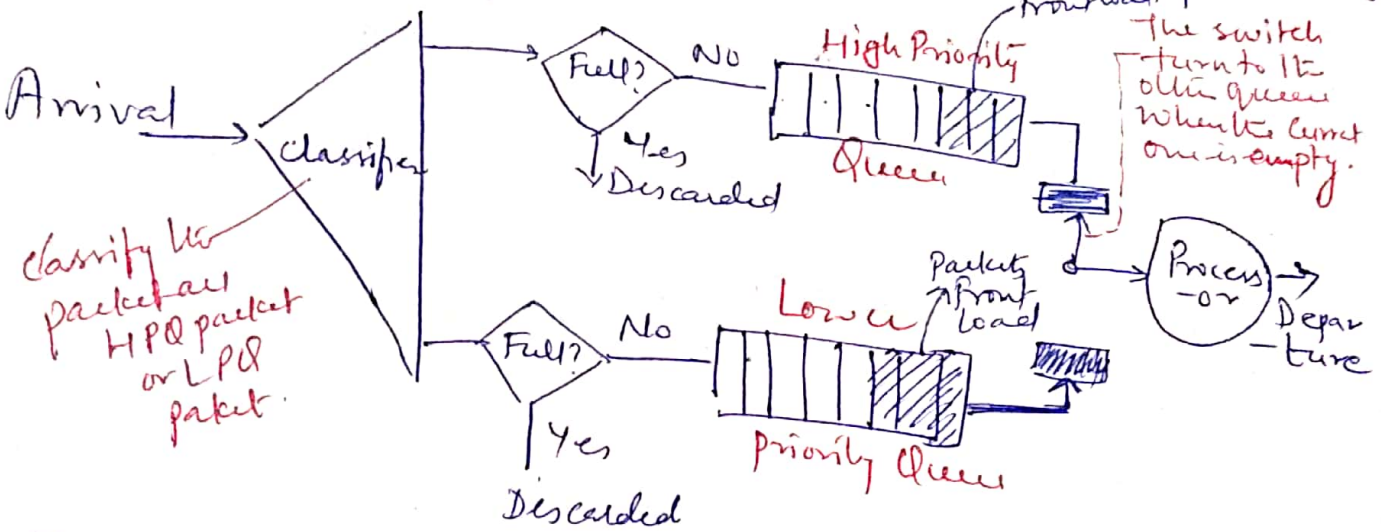
FIFO Queue



If queue can process 10 packet at a time only the remaining packet are discard only 10 packet are Remain in Queue.

ii) Priority Queuing:

In Priority queuing packet are first assigned to a Priority class. Each priority class has its own queue. The packet in the highest-priority queue are processed first. Packet in lower priority queue are processed last. Note that the system does not stop serving a queue until it is empty. Diagram show priority Queuing with two priority levels.

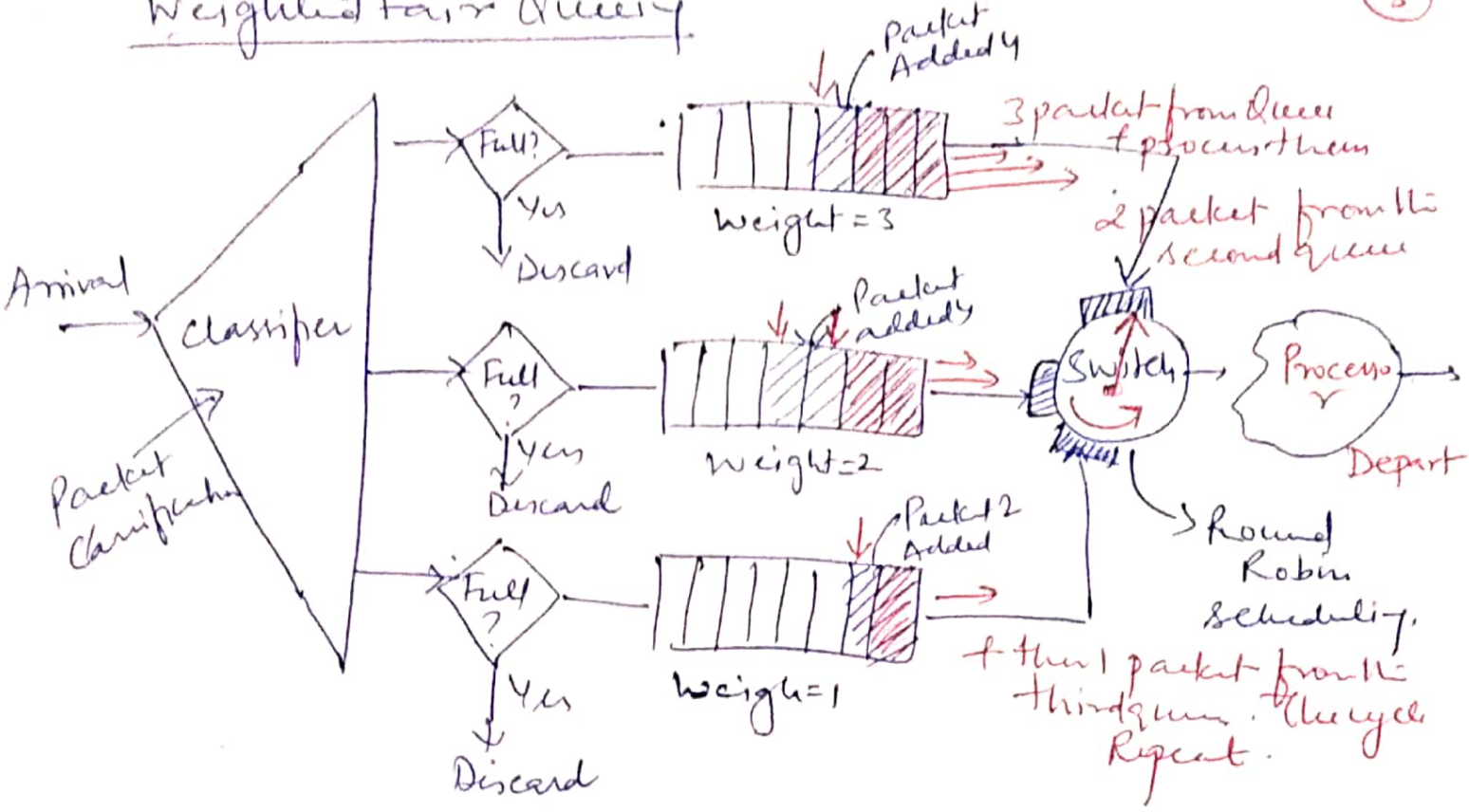


Processor choose first High Priority Queue Packet first and Processor depart the packet.

iii) Weighted Fair Queuing :- In this the Queues are weighted based on the Priority of Queue.

In this Technique, the packets are still assigned to different classes and admitted to different queue. The queues, however, are weighted based on the priority of the queue; High priority mean a higher weight. The system process packet in each queue in a Round Robin fashion with the No. of packets selected from each queue based on the corresponding weight.

Weighted Fair Queuing



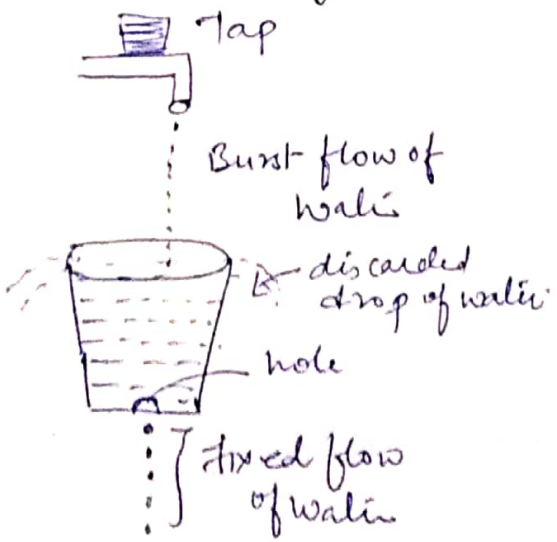
For Example if the weights are 3, 2 and 1. Three packets are processed from the first queue two from the second queue and one from the third queue. If the system does not impose priority on the classes, all weights can be equal. In this way we have fair queuing with priority.

Traffic shaping - QoS (Quality of Service)

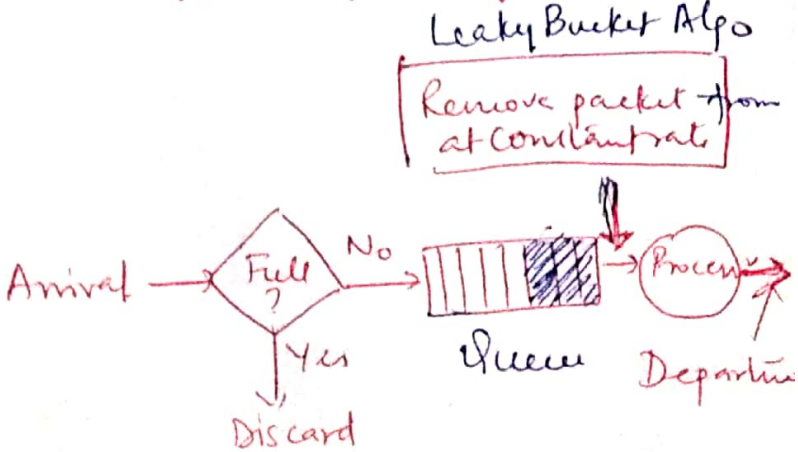
It is the mechanism to control the amount and the rate of the traffic sent to the N/w. Two Techniques can shape traffic: leaky bucket and token bucket

Leaky Bucket -

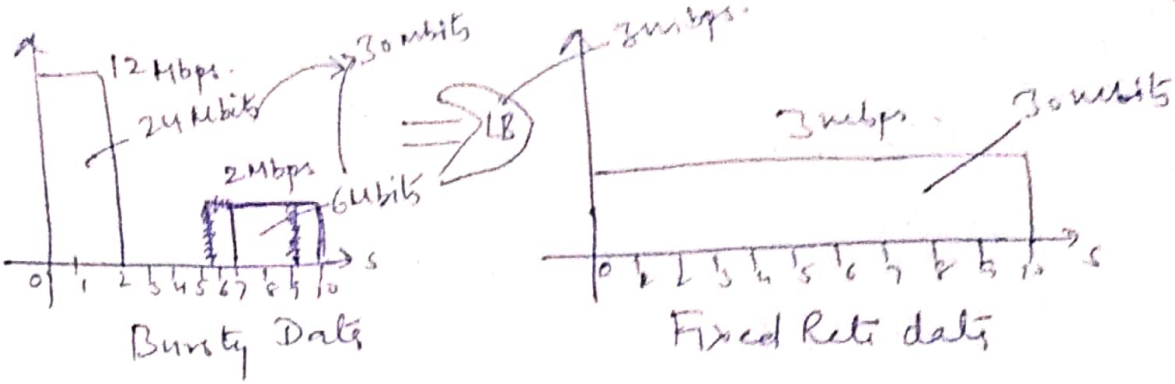
If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water ~~leaks~~ leaks does not depend on the rate at which ~~as~~ ~~input~~ the water is input to the bucket unless the bucket is empty. The input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



This Algorithm shapes Bursty traffic into Fixed-rate traffic by averaging the data rate.



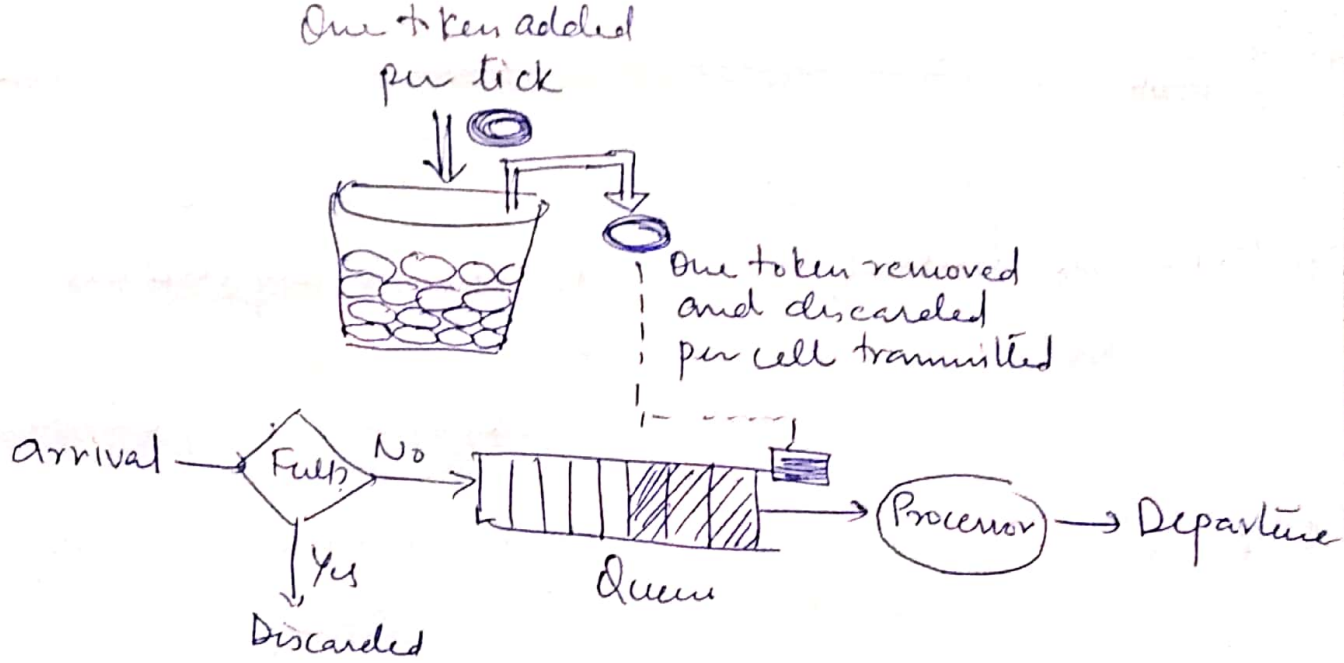
Leaky Bucket Algo

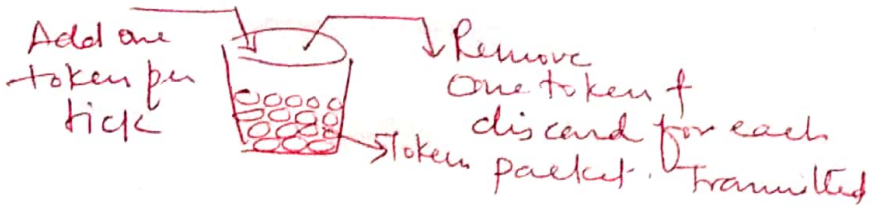


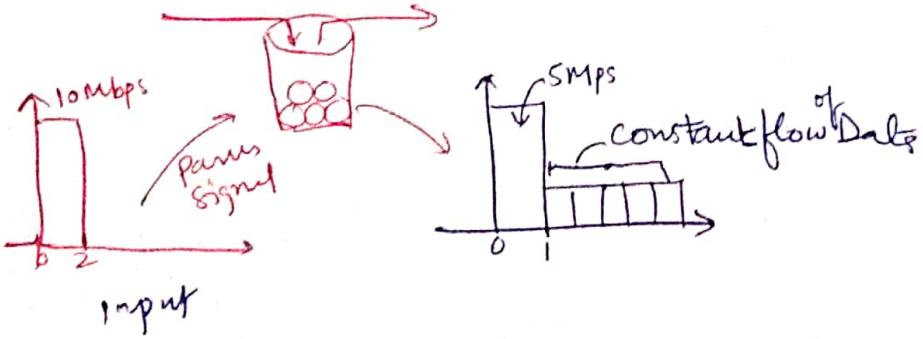
Token Bucket Algo:-

It allows Bursty traffic at a regulated Maximum rate.

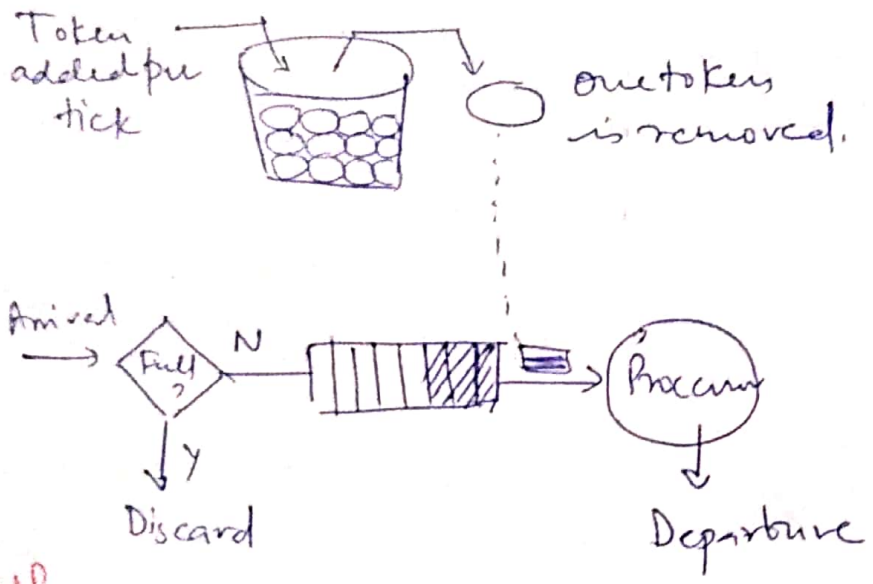
Token Bucket







Implementation of Token bucket



IMP

What is Difference b/w Token bucket and leaky Bucket

Token Bucket	Leaky Bucket
1) Token Dependent	1) Token Independent.
2) If Bucket is full token is discarded but not the packet.	2) If Bucket is full then Packets are discarded.
3) Packets can only transmit when there are enough token	3) Packets are transmitted continuously.
4) Allows large burst to be send at faster rate	4) Sends the Packet at a Constant Rate.
5) Saves token to send large bursts	5) No Concept of Token