UNIT 2

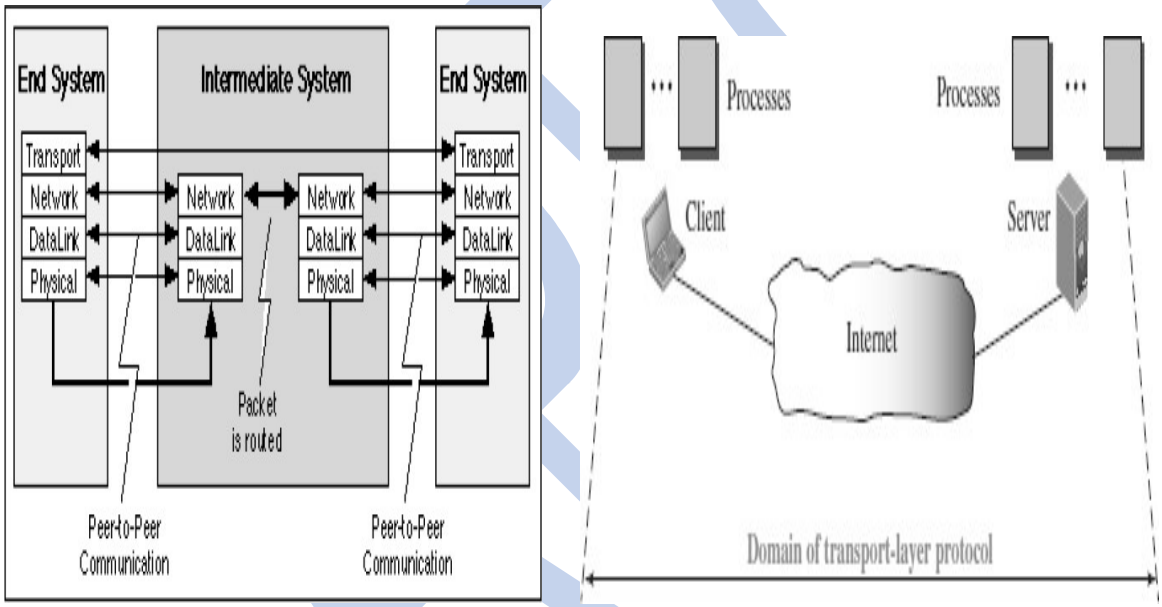
TRANSPORT LAYER

1. INTRODUCTION

The transport layer is the fourth layer of the OSI model and is the core of the Internet model. It responds to service requests from the session layer and issues service requests to the network Layer. The transport layer provides transparent transfer of data between hosts.

It provides end-to-end control and information transfer with the quality of service needed by the application program.

It is the first true end-to-end layer, implemented in all End Systems (ES).



2. TRANSPORT SERVICES

The transport layer is located between the network layer and the application layer. The transport layer is responsible for providing services to the application layer; it receives from the network layer. The services that can be provided by the transport layer are

- Process-to-Process Communication
- Addressing: Port Numbers
- Encapsulation and Decapsulation
- Flow Control
- Error Control
- Congestion Control

- Multiplexing and demultiplexing

Process-to-Process Communication

The Transport Layer is responsible for delivering data to the appropriate application process on the host computers. This involves multiplexing of data from different application processes, i.e., forming data packets, and adding source and destination port numbers in the header of each Transport Layer data packet. Together with the source and destination IP address, the port numbers constitute a network socket, i.e. an identification address of the process-to-processcommunication.

Addressing: Port Numbers

Ports are the essential ways to address multiple entities in the same location. Using port addressing it is possible to use more than one network-based applicationat the same time.

Three types of Port numbers are used:

✓ ***Well-known ports*** - These are permanent port numbers. They range between0 to 1023.These port numbers are used by Server Process.

✓ ***Registered ports*** **-** The ports ranging from 1024 to 49,151 are not assigned orcontrolled.

✓ ***Ephemeral ports (Dynamic Ports)*** **–** These are temporary port numbers. They range between 49152–65535.These port numbers are used by Client Process**.**

Encapsulation and Decapsulation

To send a message from one process to another, the transport-layer protocol encapsulates and decapsulates messages.

Encapsulation happens at the sender site. The transport layer receives the data and adds the transport-layer header. Decapsulation happens at the receiver site. When the message arrives at the destination transport layer, the header is dropped and the transport layer delivers the message to the process running at the application layer.

Multiplexing and Demultiplexing

Whenever an entity accepts items from more than one source, this is referred to as ***multiplexing*** (many to one).

Whenever an entity delivers items to more than one source, this is referred to as ***demultiplexing*** (one to many). The transport layer at the source performs multiplexing. The transport layer at the destination performs demultiplexing

3. TRANSPORT LAYER PROTOCOLS

Three protocols are associated with the Transport layer. They are

1. UDP
2. TCP
3. SCTP

Each protocol provides a different type of service and should be used appropriately.

UDP - UDP is an unreliable connectionless transport-layer protocol used for its Simplicity and efficiency in applications where error control can be provided by the application-layer process.

TCP - TCP is a reliable connection-oriented protocol that can be used in any application where reliability is important.

SCTP - SCTP is a new transport-layer protocol designed to combine some features of UDP and TCP in an effort to create a better protocol for multimedia communication.

Position of transport-layer protocols in the TCP/IP protocol suite



4. USER DATAGRAM PROTOTOCOL

User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol. UDP adds process-to-process communication to best-effort service provided by IP. UDP is a very simple protocol using a minimum of overhead. UDP is a simple demultiplexer, which allows multiple processes on each host to communicate. UDP does not provide flow control, reliable or ordered delivery. UDP can be used to send small message where reliability is not expected. Sending a small message using UDP takes much less interaction between the sender and receiver. UDP allow processes to indirectly identify each other using an abstract locator calledport or mailbox

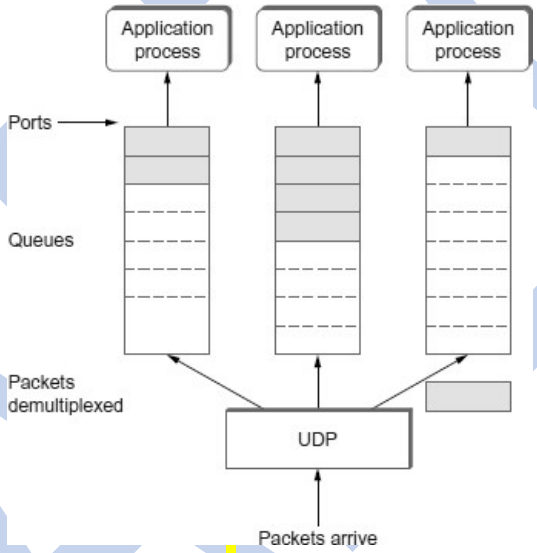
UDP PORTS

- Processes (server/client) are identified by an abstract locator known

as port.

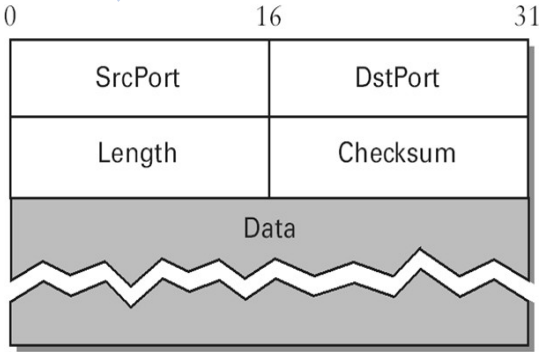
- Server accepts message at *well known port*.
- Some well-known UDP ports are 7–Echo, 53–DNS, 111–RPC, 161–SNMP, etc.
- < *port*, *host* > pair is used as key for demultiplexing.
- Ports are implemented as a *message queue*.
- When a message arrives, UDP *appends* it to end of the queue.
- When queue is *full*, the message is discarded.
- When a message is *read*, it is removed from the queue.
- When an application process wants to receive a message, one is removed from thefront of the queue.

If the queue is empty, the process blocks until a message becomes available



UDP DATA PACKET FORMAT

- UDP packets are known as user *datagrams* .
- These ***user datagrams,*** have a fixed-size header of 8 bytes made of four fields, eachof 2 bytes (16 bits).



Source port number

- Port number used by process on source host with 16 bits long.
- If the source host is client (sending request) then the port number is a temporary one requested by the process and chosen by UDP
- If the source is server (sending response) then it is well known port number.

Destination port number

- Port number used by process on Destination host with 16 bits long.
- If the destination host is the server (a client sending request) then the port number is a well-known port number.
- If the destination host is client (a server sending response) then port number is a temporary one copied by server from the request packet.

Length

- This field denotes the total length of the UDP Packet (Header plus data)
- The total length of any UDP datagram can be from 0 to 65,535 bytes.

Checksum

- UDP computes its checksum over the UDP header, the contents of the messagebody, and something called the pseudo header.
- The pseudo header consists of three fields from the IP header—protocol number,source IP address, destination IP address plus the UDP length field

Data

- Data field defines that actual payload to be transmitted.
- Its size is variable**.**

UDP SERVICES

Process-to-Process Communication

UDP provides a process-to-process communication using socket addresses**,** a combination of IP addresses and port numbers.

Connectionless Services

- UDP provides a connectionless service.
- There is no connection establishment and no connection termination.

- 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.
- Each user datagram can travel on a different path.

Flow Control

- UDP is a very simple protocol.
- There is no flow control, and hence no window mechanism.
- The receiver may overflow with incoming messages.
- The lack of flow control means that the process using UDP should provide for thisservice, if needed.

Error Control

- 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 issilently discarded.
- The lack of error control means that the process using UDP should provide for thisservice, if needed.

Checksum

- UDP checksum calculation includes three sections: a pseudo header, the UDP header,and the data coming from the application layer.
- The pseudo header is the part of the header in which the user datagram is to beencapsulated with some fields filled with 0s.

Congestion Control

- Since UDP is a connectionless protocol, it does not provide congestion control.
- UDP assumes that the packets sent are small and sporadic (occasionally or at irregular intervals) and cannot create congestion in the network.

- This assumption may or may not be true, when UDP is used for interactive real-timetransfer of audio and video.

Encapsulation and Decapsulation

- To send a message from one process to another, the UDP protocol encapsulates anddecapsulates messages.

Queuing

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

Multiplexing and Demultiplexing

- In a host running a transport protocol suite, there is only one UDP but possibly several processes that may want to use the services of UDP.
- To handle this situation, UDP multiplexes and demultiplexes.

APPLICATIONS OF UDP

- UDP is used for management processes such as SNMP.
- UDP is used for route updating protocols such as RIP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability isembedded in the UDP software
- UDP is suitable for a process with internal flow and error control mechanisms suchas Trivial File Transfer Protocol (TFTP).
- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is normally used for interactive real-time applications that cannot tolerateuneven delay between sections of a received message.

5. TRANSMISSION CONTROL PROTOCOL

TCP is a reliable, connection-oriented, byte-stream protocol's guarantees the reliable, in-order delivery of a stream of bytes. It is a full-duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction.

TCP includes a flow-control mechanism for each of these byte streams that allow thereceiver to limit how much data the sender can transmit at a given time. TCP supports a demultiplexing mechanism that allows multiple application programs on any given host to simultaneously carry on a conversation with their peers.

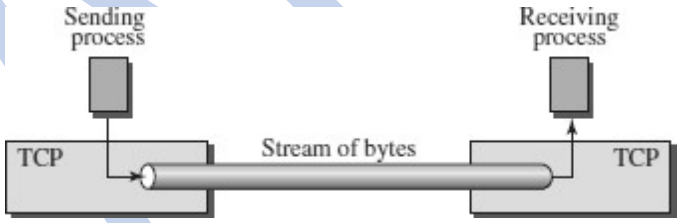
TCP also implements congestion-control mechanism. The idea of this mechanism isto prevent sender from overloading the network. Flow control is an end-to-end issue, whereas congestion control is concerned with how host and network interact.

TCP SERVICES

Process-to-Process Communication: TCP provides process-to-process communication using port numbers.

Stream Delivery Service: TCP is a stream-oriented protocol. TCP 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 bytes across the Internet. The sending process produces (writes to) the stream and the receiving processconsumes (reads from) it.

Full-Duplex Communication:



TCP offers full-duplex service, where data can flow in both directions at the sametime. Each TCP endpoint then has its own sending and receiving buffer, and segmentsmove in both directions.

Multiplexing and Demultiplexing:

TCP performs multiplexing at the sender and demultiplexing at the receiver.

Connection-Oriented Service:

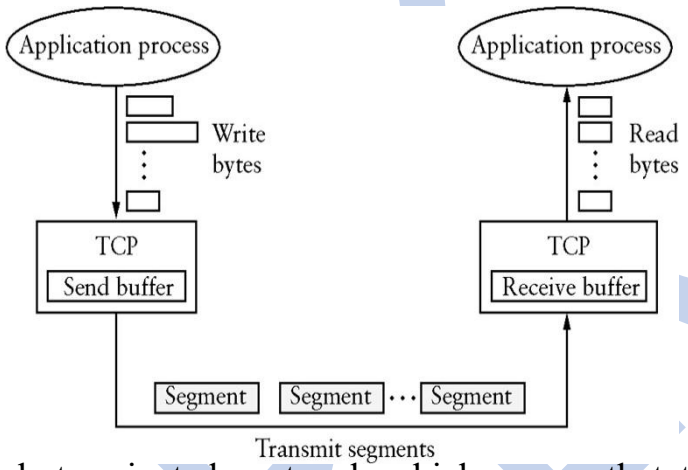
TCP is a connection-oriented protocol. A connection needs to be established for each pair of processes. When a process at site A wants to send to and receive data from anotherprocess at site B, the following three phases occur:

- The two TCP's establish a logical connection between them.
- Data are exchanged in both directions.
- The connection is terminated.

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

TCP SEGMENT

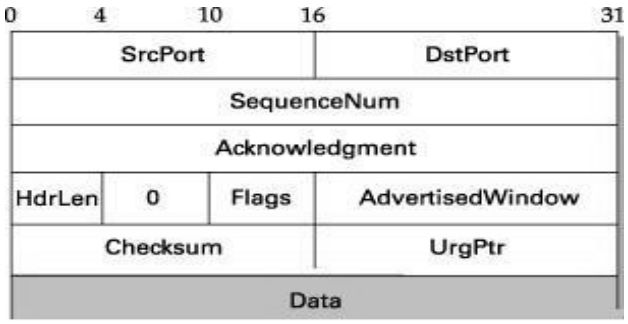
A packet in TCP is called a segment. Data unit exchanged between TCP peers are called ***segments.*** A TCP segment encapsulates the data received from the application layer. The TCP segment is encapsulated in an IP datagram, which in turn is encapsulated ina frame at the data-link layer.



TCP is a byte-oriented protocol, which means that the sender writes bytes into a TCP connection and the receiver reads bytes out of the TCP connection. TCP does not, itself, transmit individual bytes over the Internet. TCP on the source host buffers enough bytes from the sending process to fill a reasonably sized packet and then sends this packet to its peer on the destination host. TCP on the destination host then empties the contents of the packet into a receivebuffer, and the receiving process reads from this buffer at its leisure. TCP connection supports byte streams flowing in both directions. The packets exchanged between TCP peers are called segments, since each onecarries a segment of the byte stream.

TCP PACKET FORMAT

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



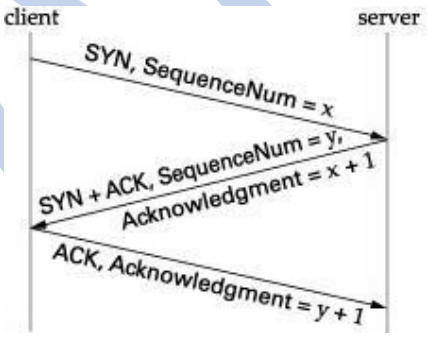
TCP CONNECTION MANAGEMENT

TCP is connection-oriented. A connection-oriented transport protocol establishes a logical path between the source and destination. All of the segments belonging to a message are then sent over this logical path. In TCP, connection-oriented transmission requires three phases: Connection Establishment, Data Transfer and Connection Termination.

Connection Establishment

While opening a TCP connection the two nodes(client and server) want to agree on aset of parameters.

The parameters are the starting sequence numbers that is to be used for their respective byte streams. Connection establishment in TCP is a *three-way handshaking*.



- Client sends a SYN segment to the server containing its initial sequence number (Flags = SYN, Sequence Num = x)
- Server responds with a segment that acknowledges client's segment and specifies its initial sequence number (Flags = SYN + ACK, ACK = x + 1 Sequence Num = y).

- Finally, client responds with a segment that acknowledges server's sequence number (Flags = ACK, ACK = y + 1).
- The reason that each side acknowledges a sequence number that is one larger than the one sent is that the Acknowledgment field actually identifies the "next sequence number expected,"
- A timer is scheduled for each of the first two segments, and if the expected response is not received, the segment is retransmitted.

Data Transfer

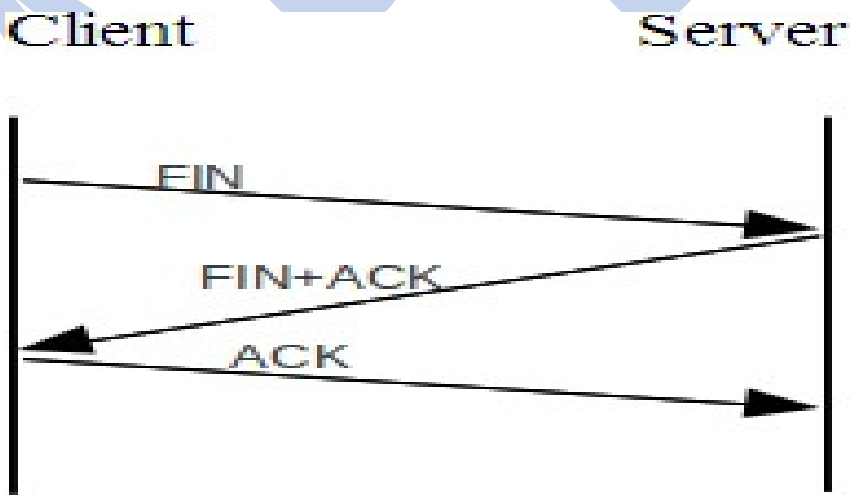
- After connection is established, bidirectional data transfer can take place.
- The client and server can send data and acknowledgments in both directions.
- The data traveling in the same direction as an acknowledgment are carried on the same segment.
- The acknowledgment is piggy backed with the data.

Connection Termination

Connection termination or teardown can be done in two ways:

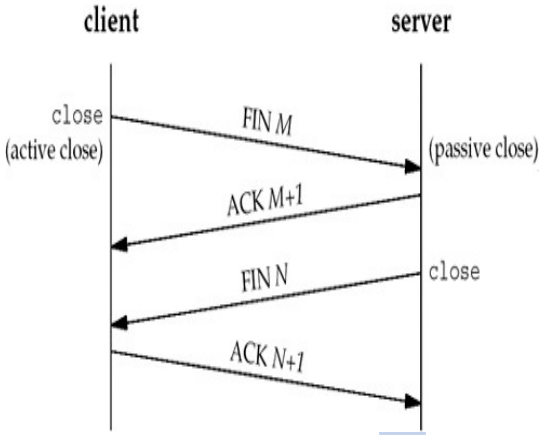
Three-way Close and Half-Close

Three-way Close—Both client and server close *simultaneously*.



Client sends a FIN segment. The FIN segment can include lastchunk of data. Server responds with FIN + ACKsegment to inform its closing. Finally, client sends an ACKsegment

Half-Close—Client stops sending but receives data.



Client half-closes the connection by sending a FIN segment. Server sends an ACK segment. Data transfer from client to the server *stops*. After sending all data, server sends FIN segment to client, which is acknowledged by the client.

6. CONGESTION CONTROL

Congestion in a network may occur if the *load* on the network (the number of packets sent to the network) is greater than the *capacity* of the network (the number of packets a network can handle). Congestion control refers to the mechanisms and techniques that control thecongestion and keep the load below the capacity. Congestion Control refers to techniques and mechanisms that can either preventcongestion, before it happens, or remove congestion, after it has happened Congestion control mechanisms are divided into two categories

1. Open loop
2. Closed loop

Congestion occurs if load (number of packets sent) is greater than capacity of the network (number of packets a network can handle). When load is less than network capacity, throughput increases proportionally. When load exceeds capacity, queues become full and the routers discard some packets and throughput declines sharply. When too many packets are contending for the same link The queue overflows, Packets get dropped, Network is congested, Network should provide a congestion control mechanism to deal with such a situation.

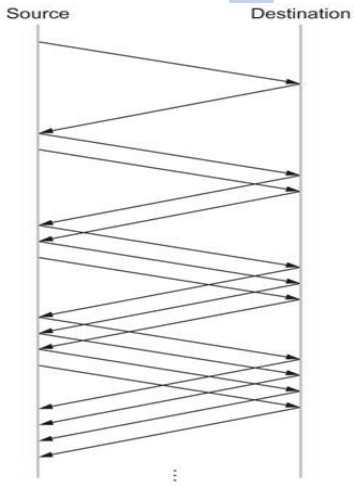
Additive Increase / Multiplicative Decrease (AIMD)

- TCP source *initializes* Congestion Window based on congestion level in the network. Source *increases* Congestion Window when level of congestion goes down and *decreases* the same when level of congestion goes up. TCP interprets *timeouts* as a sign of congestion and reduces the rate of

transmission. On timeout, source reduces its Congestion Window by half, i.e., *multiplicativedecrease*. For example, if Congestion Window = 16 packets, after timeout it is 8. Value of Congestion Window is never less than maximum segment size (MSS). When ACK arrives Congestion Window is *incremented* marginally, i.e., *additiveincrease.*

Increment = MSS × (MSS/Congestion Window) Congestion Window += Increment

For *example*, when ACK arrives for 1 packet, 2 packets are sent. When ACK for both packets arrive, 3 packets are sent and so on. Congestion Window increases and decreases throughout *lifetime* of the connection.

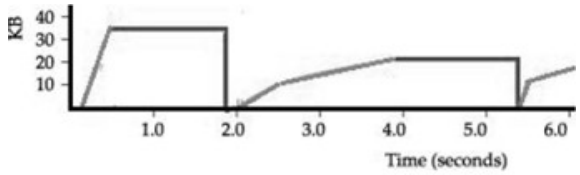


Slow Start

Slow start is used to increase Congestion Window *exponentially* from a cold start. Source TCP *initializes* Congestion Window to one packet. TCP *doubles* the number of packets sent every RTT on successful transmission. When ACK arrives for first packet TCP adds 1 packet to Congestion Window and sends two packets. When two ACKs arrive, TCP increments Congestion Window by 2 packets and sendsfour packets and so on. Instead of sending entire permissible packets at once (burst traffic), packets are sent in a phased manner, i.e., *slow start*. Initially TCP has no idea about congestion, henceforth it increasesCongestion Window rapidly until there is a timeout. On timeout:

CongestionThreshold = CongestionWindow/ 2CongestionWindow = 1

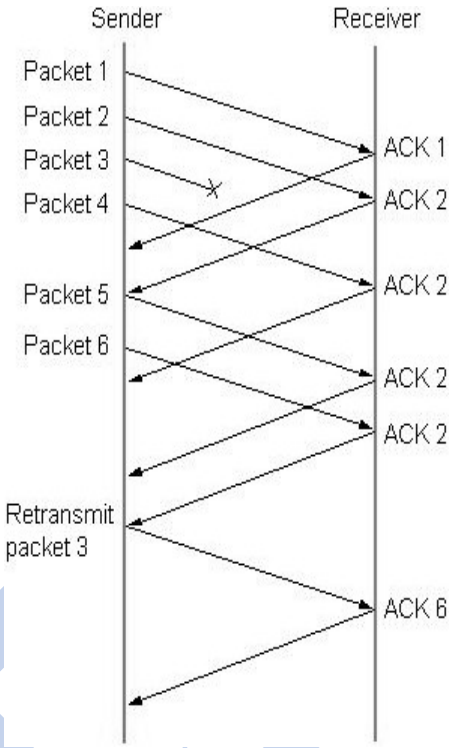
Slow start is repeated until Congestion Window reaches Congestion Threshold and thereafter 1 packet per RTT. The congestion window trace will look like



Congestion Window Trace

Fast Retransmit and Fast Recovery

TCP timeouts led to long periods of time during which the connection went dead while waiting for a timer to expire. Fast retransmit is a heuristic approach that *triggers* retransmission of a dropped packet sooner than the regular timeout mechanism. It *does not* replace regular timeouts. When a packet arrives out of order, receiving TCP resends the same acknowledgment (*duplicate ACK*) it sent last time. When *three duplicate* ACK arrives at the sender, it infers that corresponding packet may be lost due to congestion and retransmits that packet. This is called ***fast retransmit*** before regular timeout. When packet loss is detected using fast retransmit, the slow start phase is replaced by additive increase, multiplicative decrease method. This is known as ***fast recovery***. Instead of setting Congestion Window to one packet, this method uses the ACKs that are still in pipe to clock the sending of packets. Slow start is only used at the beginning of a connection and after *regular* timeout. At other times, it follows a pure AIMD pattern.



7. CONGESTION AVOIDANCE

Congestion avoidance mechanisms *prevent* congestion before it actually occurs. These mechanisms predict when congestion is about to happen and then to reduce the rate at which hosts send data just before packets start being discarded. TCP *creates* loss of packets in order to determine bandwidth of the connection. Routers *help* the end nodes by intimating when congestion is likely to occur. Congestion-avoidance mechanisms are:

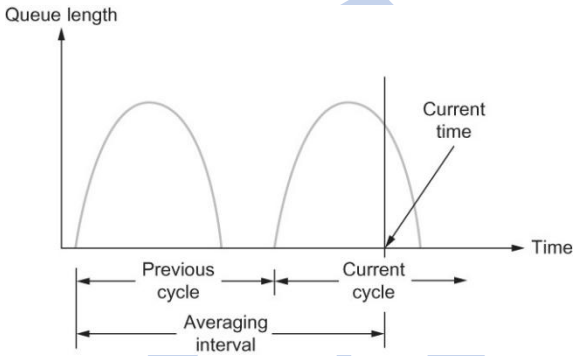
DEC bit - Destination Experiencing Congestion Bit

RED - Random Early Detection

Dec Bit - Destination Experiencing Congestion Bit

The first mechanism developed for use on the Digital Network Architecture(DNA). The idea is to evenly split the responsibility for congestion control between the routers and the end nodes. Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented by setting a binary congestion bit in the packets that flow through the router; hence the name DECbit.

The destination host then copies this congestion bit into the ACK it sends back to the source. The Source checks *how many* ACK has DEC bit set for previous window packets. If less than 50% of ACK have DEC bit set, then source *increases* its congestion window by 1 packet, Otherwise, *decreases* the congestion window by 87.5%. Finally, the source adjusts its sending rate so as to avoid congestion. *Increase by 1, decrease by 0.875* rule was based on AIMD for stabilization. A single congestion bit is added to the packet header. Using a queue length of 1 as the trigger for setting the congestion bit. A router sets this bit in a packet if its average queue length is greater than or equal to1 at the time the packet arrives.



Average queue length is measured over a time interval that includes the

***last busy* + *last idle cycle* + *current busy* cycle.**

It calculates the average queue length by *dividing* the curve area with time interval.

Red - Random Early Detection

The second mechanism of congestion avoidance is called as R*andom EarlyDetection (RED).* Each router is programmed to monitor its own queue length, and when it detects thatthere is congestion, it notifies the source to adjust its congestion window. RED differs from the DEC bit scheme by two ways: In DECbit, explicit notification about congestion is sent to source, whereas RED implicitly notifies the source by dropping a few packets.

DECbit may lead to tail drop policy, whereas RED drops packet based on drop probability in a random manner. Drop each arriving packet with some *drop probability* whenever the queue length exceeds some *drop level.* This idea is *called early random drop.*

Computation of average queue length using RED

$$Avg\ Len = (1 - Weight) \times Avg\ Len + Weight \times Sample\ Len$$

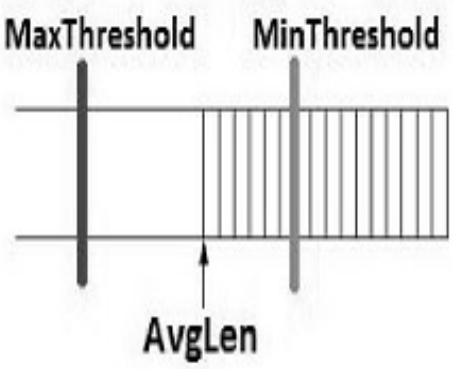
where $0 < Weight < 1$ and Sample Len – is the length

of the queue when a sample measurement is made. The queue length is measured every time a new packet arrives at the gateway.

if AvgLen ≤ MinThreshold
⟶ queue the packet

if MinThreshold < AvgLen < MaxThreshold
⟶ calculate probability P
⟶ drop the arriving packet with probability P

if AvgLen ≥ MaxThreshold
⟶ drop the arriving packet



8.STREAM CONTROL TRANSMISSION PROTOCOL

Stream Control Transmission Protocol (SCTP) is a reliable, message-oriented transport layer protocol. SCTP has mixed features of TCP and UDP. SCTP maintains the message boundaries and detects the lost data, duplicate data as well as out-of-order data. SCTP provides the Congestion control as well as Flow control. SCTP is especially designed for internet applications as well as multimediacommunication.

SCTP SERVICES

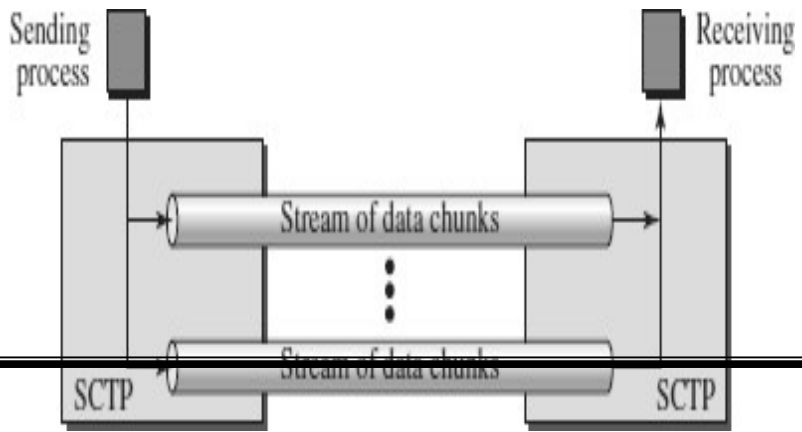
Process-to-Process Communication

SCTP provides process-to-process communication.

Multiple Streams

SCTP allows multistream service in each connection, which is called ***association*** inSCTP terminology.

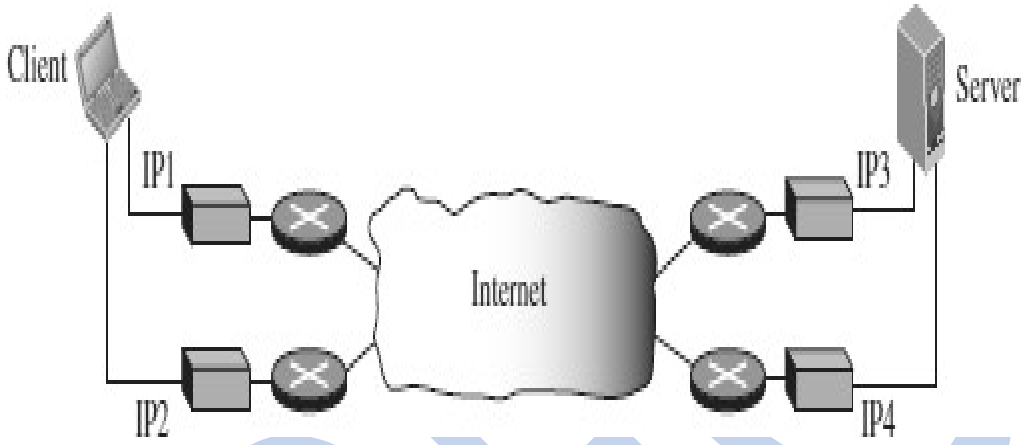
If one of the streams is blocked, the other streams can still deliver their



data.

Multihoming

An SCTP association supports multihoming service**.** 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.



Full-Duplex Communication: SCTP offers full-duplex service, where data can flow in both directions at the same time. Each SCTP then has a sending and receiving buffer and packets are sent in bothdirections.

Connection-Oriented Service:

SCTP is a connection-oriented protocol. In SCTP, a connection is called an association. If a client wants to send and receive message from server , the steps are :

Step1: The two SCTPs establish the connection with each other.

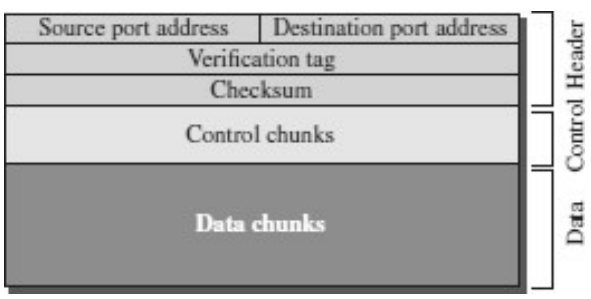
Step2: Once the connection is established, the data gets exchanged in both thedirections.

Step3: Finally, the association is terminated.

Reliable Service

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

SCTP PACKET FORMAT



An SCTP packet has a mandatory general header and a set of blocks called chunks.

General Header

Source port

Destination port

Verification tag

Checksum

Chunks

Types of Chunks

An SCTP association may send many packets, a packet may contain several chunks, and chunks may belong to different streams. SCTP defines two types of chunks - Control chunks and Data chunks. A control chunk controls and maintains the association. A data chunk carries user data.

Type	*Chunk*	*Description*
0	DATA	User data
1	INIT	Sets up an association
2	INIT ACK	Acknowledges INIT chunk
3	SACK	Selective acknowledgment
4	HEARTBEAT	Probes the peer for liveliness
5	HEARTBEAT ACK	Acknowledges HEARTBEAT chunk
6	ABORT	Aborts an association
7	SHUTDOWN	Terminates an association
8	SHUTDOWN ACK	Acknowledges SHUTDOWN chunk
9	ERROR	Reports errors without shutting down
10	COOKIE ECHO	Third packet in association establishment
11	COOKIE ACK	Acknowledges COOKIE ECHO chunk
14	SHUTDOWN COMPLETE	Third packet in association termination
192	FORWARD TSN	For adjusting cumulating TSN

SCTP ASSOCIATION

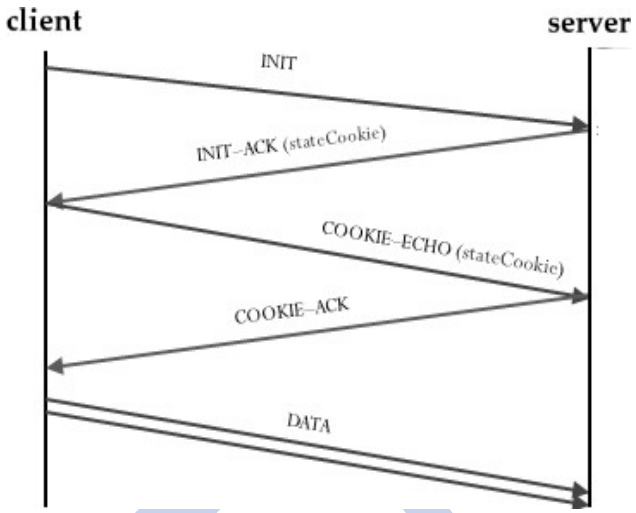
SCTP is a connection-oriented protocol. A connection in SCTP is called an association to emphasize multihoming. SCTP Associations consists of three phases:

- Association Establishment
- Data Transfer
- Association Termination

Association Establishment

Association establishment in SCTP requires a four-way handshake. In this procedure, a client process wants to establish an association with a server

process using SCTP as the transport-layer protocol. The SCTP server needs to be prepared to receive any association (passive open).



Association establishment, however, is initiated by the client (active open).

Data Transfer

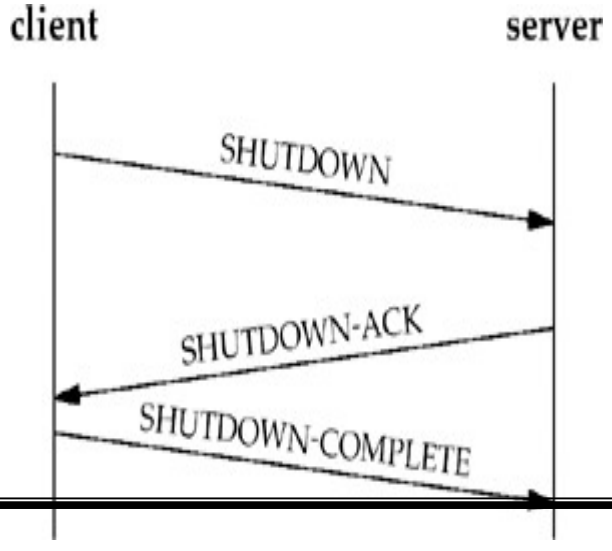
The whole purpose of an association is to transfer data between two ends. After the association is established, bidirectional data transfer can take place. The client and the server can both send data. SCTP supports piggybacking. Types of SCTP data Transfer :

Multihoming Data Transfer

Multistream Delivery

Association Termination

In SCTP,either of the two parties involved in exchanging data (client or server) can close the connection. SCTP does not allow a "half closed" association. If one end closes the association, the other end must stop sending new data. If any data are left over in the queue of the recipient of the termination request, they are sent and the association is closed. Association termination uses three packets.

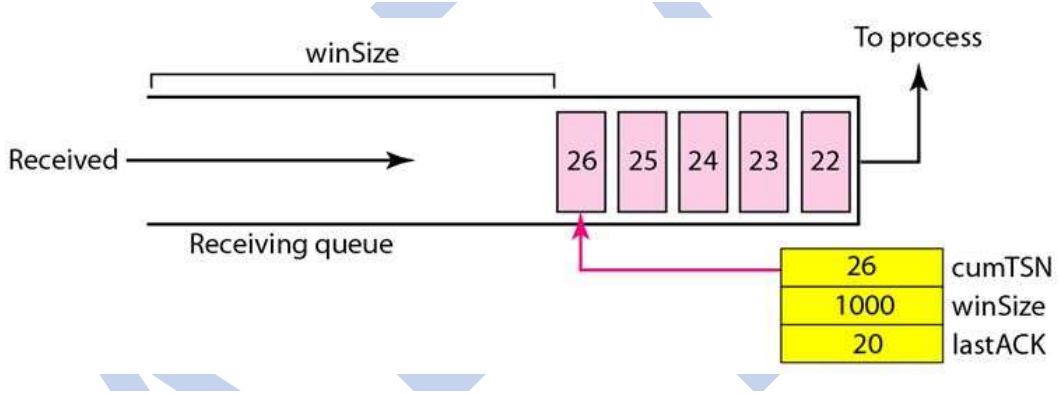


SCTP FLOW CONTROL

Flow control in SCTP is similar to that in TCP. In SCTP, we need to handle two units of data, the byte and the chunk. The values of rwnd and cwnd are expressed in bytes; the values of TSN and acknowledgments are expressed in chunks. Current SCTP implementations still use a byte-oriented window for flow control.

Receiver Site:

The receiver has one buffer (queue) and three variables. The queue holds the received data chunks that have not yet been read by the process. The first variable holds the last TSN received, cum TSN. The second variable holds the available buffer size; win size. The third variable holds the last accumulative acknowledgment, last ACK. The following figure shows the queue and variables at the receiver site.



When the site receives a data chunk, it stores it at the end of the buffer (queue) and subtracts the size of the chunk from win Size. The TSN number of the chunk is stored in the cum TSN variable. 2. When the process reads a chunk, it removes it from the queue and adds the size of the removed chunk to winSize (recycling).

3. When the receiver decides to send a SACK, it checks the value of lastAck; if it is less than cumTSN, it sends a SACK with a cumulative TSN number equal to the cumTSN. It also includes the value of win Size as the advertised window size.

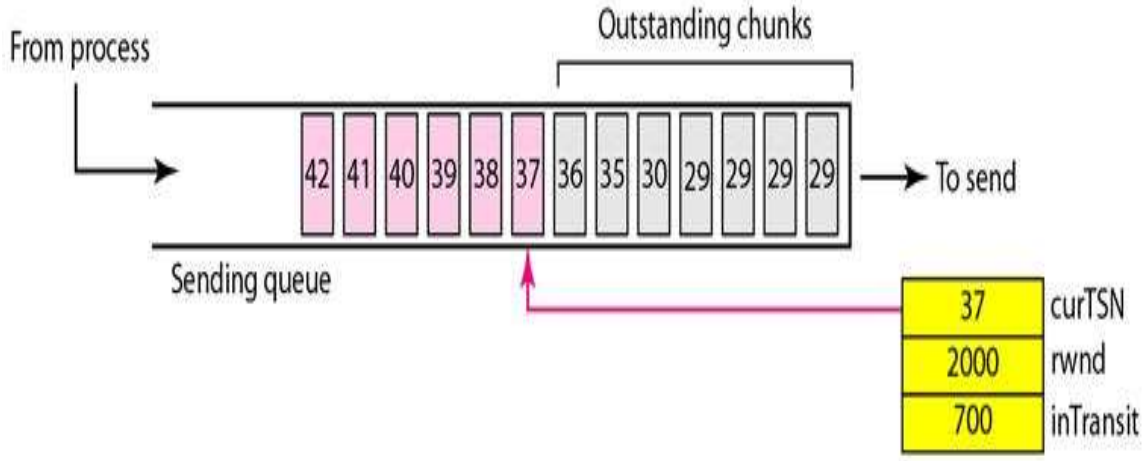
Sender Site:

The sender has one buffer (queue) and three variables: curTSN, rwnd, and in Transit, as shown in the following figure. We assume each chunk is 100 bytes long. The buffer holds the chunks produced by the process that either have been sent or are ready to be sent. The first variable, curTSN, refers to the next chunk to be sent. All chunks in the queue with a TSN less than this value have been sent, but not acknowledged; they are outstanding. The second variable, rwnd,

holds the last value advertised by the receiver (in bytes). The third variable, in Transit, holds the number of bytes in transit, bytes sent but not yet acknowledged. The following is the procedure used by the sender.

1. A chunk pointed to by curTSN can be sent if the size of the data is less than or equal to the quantity rwnd - inTransit. After sending the chunk, the value of curTSN is incremented by 1 and now points to the next chunk to be sent. The value of inTransit is incremented by the size of the data in the transmitted chunk.

2. When a SACK is received, the chunks with a TSN less than or equal to the cumulative TSN in the SACK are removed from the queue and discarded. The sender does not have to worry about them anymore. The value of inTransit is reduced by the total size of the discarded chunks. The value of rwnd is updated with the value of the advertised window in the SACK.



9. QUALITY OF SERVICE

Quality of service (QoS) is the use of mechanisms or technologies that work on a network to control traffic and ensure the performance of critical applications with limited network capacity. It enables organizations to adjust their overall network traffic by prioritizing specific high-performance applications. QoS is typically applied to networks that carry traffic for resource-intensive systems. Common services for which it is required include internet protocol television (IPTV), online gaming, streaming media, videoconferencing, video

on demand (VOD), and Voice over IP (VoIP). Using QoS in networking, organizations have the ability to optimize the performance of multiple applications on their network and gain visibility into the bit rate, delay, jitter, and packet rate of their network. This ensures they can engineer the traffic on their network and change the way that packets are routed to the internet or other networks to avoid transmission delay. This also ensures that the organization achieves the expected service quality for applications and delivers expected user experiences. As per the QoS meaning, the key goal is to enable networks and organizations to prioritize traffic, which includes offering dedicated bandwidth, controlled jitter, and lower latency. The technologies used to ensure this are vital to enhancing the performance of business applications, wide-area networks (WANs), and service provider networks. QoS networking technology works by marking packets to identify service types, then configuring routers to create separate virtual queues for each application, based on their priority. As a result, bandwidth is reserved for critical applications or websites that have been assigned priority access. QoS technologies provide capacity and handling allocation to specific flows in network traffic. This enables the network administrator to assign the order in which packets are handled and provide the appropriate amount of bandwidth to each application or traffic flow.

Types of Network Traffic

Understanding how QoS network software works is reliant on defining the various types of traffic that it measures. These are:

Bandwidth: The speed of a link. QoS can tell a router how to use bandwidth. For example, assigning a certain amount of bandwidth to different queues for different traffic types.

Delay: The time it takes for a packet to go from its source to its end destination. This can often be affected by queuing delay, which occurs during times of congestion and a packet waits in a queue before being transmitted. QoS enables organizations to avoid this by creating a priority queue for certain types of traffic.

Loss: The amount of data lost as a result of packet loss, which typically occurs due to network congestion. QoS enables organizations to decide which packets to drop in this event.

Jitter: The irregular speed of packets on a network as a result of congestion, which can result in packets arriving late and out of sequence. This can cause distortion or gaps in audio and video being delivered.

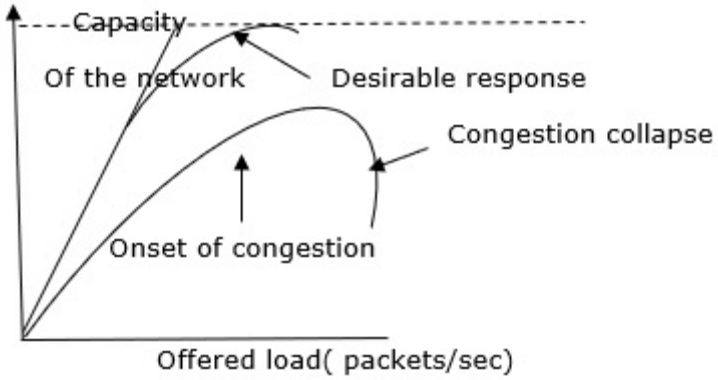


Token bucket algorithm

Token bucket algorithm is one of the techniques for congestion control algorithms. When too many packets are present in the network it causes packet delay and loss of packet which degrades the performance of the system. This situation is called congestion.

The network layer and transport layer share the responsibility for handling congestions. One of the most effective ways to control congestion is trying to reduce the load that transport layer is placing on the network. To maintain this network and transport layers have to work together.

The Token Bucket Algorithm is diagrammatically represented as follows –



The leaky bucket algorithm enforces output patterns at the average rate, no matter how busy the traffic is. So, to deal with the more traffic, we need a flexible algorithm so that the data is not lost. One such approach is the token bucket algorithm.

Let us understand this algorithm step wise as given below −

- **Step 1** − In regular intervals tokens are thrown into the bucket f.
- **Step 2** − The bucket has a maximum capacity f.
- **Step 3** − If the packet is ready, then a token is removed from the bucket, and the packet is sent.
- **Step 4** − Suppose, if there is no token in the bucket, the packet cannot be sent.

Leaky bucket algorithm

Leaky Bucket Algorithm mainly controls the total amount and the rate of the traffic sent to the network.

Step 1 − Let us imagine a bucket with a small hole at the bottom where the rate at which water is poured into the bucket is not constant and can vary but it leaks from the bucket at a constant rate.

Step 2 − So (up to water is present 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.

Step 3 − If the bucket is full, additional water that enters into the bucket that spills over the sides and is lost.

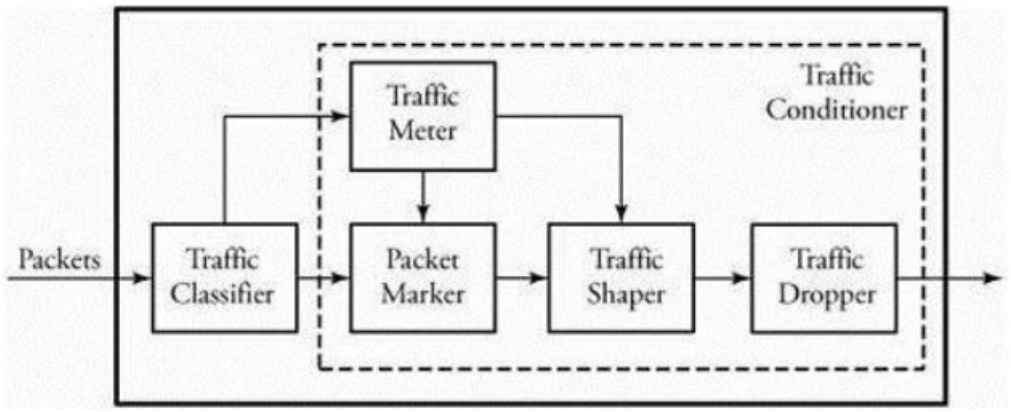
Step 4 − Thus the same concept applied to packets in the network. Consider that data is coming from the source at variable speeds. Suppose that a source sends data at 10 Mbps for 4 seconds. Then there is no data for 3 seconds. The

source again transmits data at a rate of 8 Mbps for 2 seconds. Thus, in a time span of 8 seconds, 68 Mb data has been transmitted.

That's why if a leaky bucket algorithm is used, the data flow would be 8 Mbps for 9 seconds. Thus, the constant flow is maintained.

Differentiated QoS

The differentiated services (DS), or DiffServ, approach provides a simpler and more scalable QoS. DS minimizes the amount of storage needed in a router by processing traffic flows in an aggregate manner, moving all the complex procedures from the core to the edge of the network. A traffic conditioner is one of the main features of a DiffServ node to protect the DiffServ domain. As shown in Figure 6.5, the traffic conditioner includes four major components: meter, marker, shaper, and dropper.



A meter measures the traffic to make sure that packets do not exceed their traffic profiles. A marker marks or unmarks packets in order to keep track of their situations in the DS node. A shaper delays any packet that is not compliant with the traffic profile. Finally, a dropper discards any packet that violates its traffic profile.

Advantages of QoS

The deployment of QoS is crucial for businesses that want to ensure the availability of their business-critical applications. It is vital for delivering differentiated bandwidth and ensuring data transmission takes place without interrupting traffic flow or causing packet losses. Major advantages of deploying QoS include:

1. **Unlimited application prioritization:** QoS guarantees that businesses' most mission-critical applications will always have priority and the necessary resources to achieve high performance.
2. **Better resource management:** QoS enables administrators to better manage the organization's internet resources. This also reduces costs and the need for investments in link expansions.
3. **Enhanced user experience:** The end goal of QoS is to guarantee the high performance of critical applications, which boils down to delivering optimal user experience. Employees enjoy high performance on their high-bandwidth applications, which enables them to be more effective and get their job done more quickly.
4. **Point-to-point traffic management:** Managing a network is vital however traffic is delivered, be it end to end, node to node, or point to point. The latter enables organizations to deliver customer packets in order from one point to the next over the internet without suffering any packet loss.
5. **Packet loss prevention:** Packet loss can occur when packets of data are dropped in transit between networks. This can often be caused by a failure or inefficiency, network congestion, a faulty router, loose connection, or poor signal. QoS avoids the potential of packet loss by prioritizing bandwidth of high-performance applications.
6. **Latency reduction:** Latency is the time it takes for a network request to go from the sender to the receiver and for the receiver to process it. This is typically affected by routers taking longer to analyse information and storage delays caused by intermediate switches and bridges. QoS enables organizations to reduce latency, or speed up the process of a network request, by prioritizing their critical application.