**Principles Computer Networking (CST102) Services of TCP layer**

###### The services provided by the transport layer protocols can be divided into five categories:

* End-to-end delivery
* Addressing
* Reliable delivery
* Flow control
* Multiplexing

## End-to-end delivery:

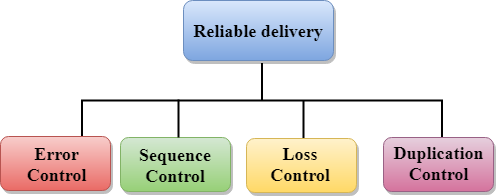
The transport layer transmits the entire message to the destination. Therefore, it ensures the end-to-end delivery of an entire message from a source to the destination.

## Reliable delivery:

The transport layer provides reliability services by retransmitting the lost and damaged packets.

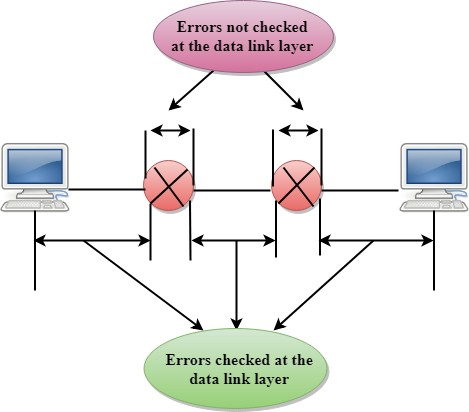
###### The reliable delivery has four aspects:

* Error control
* Sequence control
* Loss control
* Duplication control



###### Error Control

* The primary role of reliability is **Error Control**. In reality, no transmission will be 100 percent error-free delivery. Therefore, transport layer protocols are designed to provide error-free transmission.
* The data link layer also provides the error handling mechanism, but it ensures only node-to-node error-free delivery. However, node-to-node reliability does not ensure the end-to-end reliability.
* The data link layer checks for the error between each network. If an error is introduced inside one of the routers, then this error will not be caught by the data link layer. It only detects those errors that have been introduced between the beginning and end of the link. Therefore, the transport layer performs the checking for the errors end-to-end to ensure that the packet has arrived correctly.



###### Sequence Control

* The second aspect of the reliability is sequence control which is implemented at the transport layer.
* On the sending end, the transport layer is responsible for ensuring that the packets received from the upper layers can be used by the lower layers. On the receiving end, it ensures that the various segments of a transmission can be correctly reassembled.

###### Loss Control

Loss Control is a third aspect of reliability. The transport layer ensures that all the fragments of a transmission arrive at the destination, not some of them. On the sending end, all the fragments of transmission are given sequence numbers by a transport layer. These sequence numbers allow the receiver?s transport layer to identify the missing segment.

###### Duplication Control

Duplication Control is the fourth aspect of reliability. The transport layer guarantees that no duplicate data arrive at the destination. Sequence numbers are used to identify the lost packets; similarly, it allows the receiver to identify and discard duplicate segments.

## Flow Control

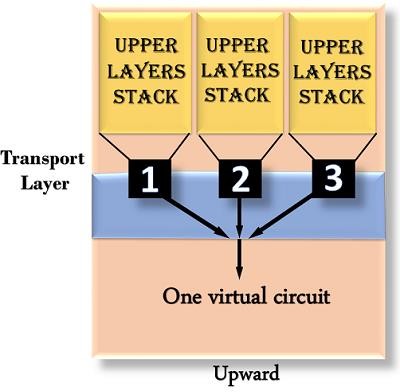
Flow control is used to prevent the sender from overwhelming the receiver. If the receiver is overloaded with too much data, then the receiver discards the packets and asking for the retransmission of packets. This increases network congestion and thus, reducing the system performance. The transport layer is responsible for flow control. It uses the sliding window protocol that makes the data transmission more efficient as well as it controls the flow of data so that the receiver does not become overwhelmed. Sliding window protocol is byte oriented rather than frame oriented.

## Multiplexing

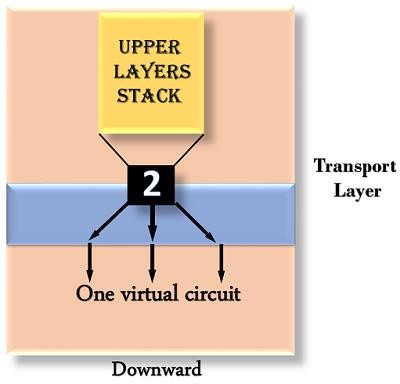
The transport layer uses the multiplexing to improve transmission efficiency.

###### Multiplexing can occur in two ways:

* **Upward multiplexing:** Upward multiplexing means multiple transport layer connections use the same network connection. To make more cost-effective, the transport layer sends several transmissions bound for the same destination along the same path; this is achieved through upward multiplexing.

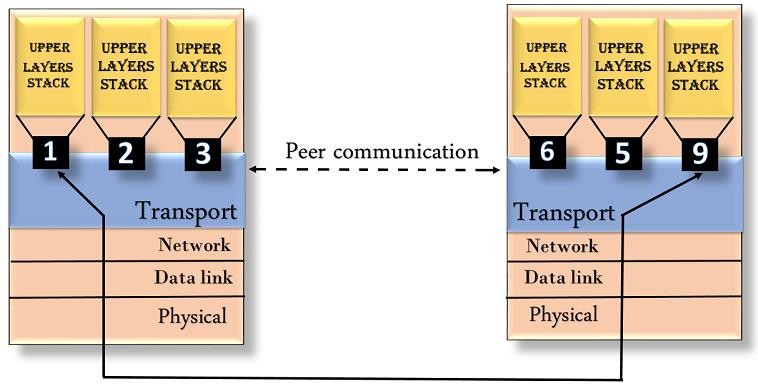


* **Downward multiplexing:** Downward multiplexing means one transport layer connection uses the multiple network connections. Downward multiplexing allows the transport layer to split a connection among several paths to improve the throughput. This type of multiplexing is used when networks have a low or slow capacity.



## Addressing

* According to the layered model, the transport layer interacts with the functions of the session layer. Many protocols combine session, presentation, and application layer protocols into a single layer known as the application layer. In these cases, delivery to the session layer means the delivery to the application layer. Data generated by an application on one machine must be transmitted to the correct application on another machine. In this case, addressing is provided by the transport layer.
* The transport layer provides the user address which is specified as a station or port. The port variable represents a particular TS user of a specified station known as a Transport Service access point (TSAP). Each station has only one transport entity.
* The transport layer protocols need to know which upper-layer protocols are communicating.



## TCPTCP Header format

* **Source port:** It defines the port of the application, which is sending the data. So, this field contains the source port address, which is 16 bits.
* **Destination port:** It defines the port of the application on the receiving side. So, this field contains the destination port address, which is 16 bits.
* **Sequence number:** This field contains the sequence number of data bytes in a particular session.
* **Acknowledgment number:** When the ACK flag is set, then this contains the next sequence number of the data byte and works as an acknowledgment for the previous data received. For example, if the receiver receives the segment number 'x', then it responds 'x+1' as an acknowledgment number.
* **HLEN:** It specifies the length of the header indicated by the 4-byte words in the header. The size of the header lies between 20 and 60 bytes. Therefore, the value of this field would lie between 5 and 15.
* **Reserved:** It is a 4-bit field reserved for future use, and by default, all are set to zero.

###### Flags

**There are six control bits or flags:**

* 1. **URG:** It represents an urgent pointer. If it is set, then the data is processed urgently.
  2. **ACK:** If the ACK is set to 0, then it means that the data packet does not contain an acknowledgment.
  3. **PSH:** If this field is set, then it requests the receiving device to push the data to the receiving application without buffering it.
  4. **RST:** If it is set, then it requests to restart a connection.
  5. **SYN:** It is used to establish a connection between the hosts.
  6. **FIN:** It is used to release a connection, and no further data exchange will happen.

###### Windowsize

It is a 16-bit field. It contains the size of data that the receiver can accept. This field is used for the flow control between the sender and receiver and also determines the amount of buffer allocated by the receiver for a segment. The value of this field is determined by the receiver.

###### Checksum

It is a 16-bit field. This field is optional in UDP, but in the case of TCP/IP, this field is mandatory.

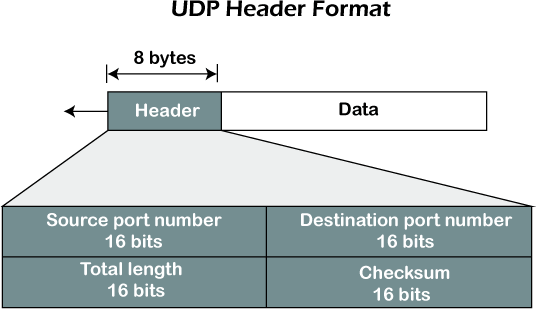
###### Urgentpointer

It is a pointer that points to the urgent data byte if the URG flag is set to 1. It defines a value that will be added to the sequence number to get the sequence number of the last urgent byte.

###### Options

It provides additional options. The optional field is represented in 32-bits. If this field contains the data less than 32-bit, then padding is required to obtain the remaining bits.

## UDP Header Format

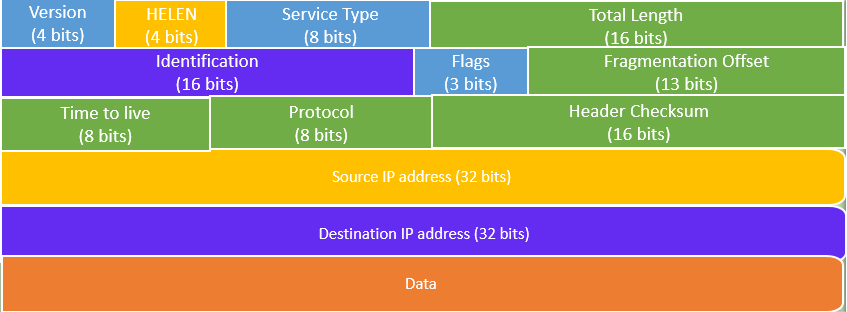


In UDP, the header size is 8 bytes, and the packet size is upto 65,535 bytes. But this packet size is not possible as the data needs to be encapsulated in the IP datagram, and an IP packet, the header size can be 20 bytes; therefore, the maximum of UDP would be 65,535 minus 20. The size of the data that the UDP packet can carry would be 65,535 minus 28 as 8 bytes for the header of the UDP packet and 20 bytes for IP header.

###### The UDP header contains four fields:

* **Source port number:** It is 16-bit information that identifies which port is going t send the packet.
* **Destination port number:** It identifies which port is going to accept the information. It is 16-bit information which is used to identify application-level service on the destination machine.
* **Length:** It is 16-bit field that specifies the entire length of the UDP packet that includes the header also. The minimum value would be 8-byte as the size of the header is 8 bytes.
* **Checksum:** It is a 16-bits field, and it is an optional field. This checksum field checks whether the information is accurate or not as there is the possibility that the information can be corrupted while transmission. It is an optional field, which means that it depends upon the application, whether it wants to write the checksum or not. If it does not want to write the checksum, then all the 16 bits are zero; otherwise, it writes the checksum. In UDP, the checksum field is applied to the entire packet, i.e., header as well as data part whereas, in IP, the checksum field is applied to only the header field.

**IPv4 Header Components/Fields**

IP

header format

#### Following are various components/fields of IP packet header

* **Version:** The first IP header field is a 4-bit version indicator. In IPv4, the value of its four bits is set to 0100, which indicates 4 in binary. However, if the router does not support the specified version, this packet will be dropped.
* **Internet Header Length:** Internet header length, shortly known as IHL, is 4 bits in size. It is also called HELEN (Header Length). This IP component is used to show how many 32-bit words are present in the header.
* **Type of Service:** Type of Service is also called Differentiated Services Code Point or DSCP. This field is provided features related to the quality of service for data streaming or VoIP calls. The first 3 bits are the priority bits. It is also used for specifying how you can handle Datagram.
* **Total length:** The total length is measured in bytes. The minimum size of an IP datagram is 20 bytes and the maximum, it can be 65535 bytes . HELEN and Total length can be used to calculate the dimension of the payload. All hosts are required to be able to read 576-byte datagrams. However, if a datagram is too large for the hosts in the network, the fragmentation method is widely used.
* **Identification:** Identification is a packet that is used to identify fragments of an IP datagram uniquely. Some have recommended using this field for other things like adding information for packet tracing, etc.
* **IP Flags:** Flag is a three-bit field that helps you to control and identify fragments. The following can be their possible configuration:

Bit 0: is reserved and has to be set to zero Bit 1: means do not fragment

Bit 2: means more fragments.

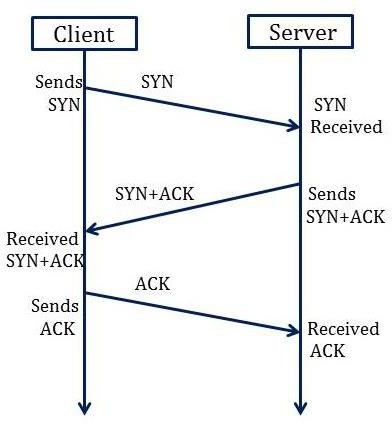
* **Fragment Offset:** Fragment Offset represents the number of Data Bytes ahead of the particular fragment in the specific Datagram. It is specified in terms of the number of 8 bytes, which has a maximum value of 65,528 bytes.
* **Time to live:** It is an 8-bit field that indicates the maximum time the Datagram will be live in the internet system. The time duration is measured in seconds, and when the value of TTL is zero, the Datagram will be erased. Every time a datagram is processed its TTL value is decreased by one second. TTL are used so that datagrams are not delivered and discarded automatically. The value of TTL can be 0 to 255.
* **Protocol:** This IPv4 header is reserved to denote that internet protocol is used in the latter portion of the Datagram. For Example, 6 number digit is mostly used to indicate TCP, and 17 is used to denote the UDP protocol.
* **Header Checksum:** The next component is a 16 bits header checksum field, which is used to check the header for any errors. The IP header is compared to the value of its checksum. When the header checksum is not matching, then the packet will be discarded.
* **Source Address:** The source address is a 32-bit address of the source used for the IPv4 packet.
* **Destination address:** The destination address is also 32 bit in size stores the address of the receiver.
* **IP Options:** It is an optional field of IPv4 header used when the value of IHL (Internet Header Length) is set to greater than 5. It contains values and settings related with security, record route and time stamp, etc. You can see that list of options component ends with an End of Options or EOL in most cases.
* **Data:** This field stores the data from the protocol layer, which has handed over the data to the IP layer.

**3-Way Handshake**

Transmission Control Protocol (TCP) provides a secure and reliable connection between two devices using the 3-way handshake process. TCP uses the full-duplex connection to synchronize (SYN) and acknowledge (ACK) each other on both sides. There are three steps for both establishing and closing a connection. They are − SYN, SYN-ACK, and ACK.

# Way Handshake Connection Establishment Process

The following diagram shows how a reliable connection is established using 3-way handshake. It will support communication between a web browser on the client and server sides whenever a user navigates the Internet.



### Synchronization Sequence Number (SYN) − The client sends the SYN to the server

* + When the client wants to connect to the server, then it sends the message to the server by setting the SYN flag as 1.
  + The message carries some additional information like the sequence number (32-bit random number).
  + The ACK is set to 0. The maximum segment size and the window size are also set. For example, if the window size is 1000 bits and the maximum segment size is 100 bits, then a maximum of 10 data segments can be transmitted in the connection by dividing (1000/100=10).

### Synchronization and Acknowledgement (SYN-ACK) to the client

* + The server acknowledges the client request by setting the ACK flag to 1.
  + The ACK indicates the response of the segment it received and SYN indicates with what sequence number it will start the segments.
  + For example, if the client has sent the SYN with sequence number = 500, then the server will send the ACK using acknowledgment number = 5001.
  + The server will set the SYN flag to '1' and send it to the client if the server also wants to establish the connection.
  + The sequence number used for SYN will be different from the client's SYN.
  + The server also advertises its window size and maximum segment size to the client. And, the connection is established from the client-side to the server-side.

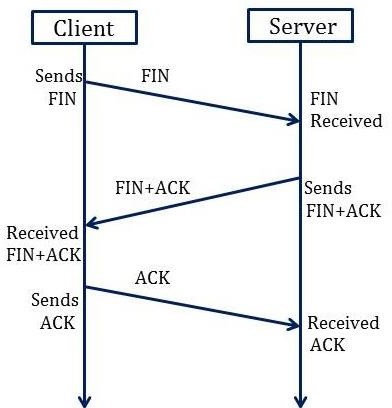
### Acknowledgment (ACK) to the server

* + The client sends the acknowledgment (ACK) to the server after receiving the synchronization (SYN) from the server.
  + After getting the (ACK) from the client, the connection is established between the client and the server.
  + Now the data can be transmitted between the client and server sides.

### 3 -Way Handshake Closing Connection Process

To close a 3-way handshake connection,

* First, the client requests the server to terminate the established connection by sending FIN.
* After receiving the client request, the server sends back the FIN and ACK request to the client.
* After receiving the FIN + ACK from the server, the client confirms by sending an ACK to the server.



# UDP application

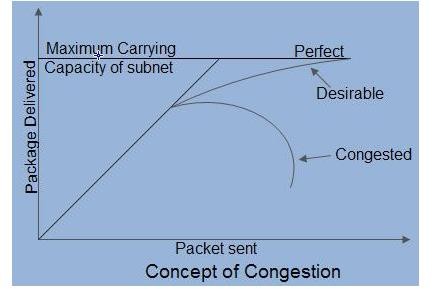
Here are few applications where UDP is used to transmit data:

* Domain Name Services
* Simple Network Management Protocol
* Trivial File Transfer Protocol
* Routing Information Protocol
* Kerberos

**Congestion Control**

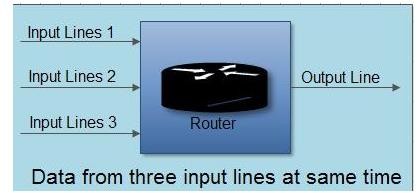
Congestion is an important issue that can arise in packet switched network. Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance degrades. Congestion in a network may occur when the load on the network *(i.e.* the number of packets sent to the network) is greater than the capacity of the network *(i.e.* the number of packets a network can handle.). Network congestion occurs in case of traffic overloading.

In other words when too much traffic is offered, congestion sets in and performance degrades sharply



**Causing of Congestion:**

The various causes of congestion in a subnet are:

* The input traffic rate exceeds the capacity of the output lines. If suddenly, a stream of packet start arriving on three or four input lines and all need the same output line. In this case, a queue will be built up. If there is insufficient [memory](https://ecomputernotes.com/fundamental/input-output-and-memory/memory) to hold all the packets, the packet will be lost. Increasing the [memory](https://ecomputernotes.com/fundamental/input-output-and-memory/memory) to unlimited size does not solve the problem. This is because, by the time packets reach front of the queue, they have already timed out (as they waited the queue). When timer goes off source transmits duplicate packet that are also added to the queue. Thus same packets are added again and again, increasing the load all the way to the destination.

The routers are too slow to perform bookkeeping tasks (queuing buffers, updating tables, etc.).

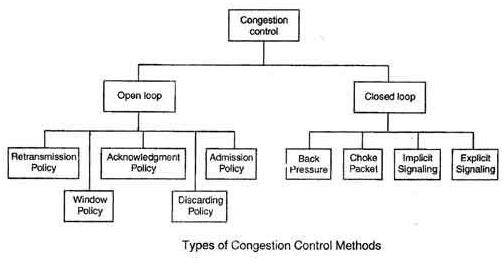
* The routers’ buffer is too limited.
* Congestion in a subnet can occur if the processors are slow. Slow speed [CPU](https://ecomputernotes.com/fundamental/introduction-to-computer/what-is-cpu) at routers will perform the routine tasks such as queuing buffers, updating table etc slowly. As a result of this, queues are built up even though there is excess line capacity.
* Congestion is also caused by slow links. This problem will be solved when high speed links are used. But it is not always the case. Sometimes increase in link bandwidth can further deteriorate the congestion problem as higher speed links may make the network more unbalanced.Congestion can make itself worse. If a route!” does not have free buffers, it start ignoring/discarding the newly arriving packets. When these packets are discarded, the sender may retransmit them after the timer goes off. Such packets are transmitted by the sender again and again until the source gets the acknowledgement of these packets. Therefore multiple transmissions of packets will force the congestion to take place at the sending end.

Congestion Control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. Congestion control mechanisms are divided into two categories, one category

prevents the congestion from happening and the other category removes congestion after it has taken place.

##### How to correct the Congestion Problem:

Congestion Control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. Congestion control mechanisms are divided into two categories, one category prevents the congestion from happening and the other category removes congestion after it has taken place.



##### Open Loop Congestion Control

* In this method, policies are used to prevent the congestion before it happens.
* Congestion control is handled either by the source or by the destination.
* The various methods used for open loop congestion control are:

##### Retransmission Policy

* The sender retransmits a packet, if it feels that the packet it has sent is lost or corrupted.
* However retransmission in general may increase the congestion in the network. But we need to implement good retransmission policy to prevent congestion.
* The retransmission policy and the retransmission timers need to be designed to optimize efficiency and at the same time prevent the congestion.

##### Window Policy

* To implement window policy, selective reject window method is used for congestion control.
* Selective Reject method is preferred over Go-back-n window as in Go-back-n method, when timer for a packet times out, several packets are resent, although some may have arrived safely at the receiver. Thus, this duplication may make congestion worse.
* Selective reject method sends only the specific lost or damaged packets.

##### Acknowledgement Policy

* The acknowledgement 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.
* Acknowledgments also add to the traffic load on the network. Thus, by sending fewer acknowledgements we can reduce load on the network.
* To implement it, several approaches can be used:

1. A receiver may send an acknowledgement only if it has a packet to be sent.
2. A receiver may send an acknowledgement when a timer expires.
3. A receiver may also decide to acknowledge only *N* packets at a time.

##### Discarding Policy

* A router may discard less sensitive packets when congestion is likely to happen.
* Such a discarding policy may prevent congestion and at the same time may not harm the integrity of the transmission.

##### Admission Policy

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

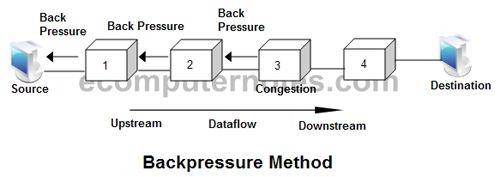
These two categories are:

1. Open loop
2. Closed loop

##### ClosedLoopCongestionControl

* Closed loop congestion control mechanisms try to remove the congestion after it happens.
* The various methods used for closed loop congestion control are:

##### Backpressure

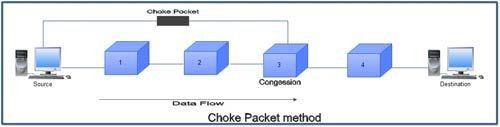
* Back pressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow.
  + The

backpressure technique can be applied only to virtual circuit networks. In such virtual circuit each node knows the upstream node from which a data flow is coming.

* In this method of congestion control, the congested node stops receiving data from the immediate upstream node or nodes.
* This may cause the upstream node on nodes to become congested, and they, in turn, reject data from their upstream node or nodes.
* As shown in fig node 3 is congested and it stops receiving packets and informs its upstream node 2 to slow down. Node 2 in turns may be congested and informs node 1 to slow down. Now node 1 may create congestion and informs the source node to slow down. In this way the congestion is alleviated. Thus, the pressure on node 3 is moved backward to the source to remove the congestion.

##### ChokePacket

* In this method of congestion control, congested router or node sends a special type of packet called choke packet to the source to inform it about the congestion.
* Here, congested node does not inform its upstream node about the congestion as in backpressure method.
* In choke packet method, congested node sends a warning directly to the source station *i.e.* the intermediate nodes through which the packet has traveled are not warned.



##### Implicit Signaling

* In implicit signaling, there is no communication between the congested node or nodes and the source.
* The source guesses that there is congestion somewhere in the network when it does not receive any acknowledgment. Therefore the delay in receiving an acknowledgment is interpreted as congestion in the network.
* On sensing this congestion, the source slows down.
* This type of congestion control policy is used by TCP.

##### Explicit Signaling

* In this method, the congested nodes explicitly send a signal to the source or destination to inform about the congestion.
* Explicit signaling is different from the choke packet method. In choke packed method, a separate packet is used for this purpose whereas in explicit signaling method, the signal is included in the packets that carry data .
* Explicit signaling can occur in either the forward direction or the backward direction .
* In backward signaling, a bit is set in a packet moving in the direction opposite to the congestion. This bit warns the source about the congestion and informs the source to slow down.
* In forward signaling, a bit is set in a packet moving in the direction of congestion. This bit warns the destination about the congestion. The receiver in this case uses policies such as slowing down the acknowledgements to remove the congestion.

**Leaky bucket algorithm**

In the network layer, before the network can make Quality of service guarantees, it must know what traffic is being guaranteed. One of the main causes of congestion is that traffic is often bursty.

To understand this concept first we have to know little about traffic shaping. **Traffic Shaping** is a mechanism to control the amount and the rate of traffic sent to the network. Approach of congestion management is called Traffic shaping. Traffic shaping helps to regulate the rate of data transmission and reduces congestion.

There are 2 types of traffic shaping algorithms:

1. Leaky Bucket
2. Token Bucket

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.

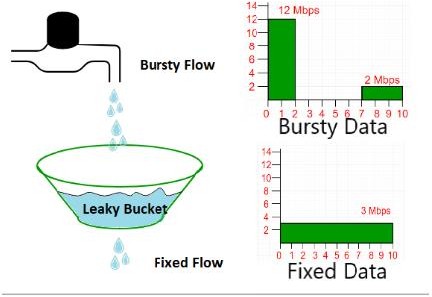
**Step 3** − If the bucket is full, additional water that enters into the bucket that spills over the sides and is lost.

**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.

Suppose we have a bucket in which we are pouring water, at random points in time, but we have to get water at a fixed rate, to achieve this we will make a hole at the bottom of the bucket. This will ensure that the water coming out is at some fixed rate, and also if the bucket gets full, then we will stop pouring water into it.

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.



In the above figure, we 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. In the above figure, the host sends a burst of data at a rate of 12 Mbps for 2s, for a total of 24 Mbits 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 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s. The leaky bucket smooths out the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

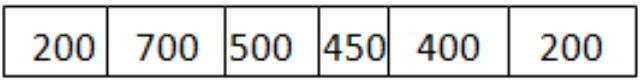
A simple leaky bucket algorithm can be implemented using FIFO queue. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. Repeat until n is smaller than the packet size of the packet at the head of the queue.
   1. Pop a packet out of the head of the queue, say P.
   2. Send the packet P, into the network
   3. Decrement the counter by the size of packet P.
3. Reset the counter and go to step 1.

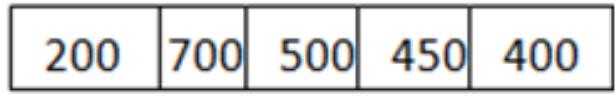
***Note:*** *In the below examples, the head of the queue is the rightmost position and the tail of the queue is the leftmost position.*

**Example:** Let n=1000 Packet=



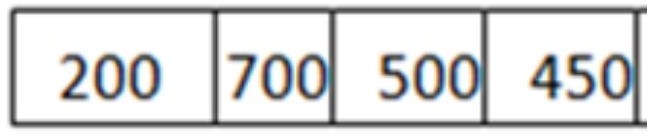
Since n > size of the packet at the head of the Queue, i.e. n > 200 Therefore, n = 1000-200 = 800

Packet size of 200 is sent into the network.



Now, again n > size of the packet at the head of the Queue, i.e. n > 400 Therefore, n = 800-400 = 400

Packet size of 400 is sent into the network.



Since, n < size of the packet at the head of the Queue, i.e. n < 450 Therefore, the procedure is stopped.

Initialise n = 1000 on another tick of the clock.

This procedure is repeated until all the packets are sent into the network. Below is the implementation of above explained approach:

##### Output

Buffer size= 4 out of bucket size= 10 Buffer size= 7 out of bucket size= 10 Buffer size= 10 out of bucket size= 10 Packet loss = 4

Buffer size= 9 out of bucket size= 10

##### Difference between Leaky and Token buckets –

|  |  |
| --- | --- |
| Leaky Bucket | Token Bucket |
| When the host has to send a packet , packet is thrown in bucket. | In this, the bucket holds tokens generated at regular intervals of time. |
| Bucket leaks at constant rate | Bucket has maximum capacity. |
| Bursty traffic is converted into uniform traffic by leaky bucket. | If there is a ready packet , a token is removed from Bucket and packet is send. |
| In practice bucket is a finite queue outputs at finite rate | If there is no token in the bucket, then the packet cannot be sent. |

**Some advantage of token Bucket over leaky bucket**

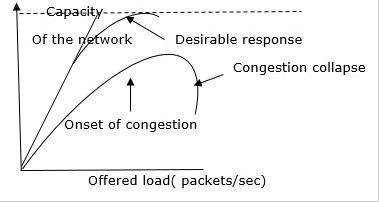
* If a bucket is full in tokens Bucket, tokens are discard not packets. While in leaky bucket, packets are discarded.
* Token Bucket can send large bursts at a faster rate while leaky bucket always sends packets at constant rate.

##### TokenbucketAlgorithm

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 −



With too much traffic, performance drops sharply.

# Token Bucket Algorithm

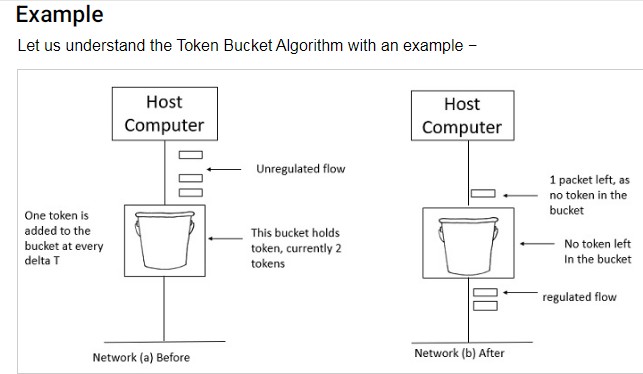
The leaky bucket algorithm enforces output patterns at the average rate, no matter how busy the traffic is. So, to deal with the more traffic, we need a flexible algorithm so that the data is not lost. One such approach is the token bucket algorithm.

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.

# Example

Let us understand the Token Bucket Algorithm with an example −



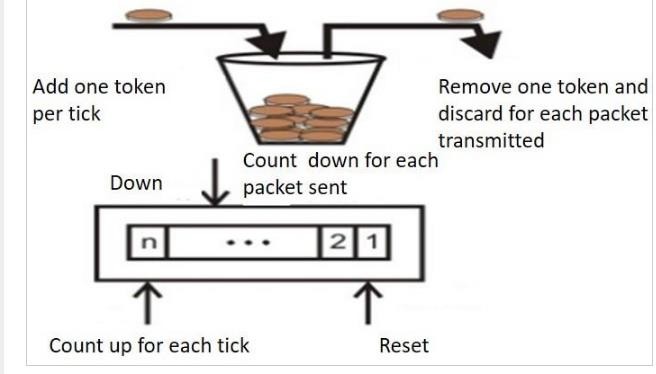
In figure (a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface.

In Figure (b) two packets have been sent out by consuming two tokens, and 1 packet is still left.

When compared to Leaky bucket the token bucket algorithm is less restrictive that means it allows more traffic. The limit of busyness is restricted by the number of tokens available in the bucket at a particular instant of time.

The implementation of the token bucket algorithm is easy − a variable is used to count the tokens. For every t seconds the counter is incremented and then it is decremented whenever a packet is sent. When the counter reaches zero, no further packet is sent out.

This is shown in below given diagram −



Dialog Control – Session layer allows two systems to enter into a dialog exchange mechanism which can either be full or half-duplex.

* 1. Managing Tokens –

The communicating systems in a network try to perform some critical operations and it is Session Layer

which prevents collisions which might occur while performing these operations which would otherwise result in a loss.

* 1. Synchronization –

Checkpoints are the midway marks that are added after a particular interval during stream of data

transfer. These points are also referred to as synchronization points. The Session layer permits process to add these checkpoints.

For example, suppose a file of 400 pages is being sent over a network, then it is highly beneficial to set up a checkpoint after every 50 pages so that next 50 pages are sent only when previous pages are received and acknowledged.

Design Issues with Session Layer :

1. Establish sessions between machines –

The establishment of session between machines is an important service provided by session layer. This session is responsible for creating a dialog between connected machines. The Session Layer provides mechanism for opening, closing and managing a session between end-user application processes, i.e. a semi- permanent dialogue. This session consists of requests and responses that occur between applications.

1. Enhanced Services –

Certain services such as checkpoints and management of tokens are the key features of session layer and thus it becomes necessary to keep enhancing these features during the layer’s design.

1. To help in Token management and Synchronization –

The session layer plays an important role in preventing collision of several critical operation as well as ensuring better data transfer over network by establishing synchronization points at specific intervals. Thus it becomes highly important to ensure proper execution of these services.