Transport Layer

The basic function of the Transport layer is to accept data from the layer above, split it up into smaller units, pass these data units to the Network layer, and ensure that all the pieces arrive correctly at the other end.

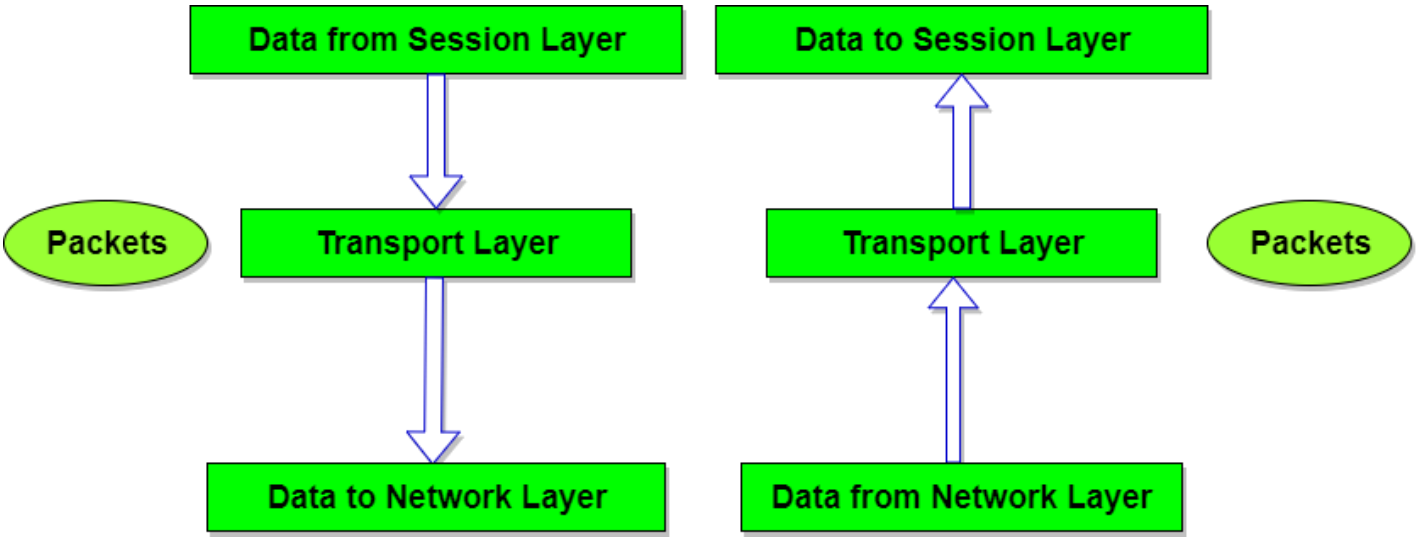
Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology.

The Transport layer also determines what type of service to provide to the Session layer, and, ultimately, to the users of the network. The most popular type of transport connection is an **error-free point-to-point channel** that delivers messages or bytes in the order in which they were sent.

The Transport layer is a true end-to-end layer, all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages.

Functions of Transport Layer

1. **Service Point Addressing:** Transport Layer header includes service point address which is port address. This layer gets the message to the correct process on the computer unlike Network Layer, which gets each packet to the correct computer.
2. **Segmentation and Reassembling:** A message is divided into segments; each segment contains sequence number, which enables this layer in reassembling the message. Message is reassembled correctly upon arrival at the destination and replaces packets which were lost in transmission.
3. **Connection Control:** It includes 2 types:
    - Connectionless Transport Layer : Each segment is considered as an independent packet and delivered to the transport layer at the destination machine.
    - Connection Oriented Transport Layer : Before delivering packets, connection is made with transport layer at the destination machine.
4. **Flow Control:** In this layer, flow control is performed end to end.
5. **Error Control:** Error Control is performed end to end in this layer to ensure that the complete message arrives at the receiving transport layer without any error. Error Correction is done through retransmission.



Design Issues with Transport Layer

- Accepting data from Session layer, split it into segments and send to the network layer.
- Ensure correct delivery of data with efficiency.
- Isolate upper layers from the technological changes.
- Error control and flow control.

Process to Process Delivery:

The data link layer is responsible for delivery of frames between two neighboring nodes over a link. This is called node-to-node delivery. The network layer is responsible for delivery of datagrams between two hosts. This is called host-to-host delivery. Real communication takes place between two processes (application programs). We need process-to-process delivery. The transport layer is responsible for process-to-process delivery-the delivery of a packet, part of a message, from one process to another. Figure 4.1 shows these three types of deliveries and their domains

The transport layer is responsible for process-to-process delivery.

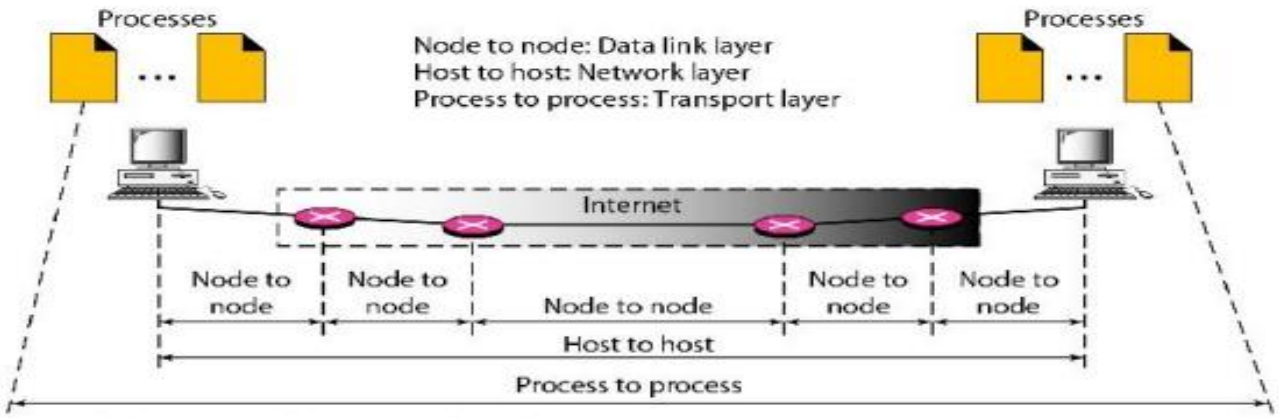


Figure 4.1 Types of data deliveries

1. Client/Server Paradigm

Although there are several ways to achieve process-to-process communication, the most common one is through the client/server paradigm. A process on the local host, called a client, needs services from a process usually on the remote host, called a server. Both processes (client and server) have the same name. For example, to get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine. For communication, we must define the following:

1. Local host
2. Local process
3. Remote host
4. Remote process

2. Addressing

Whenever we need to deliver something to one specific destination among many, we need an address. At the data link layer, we need a MAC address to choose one node among several nodes if the connection is not point-to-point. A frame in the data link layer needs a Destination MAC address for delivery and a source address for the next node's reply.

Figure 4.2 shows this concept.

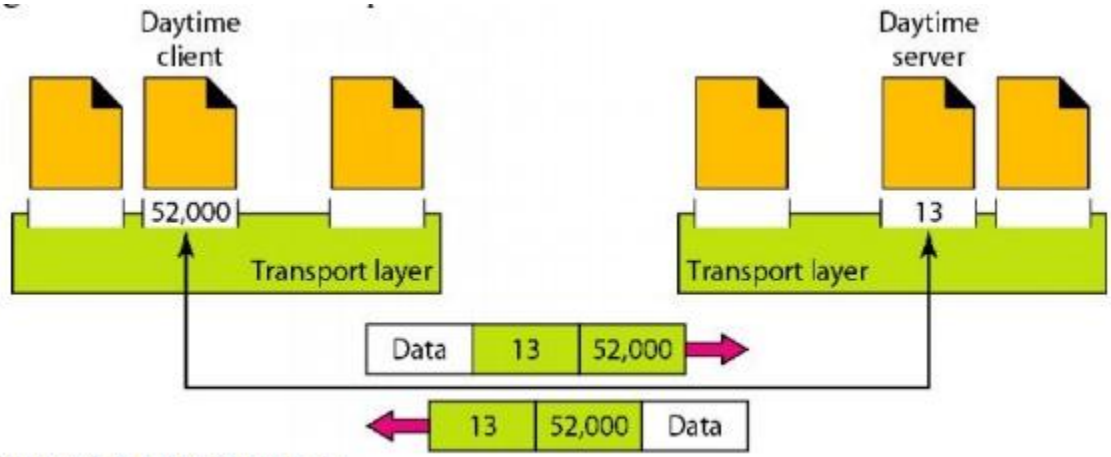


Figure 4.2 Port Numbers

The IP addresses and port numbers play different roles in selecting the final destination of data. The destination IP address defines the host among the different hosts in the world. After the host has been selected, the port number defines one of the processes on this particular host (see Figure 4.3).

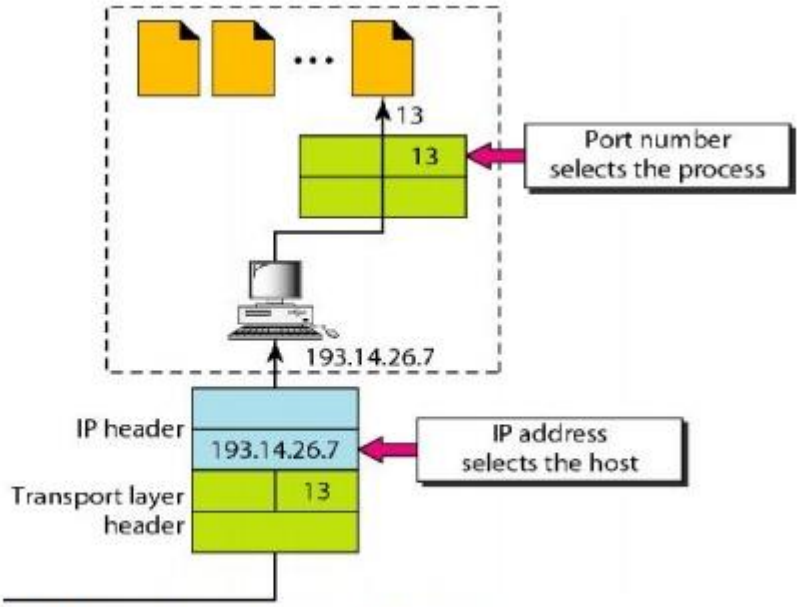


Figure 4.3 IP addresses versus port numbers

3. lANA Ranges

The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (or private), as shown in Figure 4.4.

1. **Well-known ports**. The ports ranging from 0 to 1023 are assigned and controlledby lANA. These are the well-known ports.
2. **Registered ports.** The ports ranging from 1024 to 49,151 are not assigned orcontrolled by lANA. They can only be registered with lANA to prevent duplication.
3. **Dynamic ports**. The ports ranging from 49,152 to 65,535 are neither controllednor registered. They can be used by any process. These are the ephemeral ports.

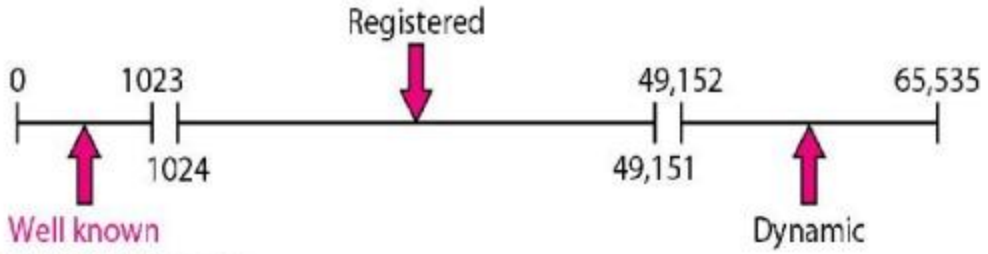


Figure 4.4 lANA Ranges

4. Socket Addresses

Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a socket address. The client socket address

defines the client process uniquely just as the server socket address defines the server process uniquely (see Figure 4.5).

UDP or TCP header contains the port numbers.

Figure 4.5 Socket Address

5. Multiplexing and Demultiplexing

The addressing mechanism allows multiplexing and demultiplexing by the transport layer, as shown in Figure 4.6.

Multiplexing

At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing.

Demultiplexing

At the receiver site, the relationship is one-to-many and requires demultiplexing. The transport layer receives datagrams from the network layer. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.

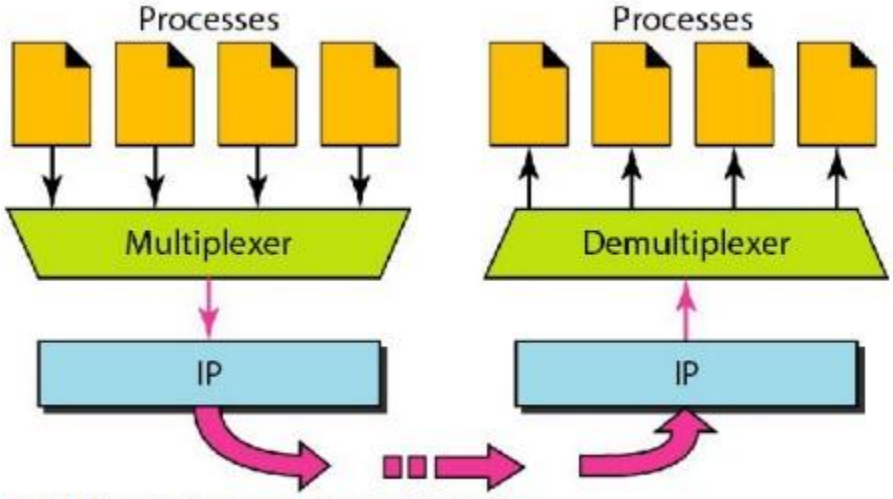


Figure 4.6 Multiplexing and Demultiplexing

6. Connectionless Versus Connection-Oriented Service

A transport layer protocol can either be connectionless or connection-oriented.

Connectionless Service

In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either.

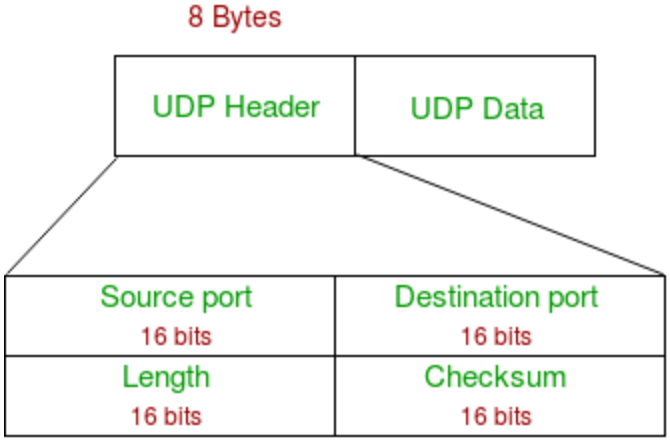
Connection~Oriented Service

In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released.

UDP (User Datagram Protocol)

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an **unreliable and connectionless protocol.** So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network.The UDP enables process to process communication.

UDP header is an **8-bytes** fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contains all necessary header information and the remaining part consist of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.



1. **Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.
2. **Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.
3. **Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.
4. **Checksum:** Checksum is 2 Bytes long field. It is the 16-bit one's complement of the one's complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Advantages of UDP:

1. Speed: UDP is faster than TCP because it does not have the overhead of establishing a connection and ensuring reliable data delivery.

2. Lower latency: Since there is no connection establishment, there is lower latency and faster response time.

3. Simplicity: UDP has a simpler protocol design than TCP, making it easier to implement and manage.

4. Broadcast support: UDP supports broadcasting to multiple recipients, making it useful for applications such as video streaming and online gaming.

5. Smaller packet size: UDP uses smaller packet sizes than TCP, which can reduce network congestion and improve overall network performance.

Disadvantages of UDP:

1. No reliability: UDP does not guarantee delivery of packets or order of delivery, which can lead to missing or duplicate data.

2. No congestion control: UDP does not have congestion control, which means that it can send packets at a rate that can cause network congestion.

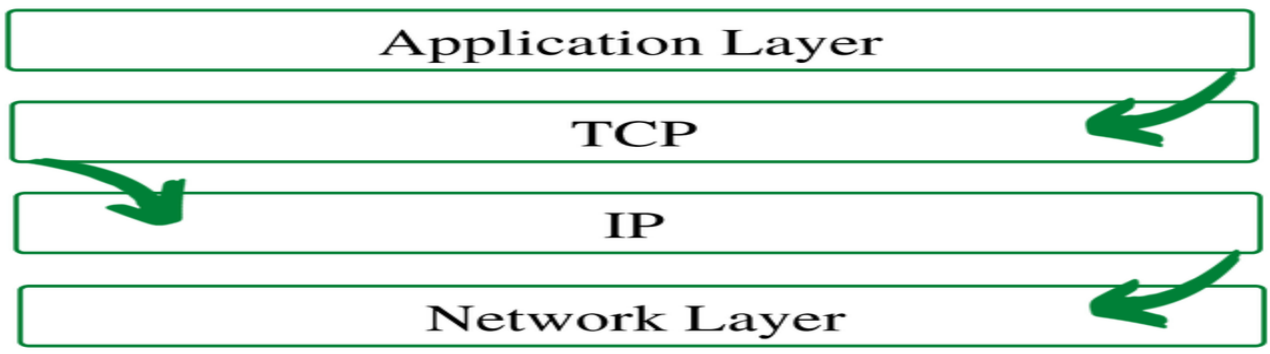
3. No flow control: UDP does not have flow control, which means that it can overwhelm the receiver with packets that it cannot handle.

4. Vulnerable to attacks: UDP is vulnerable to denial-of-service attacks, where an attacker can flood a network with UDP packets, overwhelming the network and causing it to crash.

5. Limited use cases: UDP is not suitable for applications that require reliable data delivery, such as email or file transfers, and is better suited for applications that can tolerate some data loss, such as video streaming or online gaming.

What is Transmission Control Protocol (TCP)?

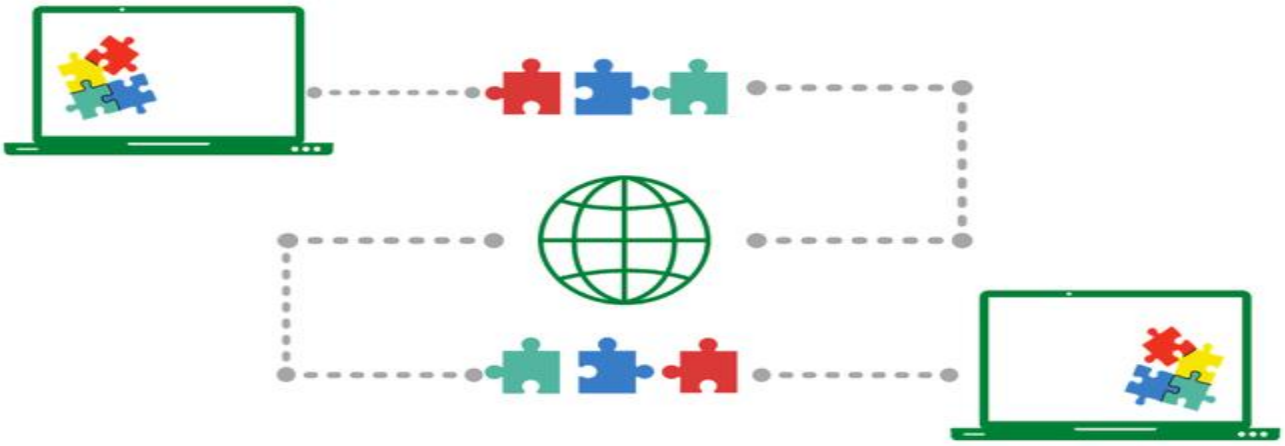
TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.



TCP/IP Layer

Working of TCP

To make sure that each message reaches its target location intact, the TCP/IP model breaks down the data into small bundles and afterward reassembles the bundles into the original message on the opposite end. Sending the information in little bundles of information makes it simpler to maintain efficiency as opposed to sending everything in one go.
After a particular message is broken down into bundles, these bundles may travel along multiple routes if one route is jammed but the destination remains the same.



We can see that the message is being broken down, then reassembled from a different order at the destination

For example, When a user requests a web page on the internet, somewhere in the world, the server processes that request and sends back an HTML Page to that user. The server makes use of a protocol called the HTTP Protocol. The HTTP then requests the TCP layer to set the required connection and send the HTML file.
Now, the TCP breaks the data into small packets and forwards it toward the Internet Protocol (IP) layer. The packets are then sent to the destination through different routes.

The TCP layer in the user's system waits for the transmission to get finished and acknowledges once all packets have been received.

Features of TCP/IP

Some of the most prominent features of Transmission control protocol are

1. Segment Numbering System

- TCP keeps track of the segments being transmitted or received by assigning numbers to each and every single one of them.
- A specific *Byte Number* is assigned to data bytes that are to be transferred while segments are assigned *sequence numbers*.
- *Acknowledgment Numbers* are assigned to received segments.

2. Connection Oriented

- It means sender and receiver are connected to each other till the completion of the process.
- The order of the data is maintained i.e. order remains same before and after transmission.

3. Full Duplex

- In TCP data can be transmitted from receiver to the sender or vice – versa at the same time.
- It increases efficiency of data flow between sender and receiver.

4. Flow Control

- Flow control limits the rate at which a sender transfers data. This is done to ensure reliable delivery.
- The receiver continually hints to the sender on how much data can be received (using a sliding window)

5. Error Control

- TCP implements an error control mechanism for reliable data transfer
- Error control is byte-oriented
- Segments are checked for error detection
- Error Control includes – *Corrupted Segment & Lost Segment Management, Out-of-order segments, Duplicate segments*, etc.

6. Congestion Control

- TCP takes into account the level of congestion in the network
- Congestion level is determined by the amount of data sent by a sender

Advantages

- It is a reliable protocol.
- It provides an error-checking mechanism as well as one for recovery.
- It gives flow control.
- It makes sure that the data reaches the proper destination in the exact order that it was sent.
- Open Protocol, not owned by any organization or individual.
- It assigns an IP address to each computer on the network and a domain name to each site thus making each device site to be distinguishable over the network.

Disadvantages

- TCP is made for Wide Area Networks, thus its size can become an issue for small networks with low resources.
- TCP runs several layers so it can slow down the speed of the network.
- It is not generic in nature. Meaning, it cannot represent any protocol stack other than the TCP/IP suite. E.g., it cannot work with a Bluetooth connection.
- No modifications since their development around 30 years ago.

Congestion Control in Computer Networks

What is **congestion**?

A state occurring in network layer when the message traffic is so heavy that it slows down network response time.

Effects of Congestion

- As delay increases, performance decreases.
- If delay increases, retransmission occurs, making situation worse.

Congestion control algorithms

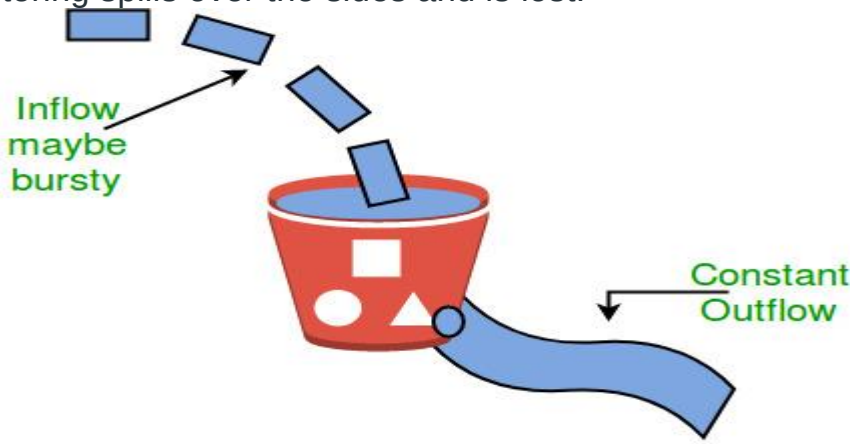
- Congestion Control is a mechanism that controls the entry of data packets into the network, enabling a better use of a shared network infrastructure and avoiding congestive collapse.
- Congestive-Avoidance Algorithms (CAA) are implemented at the TCP layer as the mechanism to avoid congestive collapse in a network.
- There are two congestion control algorithm which are as follows:

Leaky Bucket Algorithm

- The leaky bucket algorithm discovers its use in the context of network traffic shaping or rate-limiting.
- A leaky bucket execution and a token bucket execution are predominantly used for traffic shaping algorithms.
- This algorithm is used to control the rate at which traffic is sent to the network and shape the burst traffic to a steady traffic stream.
- The disadvantages compared with the leaky-bucket algorithm are the inefficient use of available network resources.
- The large area of network resources such as bandwidth is not being used effectively.

Let us consider an example to understand

Imagine a bucket with a small hole in the bottom.No matter at what rate water enters the bucket, the outflow is at constant rate.When the bucket is full with water additional water entering spills over the sides and is lost.



Similarly, each network interface contains a leaky bucket and the following **steps** are involved in leaky bucket algorithm:

1. When host wants to send packet, packet is thrown into the bucket.
2. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
3. Bursty traffic is converted to a uniform traffic by the leaky bucket.
4. In practice the bucket is a finite queue that outputs at a finite rate.

Token bucket Algorithm

- The leaky bucket algorithm has a rigid output design at an average rate independent of the bursty traffic.
- In some applications, when large bursts arrive, the output is allowed to speed up. This calls for a more flexible algorithm, preferably one that never loses information. Therefore, a token bucket algorithm finds its uses in network traffic shaping or rate-limiting.

- It is a control algorithm that indicates when traffic should be sent. This order comes based on the display of tokens in the bucket.
- The bucket contains tokens. Each of the tokens defines a packet of predetermined size. Tokens in the bucket are deleted for the ability to share a packet.
- When tokens are shown, a flow to transmit traffic appears in the display of tokens.
- No token means no flow sends its packets. Hence, a flow transfers traffic up to its peak burst rate in good tokens in the bucket.

Need of token bucket Algorithm:-

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

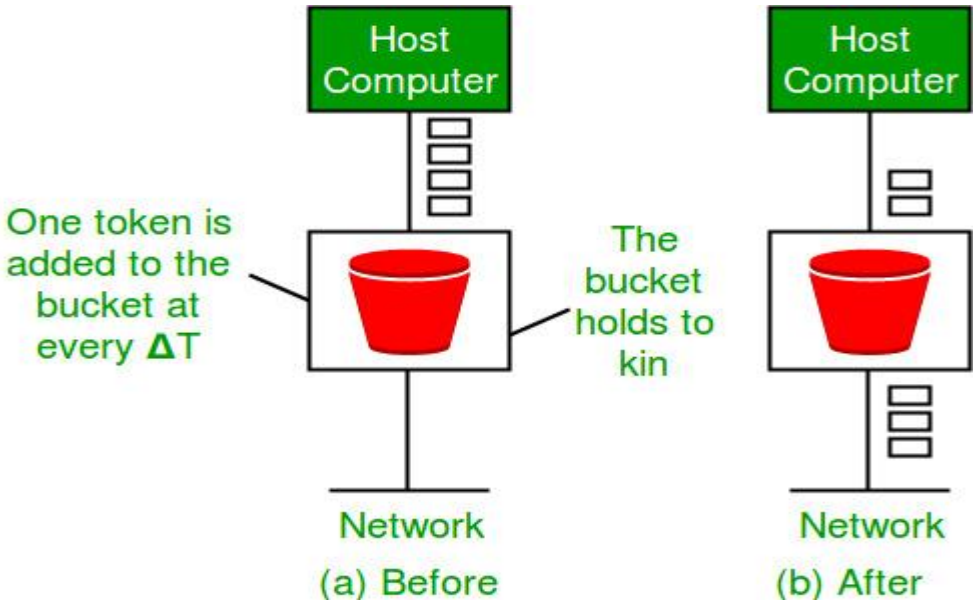
Steps of this algorithm can be described as follows:

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

Let's understand with an example,

In figure (A) we see a bucket holding three tokens, with five packets waiting to be transmitted. For a packet to be transmitted, it must capture and destroy one token. In figure (B) We see that three of the five packets have gotten through, but the other two are stuck waiting for more tokens to be generated.

Ways in which token bucket is superior to leaky bucket: The leaky bucket algorithm controls the rate at which the packets are introduced in the network, but it is very conservative in nature. Some flexibility is introduced in the token bucket algorithm. In the token bucket, algorithm tokens are generated at each tick (up to a certain limit). For an incoming packet to be transmitted, it must capture a token and the transmission takes place at the same rate. Hence some of the busty packets are transmitted at the same rate if tokens are available and thus introduces some amount of flexibility in the system.
Formula: $M * s = C + \rho * s$ where S – is time taken M – Maximum output rate ρ – Token arrival rate C – Capacity of the token bucket in byte
Let's understand with an example,



Choke Packet Technique :

Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. Whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets has traveled are not warned about congestion.