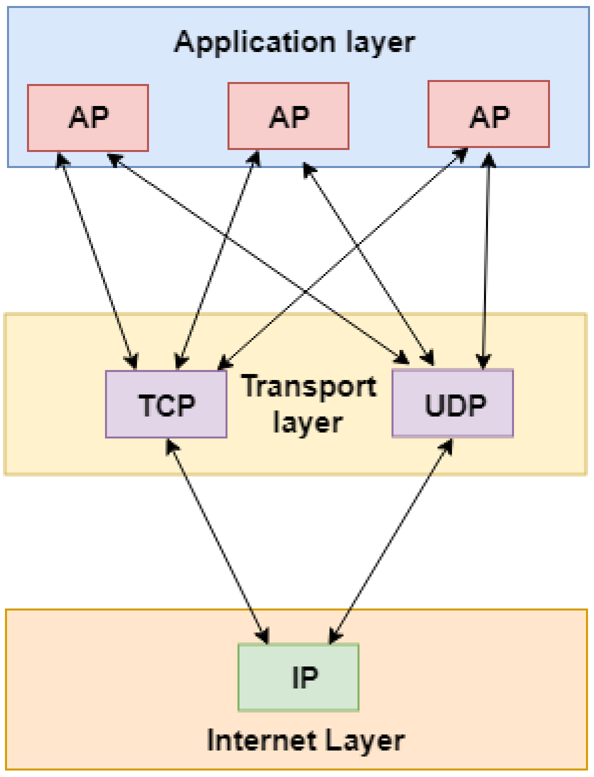
Transport Layer

- The transport layer is a 4th layer from the top.
- The main role of the transport layer is to provide the communication services directly to the application processes running on different hosts.
- The transport layer provides a logical communication between application processes running on different hosts. Although the application processes on different hosts are not physically connected, application processes use the logical communication provided by the transport layer to send the messages to each other.
- The transport layer protocols are implemented in the end systems but not in the network routers.
- A computer network provides more than one protocol to the network applications. For example, TCP and UDP are two transport layer protocols that provide a different set of services to the network layer.
- All transport layer protocols provide multiplexing/demultiplexing service. It also provides other services such as reliable data transfer, bandwidth guarantees, and delay guarantees.
- Each of the applications in the application layer has the ability to send a message by using TCP or UDP. The application communicates by using either of these two protocols. Both TCP and UDP will then communicate with the internet protocol in the internet layer. The applications can read and write to the transport layer. Therefore, we can say that communication is a two-way process.

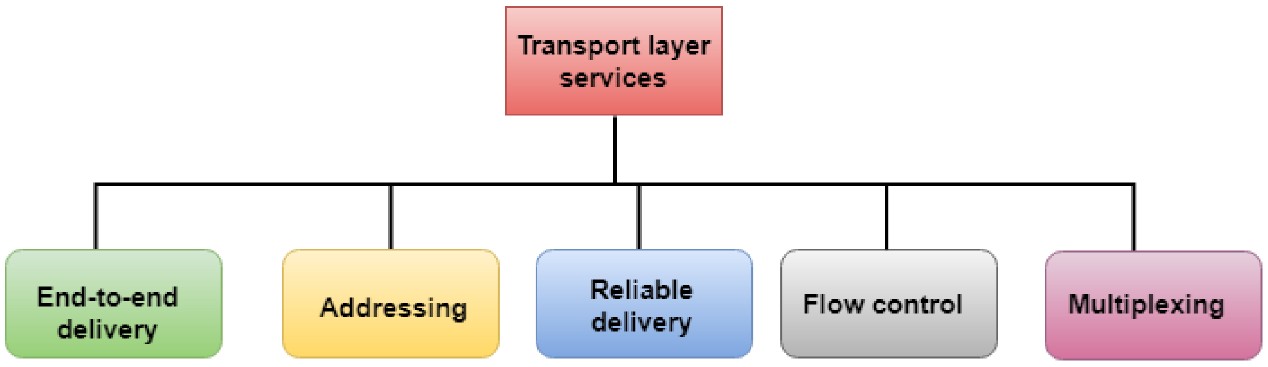


Services provided by the Transport Layer

The services provided by the transport layer are similar to those of the data link layer. The data link layer provides the services within a single network while the transport layer provides the services across an internetwork made up of many networks. The data link layer controls the physical layer while the transport layer controls all the lower layers.

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:

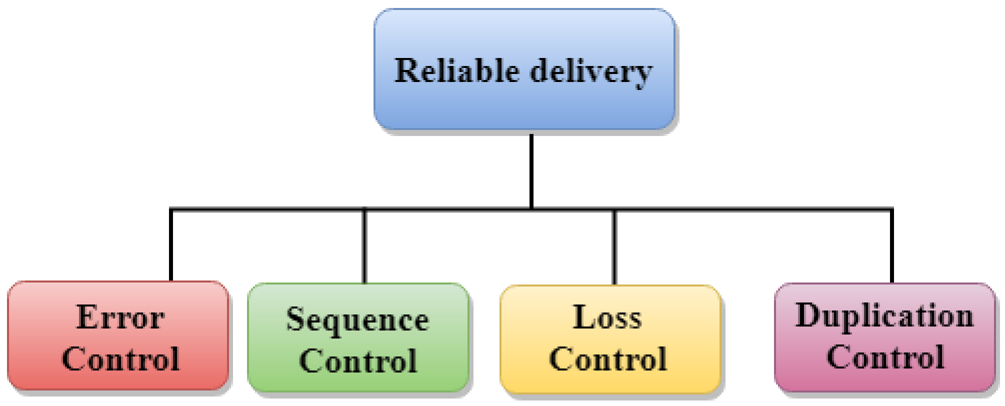
The transport layer transmits the entire message to the destination. Therefore, it ensures the end-to-end delivery of an entire message from a source to the destination.

Reliable delivery:

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

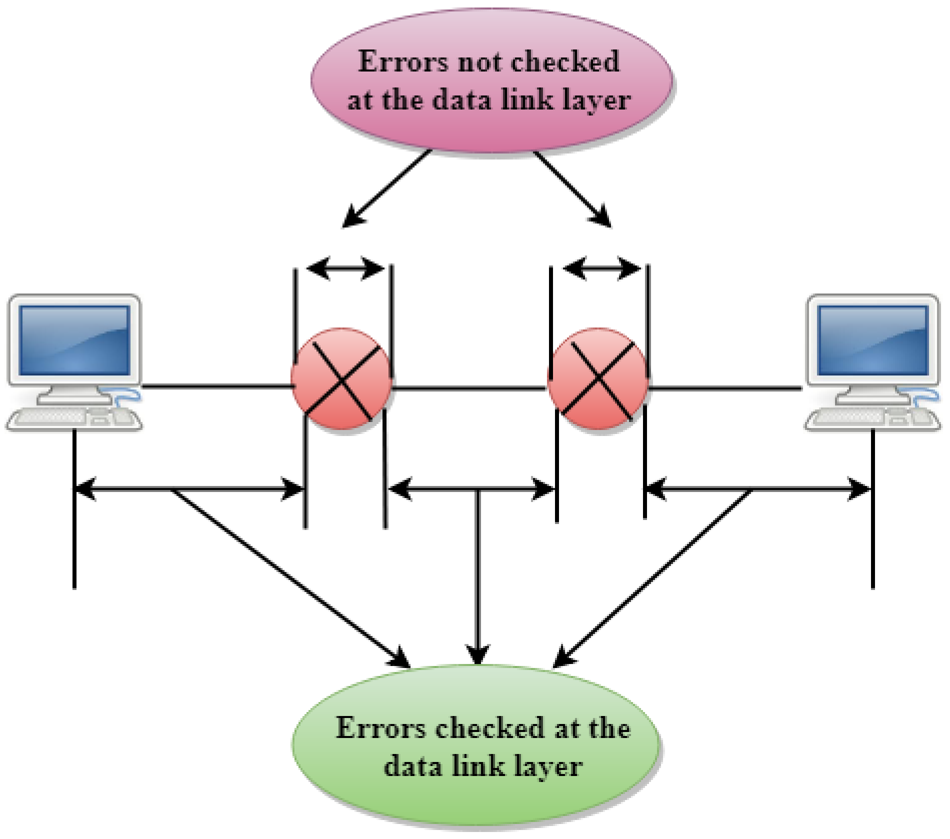
The reliable delivery has four aspects:

- 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.


Errors not checked at the data link layer
Errors checked at the data link layer

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

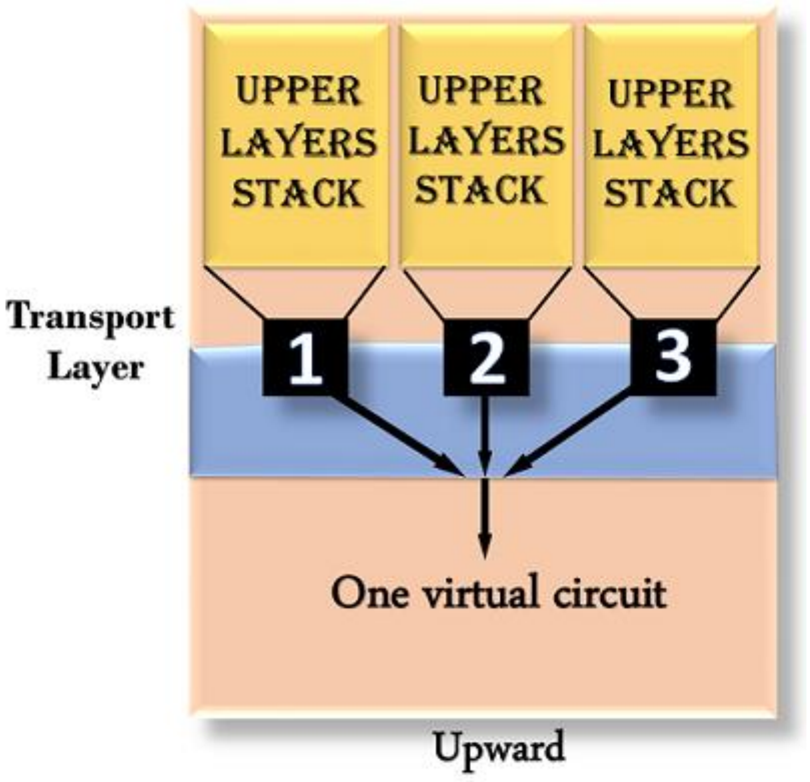
Flow control is used to prevent the sender from overwhelming the receiver. If the receiver is overloaded with too much data, then the receiver discards the packets and asking for the retransmission of packets. This increases network congestion and thus, reducing the system performance. The transport 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

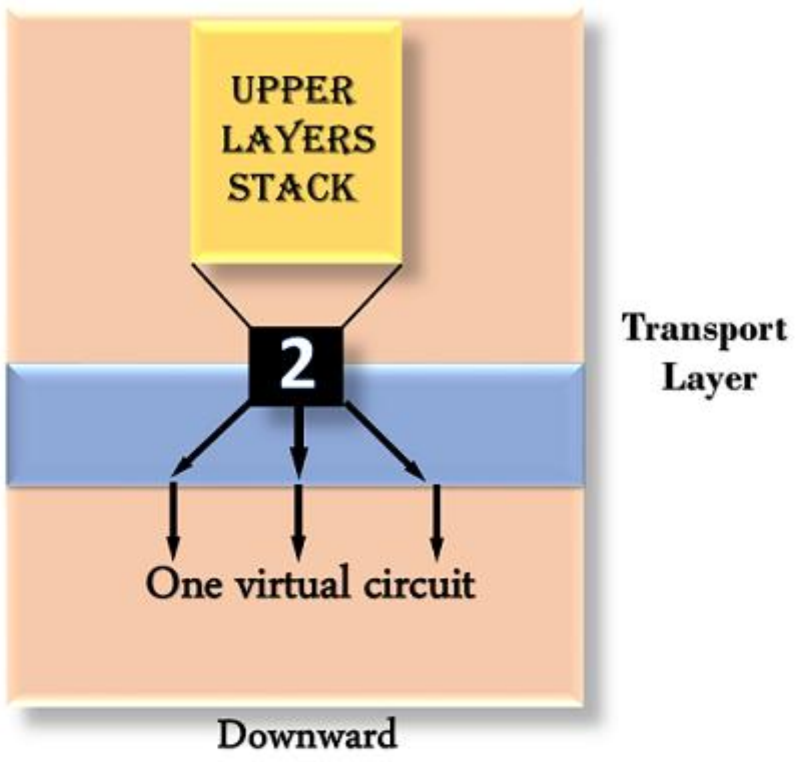
The transport layer uses the multiplexing to improve transmission efficiency.

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.

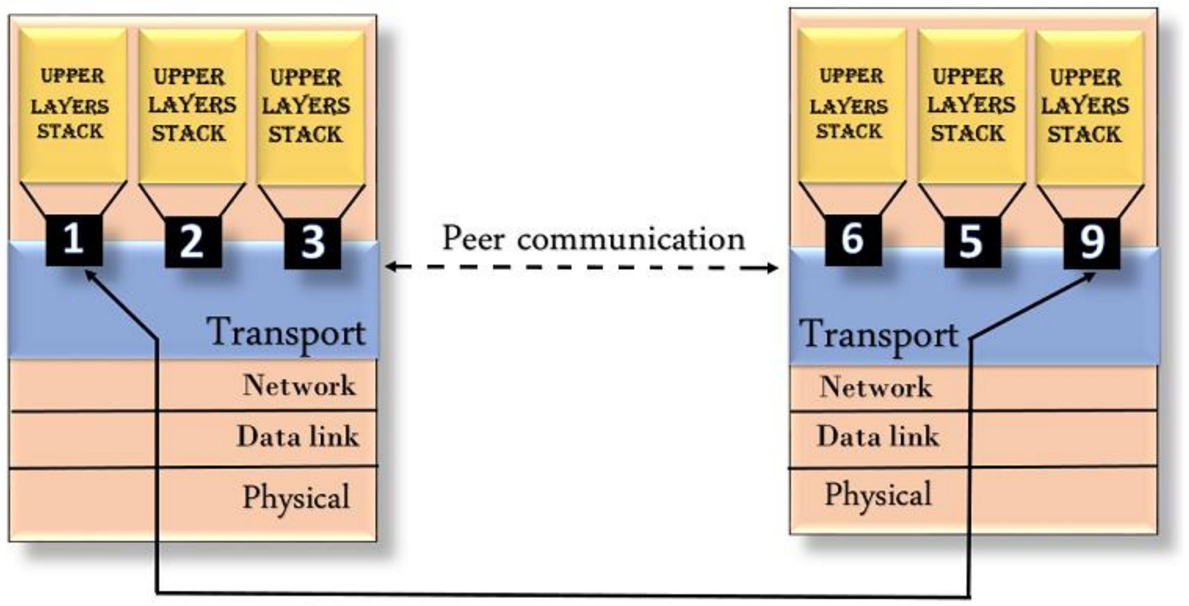


- **Downward multiplexing:** Downward multiplexing means one transport layer connection uses the multiple network connections. Downward multiplexing allows the transport layer to split a connection among several paths to improve the throughput. This type of multiplexing is used when networks have a low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

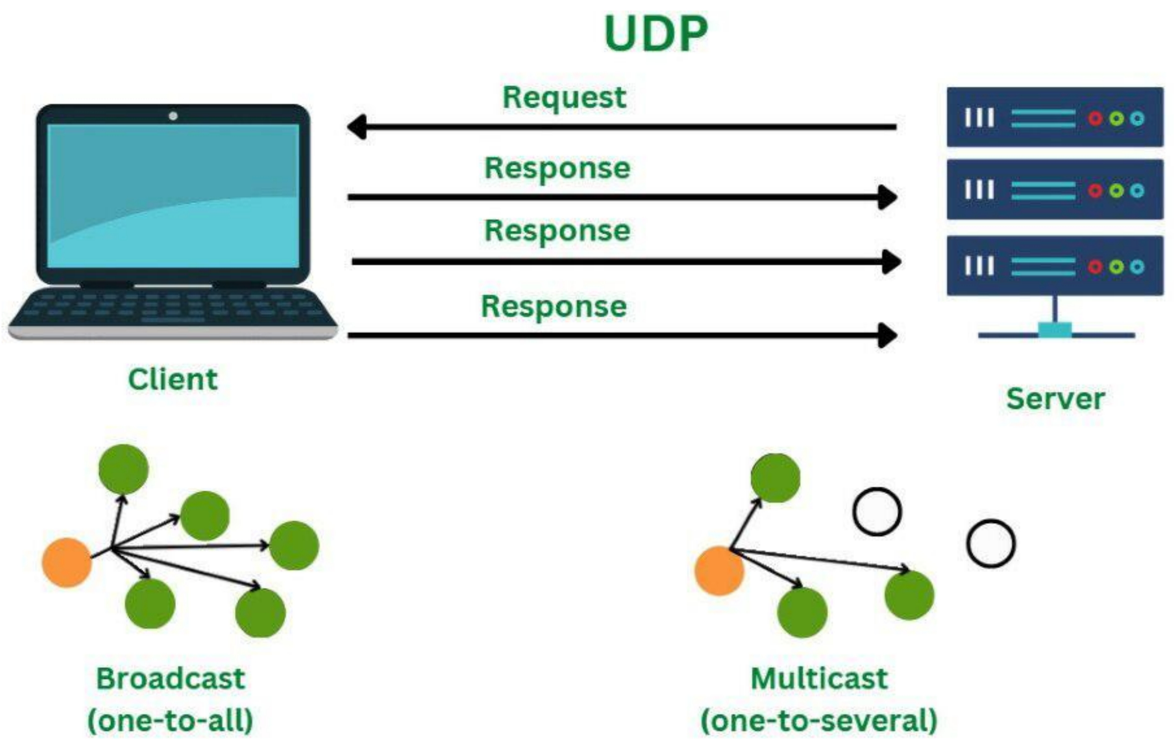


Transport Layer Protocols

The transport layer is represented majorly by TCP and UDP protocols. Today almost all operating systems support multiprocessing multi-user environments. This transport layer protocol provides connections to the individual ports. These ports are known as protocol ports. Transport layer protocols work above the IP protocols and deliver the data packets from IP serves to destination port and from the originating port to destination IP services. Below are the protocols used at the transport layer.

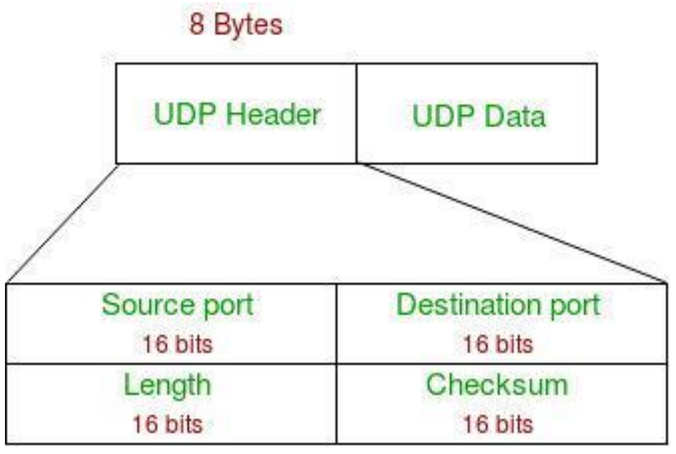
1. UDP

UDP stands for User Datagram Protocol. User Datagram Protocol provides a nonsequential transmission of data. It is a connectionless transport protocol. UDP protocol is used in applications where the speed and size of data transmitted is considered as more important than the security and reliability. User Datagram is defined as a packet produced by User Datagram Protocol. UDP protocol adds checksum error control, transport level addresses, and information of length to the data received from the layer above it. Services provided by User Datagram Protocol(UDP) are connectionless service, faster delivery of messages, checksum, and process-to-process communication.



UDP Segment

While the TCP header can range from 20 to 60 bytes, the UDP header is a fixed, basic 8 bytes. All required header information is contained in the first 8 bytes, with data making up the remaining portion. Because UDP port number fields are 16 bits long, the range of possible port numbers is defined as 0 to 65535, with port 0 being reserved.



UDP

- **Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.
- **Destination Port:** This 2-byte element is used to specify the packet's destination port.
- **Length:** The whole length of a UDP packet, including the data and header. The field has sixteen bits.

- **Cheksum:** The checksum field is two bytes long. The data is padded with zero octets at the end (if needed) to create a multiple of two octets. It is the 16-bit one's complement of the one's complement sum of the UDP header, the pseudo-header containing information from the IP header, and the data.

Advantages of UDP

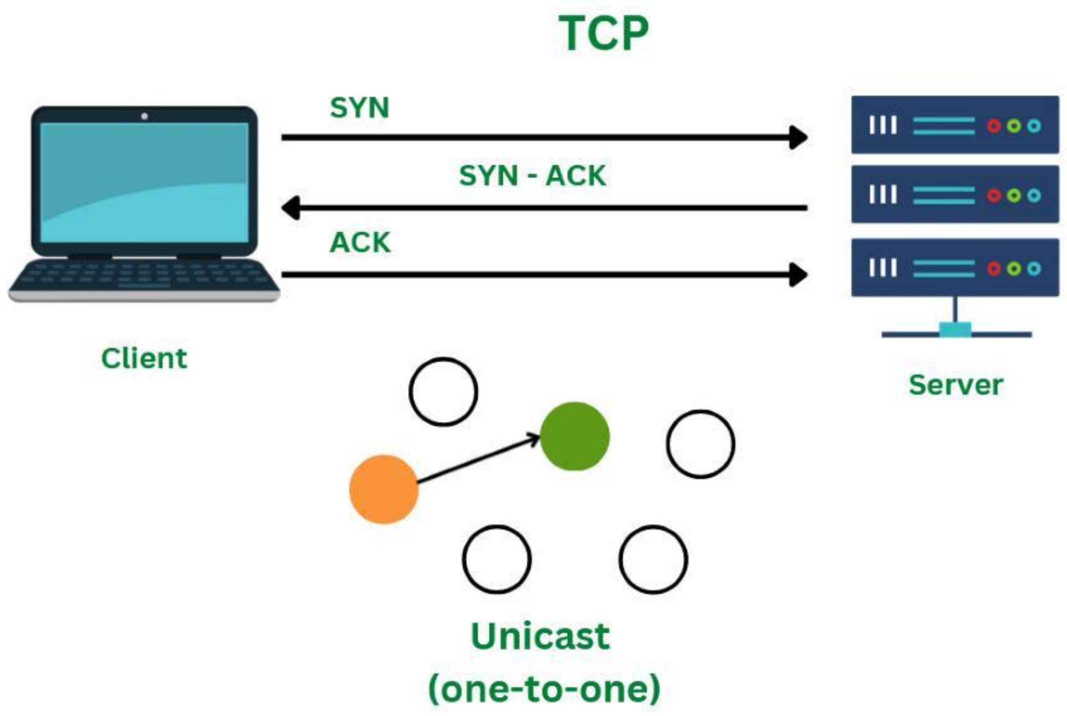
- UDP also provides multicast and broadcast transmission of data.
- UDP protocol is preferred more for small transactions such as DNS lookup.
- It is a connectionless protocol, therefore there is no compulsion to have a connection-oriented network.
- UDP provides fast delivery of messages.

Disadvantages of UDP

- In UDP protocol there is no guarantee that the packet is delivered.
- UDP protocol suffers from worse packet loss.
- UDP protocol has no congestion control mechanism.
- UDP protocol does not provide the sequential transmission of data.

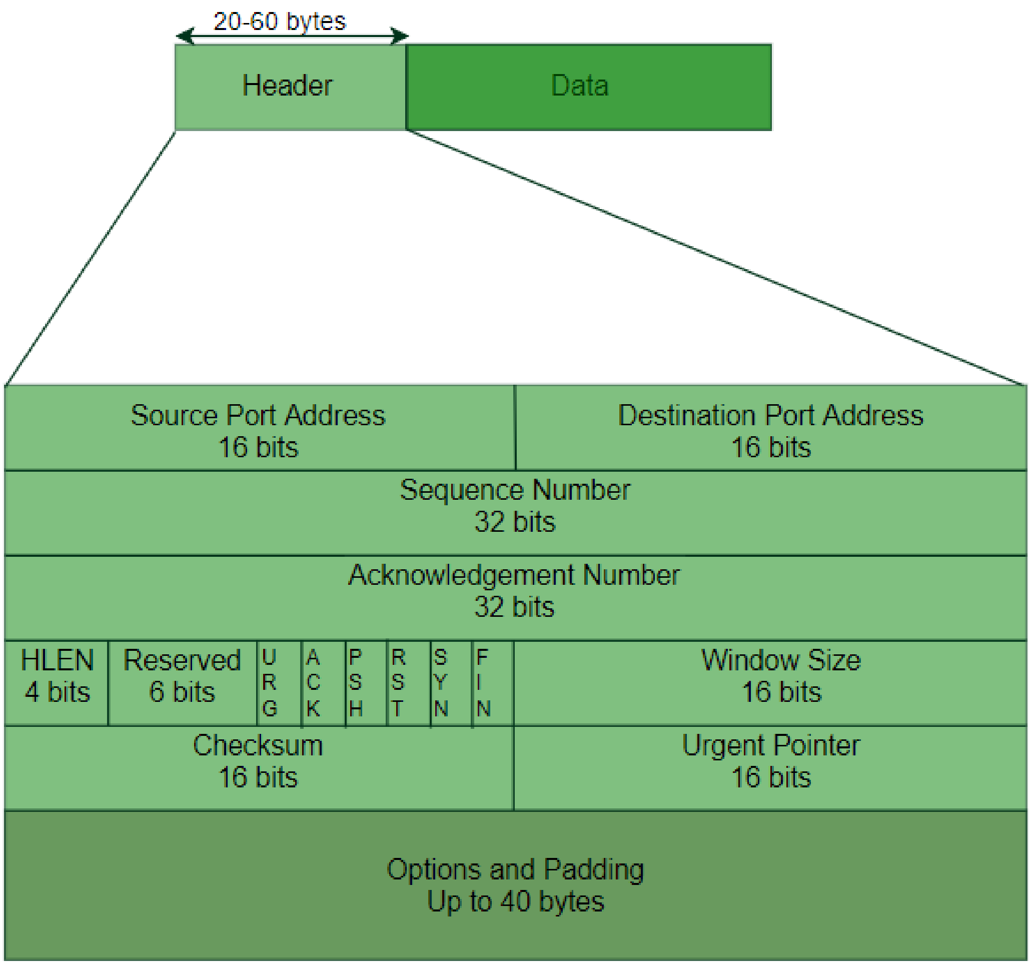
2. TCP

TCP stands for Transmission Control Protocol. TCP protocol provides transport layer services to applications. TCP protocol is a connection-oriented protocol. A secured connection is being established between the sender and the receiver. For a generation of a secured connection, a virtual circuit is generated between the sender and the receiver. The data transmitted by TCP protocol is in the form of continuous byte streams. A unique sequence number is assigned to each byte. With the help of this unique number, a positive acknowledgment is received from receipt. If the acknowledgment is not received within a specific period the data is retransmitted to the specified destination.



TCP Segment

A TCP segment's header may have 20–60 bytes. The options take about 40 bytes. A header consists of 20 bytes by default, although it can contain up to 60 bytes.



- **Source Port Address:** The port address of the programme sending the data segment is stored in the 16-bit field known as the source port address.
- **Destination Port Address:** The port address of the application running on the host receiving the data segment is stored in the destination port address, a 16-bit field.
- **Sequence Number:** The sequence number, or the byte number of the first byte sent in that specific segment, is stored in a 32-bit field. At the receiving end, it is used to put the message back together once it has been received out of sequence.
- **Acknowledgement Number** : The acknowledgement number, or the byte number that the recipient anticipates receiving next, is stored in a 32-bit field called the acknowledgement number. It serves as a confirmation that the earlier bytes were successfully received.
- **Header Length (HLEN):** This 4-bit field stores the number of 4-byte words in the TCP header, indicating how long the header is. For example, if the header is 20 bytes (the minimum length of the TCP header), this field will store 5 because 5 x 4 = 20, and if the header is 60 bytes (the maximum length), it will store 15 because 15 x 4 = 60. As a result, this field's value is always between 5 and 15.

- **Control flags:** These are six 1-bit control bits that regulate flow control, method of transfer, connection abortion, termination, and establishment. They serve the following purposes:
    - **Urgent:** This pointer is legitimate
    - **ACK:** The acknowledgement number (used in cumulative acknowledgement cases) is valid.
    - **PSH:** Push request
    - **RST:** Restart the link.
    - SYN: Sequence number synchronisation
    - **FIN:** Cut off the communication
    - **Window size:** This parameter provides the sender TCP's window size in bytes.
- **Checksum:** The checksum for error control is stored in this field. Unlike UDP, it is required for TCP.
- **Urgent pointer:** This field is used to point to data that must urgently reach the receiving process as soon as possible. It is only valid if the URG control flag is set. To obtain the byte number of the final urgent byte, the value of this field is appended to the sequence number.

Advantages of TCP

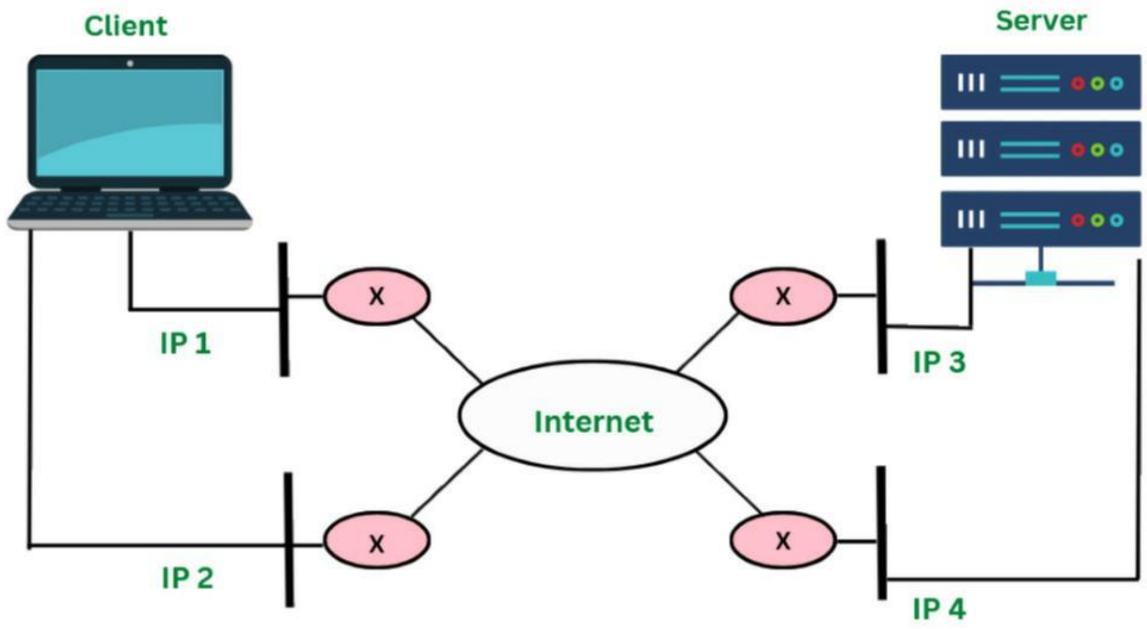
- TCP supports multiple routing protocols.
- TCP protocol operates independently of that of the operating system.
- TCP protocol provides the features of error control and flow control.
- TCP provides a connection-oriented protocol and provides the delivery of data.

Disadvantages of TCP

- TCP protocol cannot be used for broadcast or multicast transmission.
- TCP protocol has no block boundaries.
- No clear separation is being offered by TCP protocol between its interface, services, and protocols.
- In TCP/IP replacement of protocol is difficult.

3. SCTP

SCTP stands for Stream Control Transmission Protocol. SCTP is a connection-oriented protocol. Stream Control Transmission Protocol transmits the data from sender to receiver in full duplex mode. SCTP is a unicast protocol that provides with point to point-to-point connection and uses different hosts for reaching the destination. SCTP protocol provides a simpler way to build a connection over a wireless network. SCTP protocol provides a reliable transmission of data. SCTP provides a reliable and easier telephone conversation over the internet. SCTP protocol supports the feature of multihoming ie. it can establish more than one connection path between the two points of communication and does not depend on the IP layer. SCTP protocol also ensures security by not allowing the half-open connections.



Advantages of SCTP

- SCTP provides a full duplex connection. It can send and receive the data simultaneously.
- SCTP protocol possesses the properties of both TCP and UDP protocol.
- SCTP protocol does not depend on the IP layer.
- SCTP is a secure protocol.

Disadvantages of SCTP

- To handle multiple streams simultaneously the applications need to be modified accordingly.
- The transport stack on the node needs to be changed for the SCTP protocol.
- Modification is required in applications if SCTP is used instead of TCP or UDP protocol.

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.

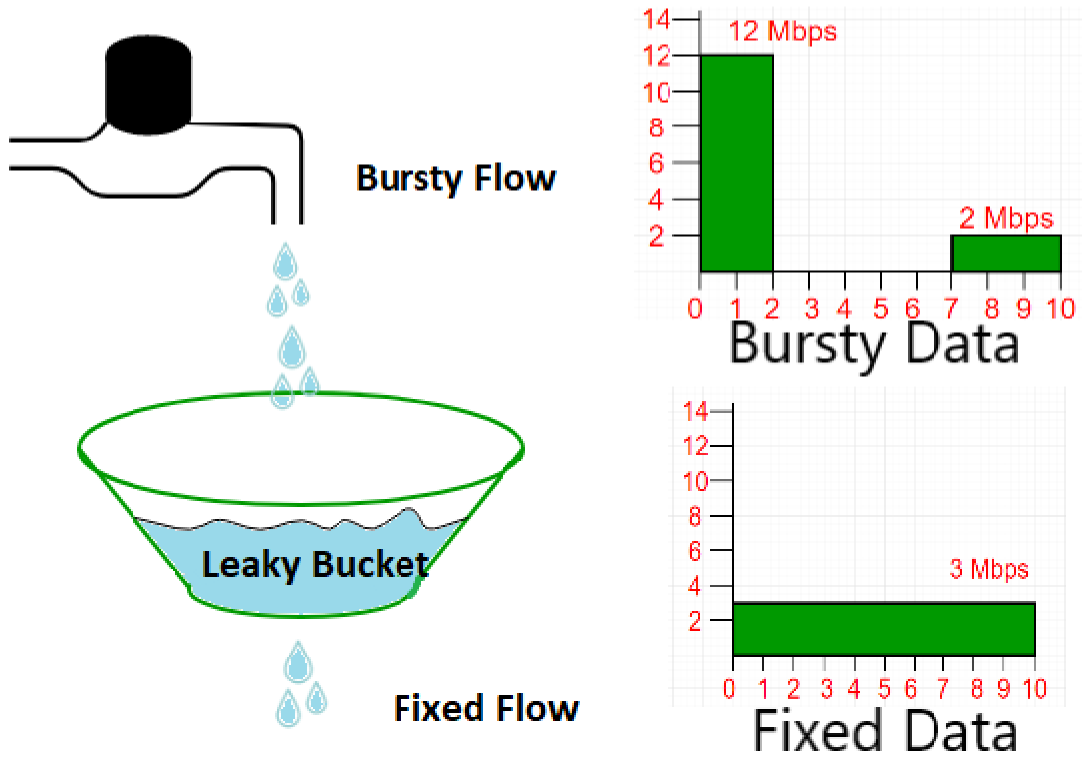
To understand this concept first we have to know little about traffic shaping. **Traffic Shaping** is a mechanism to control the amount and the rate of traffic sent to the network. Approach of congestion management is called Traffic shaping. Traffic shaping helps to regulate the rate of data transmission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

There are 2 types of traffic shaping algorithms:

1. Leaky Bucket
2. Token Bucket

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=

200	700	500	450	400	200

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.

200	700	500	450	400

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.

200	700	500	450

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.

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.
- **Predictable Traffic Shaping:** Token Bucket offers more predictable traffic shaping compared to leaky bucket. With token bucket, the network administrator can set the rate at which tokens are added to the bucket, and the maximum number of tokens that the bucket can hold. This allows for better control over the network traffic and can help prevent congestion.
- **Better Quality of Service (QoS):** Token Bucket provides better QoS compared to leaky bucket. This is because token bucket can prioritize certain types of traffic by assigning different token arrival rates to different classes of packets. This ensures that important packets are sent first, while less important packets are sent later, helping to ensure that the network runs smoothly.

- **More efficient use of network bandwidth:** Token Bucket allows for more efficient use of network bandwidth as it allows for larger bursts of data to be sent at once. This can be useful for applications that require high-speed data transfer or for streaming video content.
- **More granular control:** Token Bucket provides more granular control over network traffic compared to leaky bucket. This is because it allows the network administrator to set the token arrival rate and the maximum token count, which can be adjusted according to the specific needs of the network.
- **Easier to implement:** Token Bucket is generally considered easier to implement compared to leaky bucket. This is because token bucket only requires the addition and removal of tokens from a bucket, while leaky bucket requires the use of timers and counters to determine when to release packets.

Some Disadvantage of token Bucket over leaky bucket

- **Tokens may be wasted:** In Token Bucket, tokens are generated at a fixed rate, even if there is no traffic on the network. This means that if no packets are sent, tokens will accumulate in the bucket, which could result in wasted resources. In contrast, with leaky bucket, the network only generates packets when there is traffic, which helps to conserve resources.
- **Delay in packet delivery:** Token Bucket may introduce delay in packet delivery due to the accumulation of tokens. If the token bucket is empty, packets may need to wait for the arrival of new tokens, which can lead to increased latency and packet loss.
- **Lack of flexibility:** Token Bucket is less flexible compared to leaky bucket in terms of shaping network traffic. This is because the token generation rate is fixed, and cannot be changed easily to meet the changing needs of the network. In contrast, leaky bucket can be adjusted more easily to adapt to changes in network traffic.
- **Complexity:** Token Bucket can be more complex to implement compared to leaky bucket, especially when different token generation rates are used for different types of traffic. This can make it more difficult for network administrators to configure and manage the network.
- **Inefficient use of bandwidth:** In some cases, Token Bucket may lead to inefficient use of bandwidth. This is because Token Bucket allows for large bursts of data to be sent at once, which can cause congestion and lead to packet loss. In contrast, leaky bucket helps to prevent congestion by limiting the amount of data that can be sent at any given time.

Token Bucket Algorithm

The Token Bucket algorithm is a popular and simple method used in computer networking and telecommunications for traffic shaping and rate limiting. It is designed to control the amount of data that a system can send or receive in some sort of period, ensuring that the traffic conforms to a specified rate.

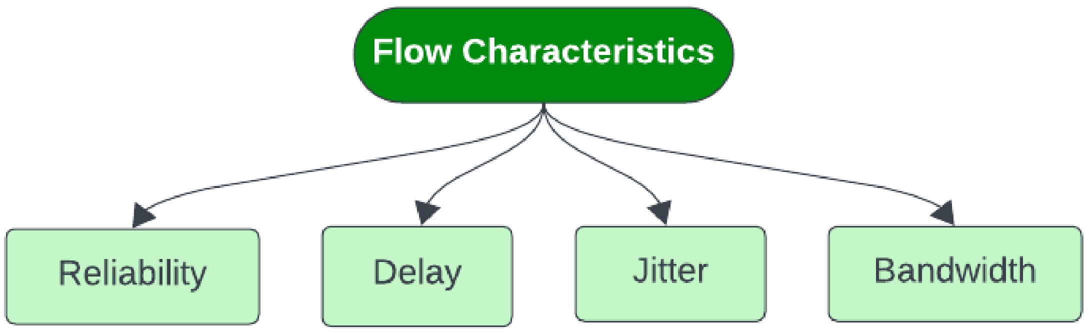
It refers to traffic control mechanisms that seek to either differentiate performance based on application or network-operator requirements or provide predictable or guaranteed performance to applications, sessions, or traffic aggregates. It is something that data flow seeks to attain.

Need for Token Bucket Algorithm

- Video and audio conferencing require a bounded delay and loss rate.
- Video and audio streaming requires a bounded packet loss rate, it may not be so sensitive to delay.
- a -critical applications (real-time control) in which bounded delay is considered to be an important factor.
- Valuable applications should provide better services than less valuable applications.

Flow Characteristics of Token Bucket Algorithm

Four types of characteristics are attributed to a flow: reliability, delay, jitter, and bandwidth.



Types of Characteristics for Quality of Service

Reliability

It implies packet reached or not, information lost or not. Lack of reliability means losing a packet or acknowledgement, which entails re-transmission. Reliability requirements may differ from program to program. For example, it is more important that electronic mail, file transfer and internet access have reliable transmissions than telephony or audio conferencing.

Delay

It denotes source-to-destination delay. Different applications can tolerate delay in different degrees. Telephony, audio conferencing, video conferencing, and remote log-in need minimum delay, while delay in file transfer or e-mail is less important.

Jitter

Jitter is the variation in delay for packets belonging in same flow. High jitter means the difference between delays is large; low jitter means the variation is small. For example, if packets 0,1,2,3s arrive at 6,7,8,9s it represents same delay. Jitter would signify that packets departed at 0,1,2,3s reach destination at 4,6,10,15s. Audio and video applications don't allow jitter.

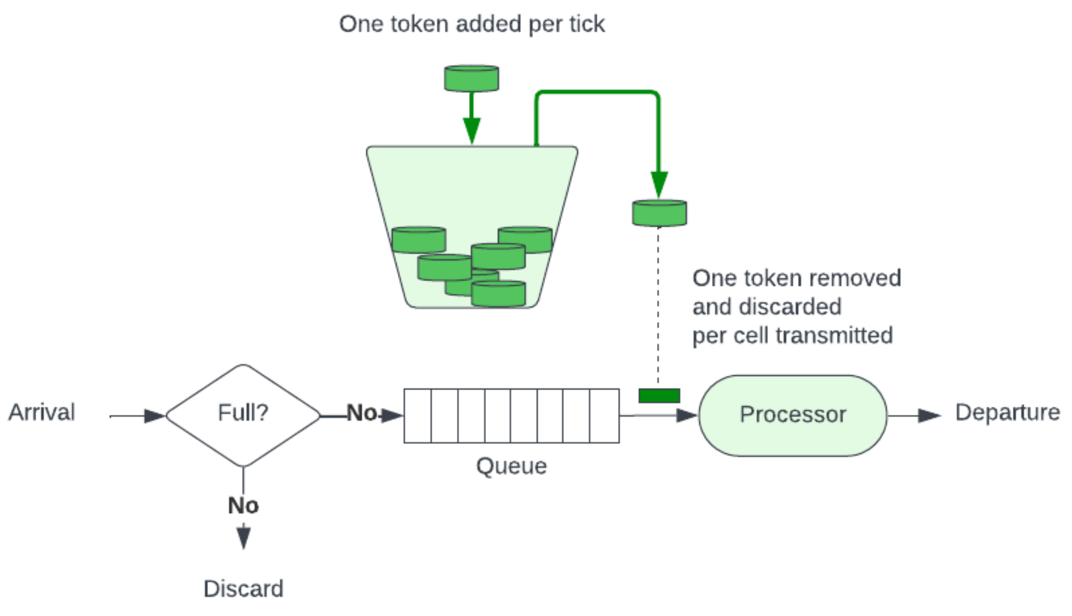
Bandwidth

Different applications need different bandwidths. In video conferencing we need to send millions of bits per second to refresh a color screen while the total number of bits in an e-mail may not reach even a million.

Working of Token Bucket Algorithm

It allows bursty traffic at a regulated maximum rate. It allows idle hosts to accumulate credit for the future in the form of tokens. The system removes one token for every cell of data sent. For each tick of the clock the system send n tokens to the bucket. If n is 100 and host is idle for 100 ticks, bucket collects 10000 tokens. Host can now consume all these tokens with 10 cells per tick.

Token bucket can be easily implemented with a counter. The token is initiated to zero. Each time a token is added, counter is incremented to 1. Each time a unit of data is sent, counter is decremented by 1. When the counter is zero, host cannot send data.



Process depicting how token bucket algorithm works

Steps Involved in Token Bucket Algorithm

Step 1: Creation of Bucket: An imaginative bucket is assigned a fixed capacity, known as “rate limit”. It can hold up to a certain number of tokens.

Step 2: Refill the Bucket: The bucket is dynamic; it gets periodically filled with tokens. Tokens are added to the bucket at a fixed rate.

Step 3: Incoming Requests: Upon receiving a request, we verify the presence of tokens in the bucket.

Step 4: Consume Tokens: If there are tokens in the bucket, we pick one token from it. This means the request is allowed to proceed. The time of token consumption is also recorded.

Step 5: Empty Bucket: If the bucket is depleted, meaning there are no tokens remaining, the request is denied. This precautionary measure prevents server or system overload, ensuring operation stays within predefined limits.

Advantage of Token Bucket over Leaky Bucket

- If a bucket is full in tokens, then tokens are discarded and not the packets. While in leaky bucket algorithm, **packets are discarded**.
- Token bucket can send large bursts at a **faster rate** while leaky bucket always sends packets at constant rate.
- Token bucket ensures **predictable traffic shaping** as it allows for setting token arrival rate and maximum token count. In leaky bucket, such control may not be present.
- Premium Quality of Service(QoS) is provided by prioritizing different traffic types through distinct token arrival rates. Such **flexibility in prioritization** is not provided by leaky bucket.
- Token bucket is suitable for high-speed data transfer or streaming video content as it allows **transmission of large bursts of data**. As leaky bucket operates at a constant rate, it can lead to less efficient bandwidth utilization.
- Token Bucket provides more **granular control** as administrators can adjust token arrival rate and maximum token count based on network requirements. Leaky Bucket has limited granularity in controlling traffic compared to Token Bucket.

Disadvantages of Token Bucket Algorithm

- Token Bucket has the tendency to generate tokens at a fixed rate, even when the network traffic is not present. This is leads of accumulation of unused tokens during times when there is no traffic, hence leading to wastage.

- Due to token accumulation, delays can introduced in the packet delivery. If the token bucket happens to be empty, packets will have to wait for new tokens, leading to increased latency and potential packet loss.
- Token Bucket happens to be less flexible than leaky bucket when it comes to network traffic shaping. The fixed token generation rate cannot be easily altered to meet changing network requirements, unlike the adaptable nature of leaky bucket.
- The implementation involved in token bucket can be more complex, especially due to the fact that different token generation rates are used for different traffic types. Configuration and management might be more difficult due to this.
- Usage of large bursts of data may lead to inefficient use of bandwidth, and may cause congestion. Leaky bucket algorithm, on the other hand helps prevent congestion by limiting the amount of data sent at any given time, promoting more efficient bandwidth utilization.