Transport Layer responsibilities

The transport Layer is the second layer in the TCP/IP model and the fourth layer in the OSI model. It is an end-to-end layer used to deliver messages to a host. It is termed an end-to-end layer because it provides a point-to-point connection rather than hop-to-hop, between the source host and destination host to deliver the services reliably. The unit of data encapsulation in the Transport Layer is a segment.

Working of Transport Layer

The transport layer takes services from the Application layer and provides services to the Network layer.

At the sender's side: The transport layer receives data (message) from the Application layer and then performs Segmentation, divides the actual message into segments, adds the source and destination's port numbers into the header of the segment, and transfers the message to the Network layer.

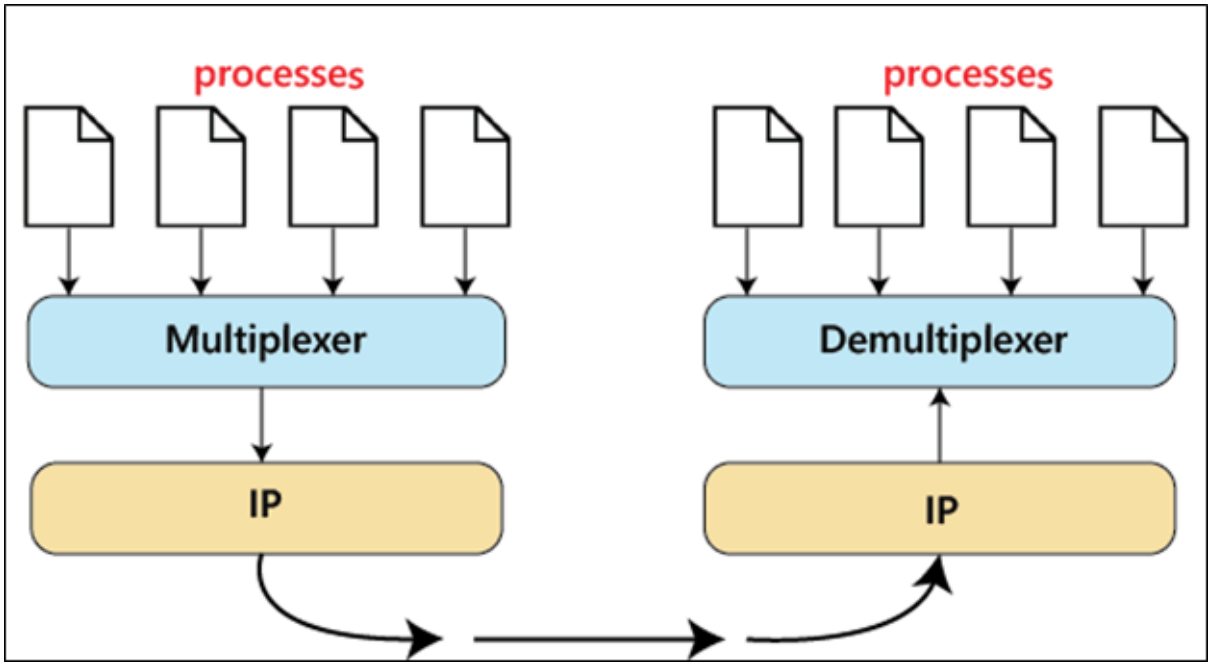
At the receiver's side: The transport layer receives data from the Network layer, reassembles the segmented data, reads its header, identifies the port number, and forwards the message to the appropriate port in the Application layer.

Responsibilities of a Transport Layer

- The Process-to-Process Delivery
- End-to-End Connection between Hosts
- Multiplexing and Demultiplexing
- Congestion Control

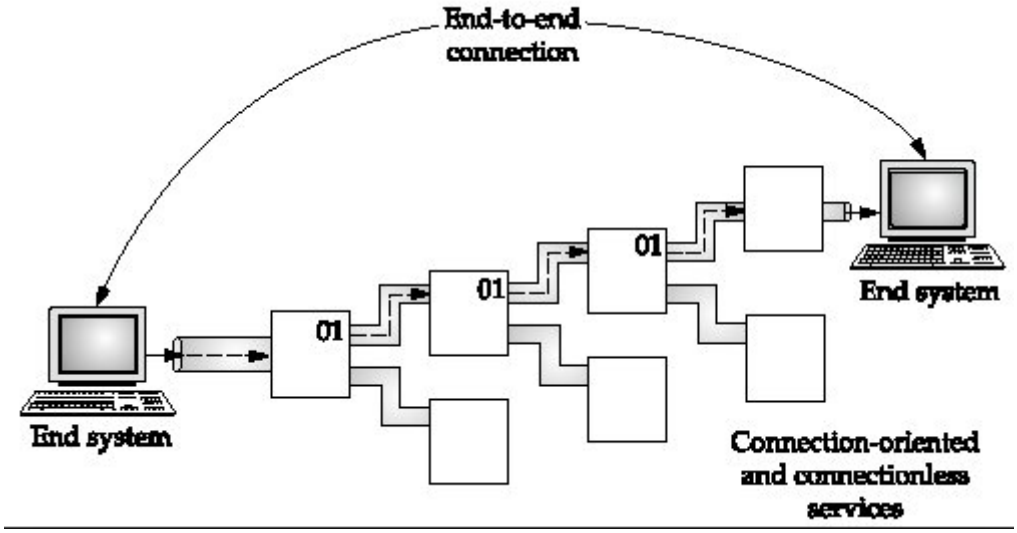
The Process-to-Process Delivery

While Data Link Layer requires the MAC address (48 bits address contained inside the Network Interface Card of every host machine) of source-destination hosts to correctly deliver a frame and the Network layer requires the IP address for appropriate routing of packets, in a similar way Transport Layer requires a Port number to correctly deliver the segments of data to the correct process amongst the multiple processes running on a particular host. A port number is a 16-bit address used to identify any client-server program uniquely.



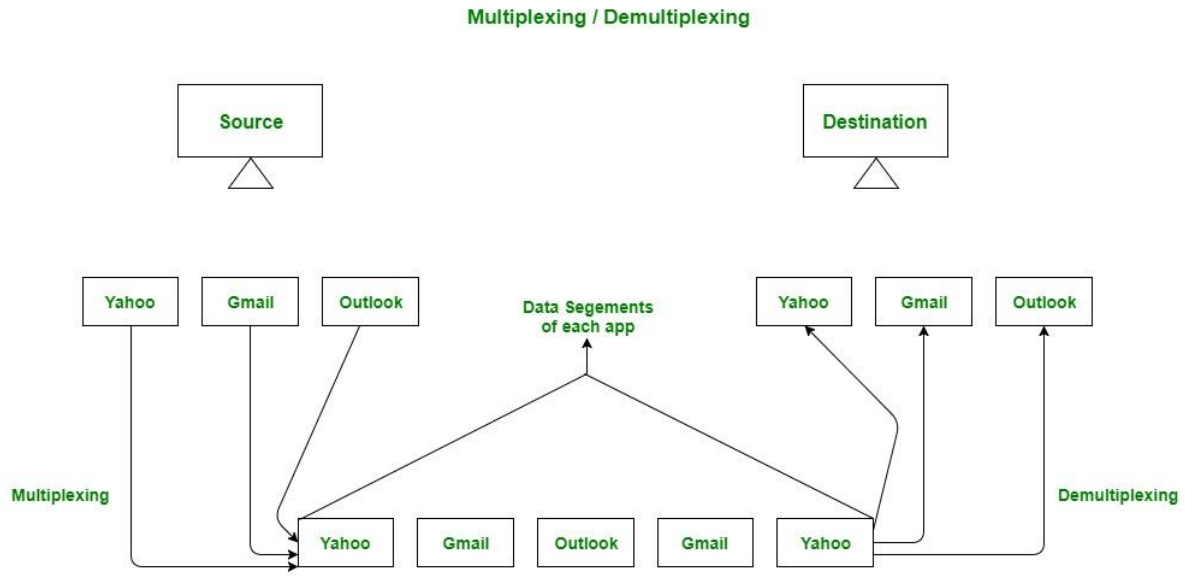
End-to-end Connection between Hosts

The transport layer is also responsible for creating the end-to-end Connection between hosts for which it mainly uses TCP and UDP. TCP is a secure, connection-orientated protocol that uses a handshake protocol to establish a robust connection between two end hosts. TCP ensures the reliable delivery of messages and is used in various applications. UDP, on the other hand, is a stateless and unreliable protocol that ensures best-effort delivery. It is suitable for applications that have little concern with flow or error control and requires sending the bulk of data like video conferencing. It is often used in multicasting protocols.



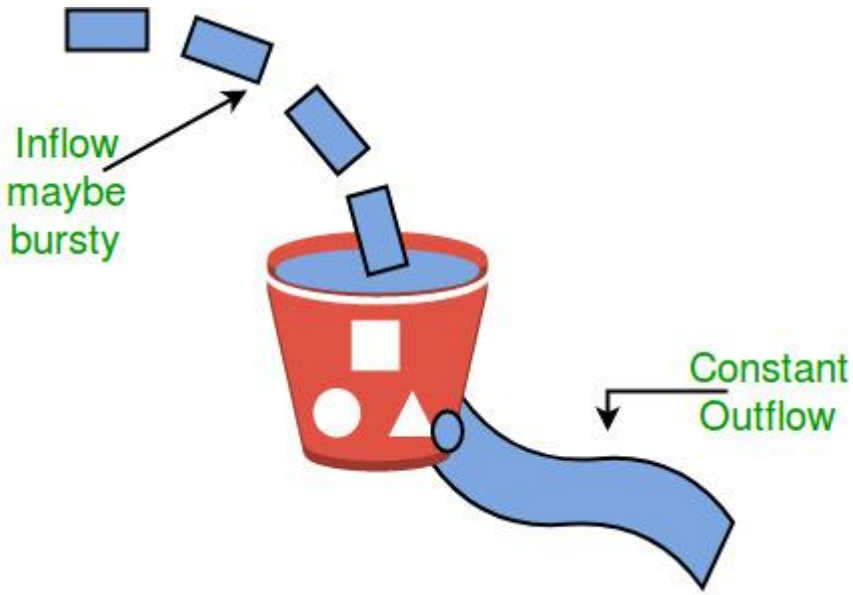
Multiplexing and Demultiplexing

Multiplexing (many to one) is when data is acquired from several processes from the sender and merged into one packet along with headers and sent as a single packet. Multiplexing allows the simultaneous use of different processes over a network that is running on a host. The processes are differentiated by their port numbers. Similarly, Demultiplexing (one to many is required at the receiver side when the message is distributed into different processes. Transport receives the segments of data from the network layer distributes and delivers it to the appropriate process running on the receiver's machine.



Congestion Control

Congestion is a situation in which too many sources over a network attempt to send data and the router buffers start overflowing due to which loss of packets occurs. As a result, the retransmission of packets from the sources increases the congestion further. In this situation, the Transport layer provides Congestion Control in different ways. It uses open-loop congestion control to prevent congestion and closed-loop congestion control to remove the congestion in a network once it occurred.



User Datagram Protocol

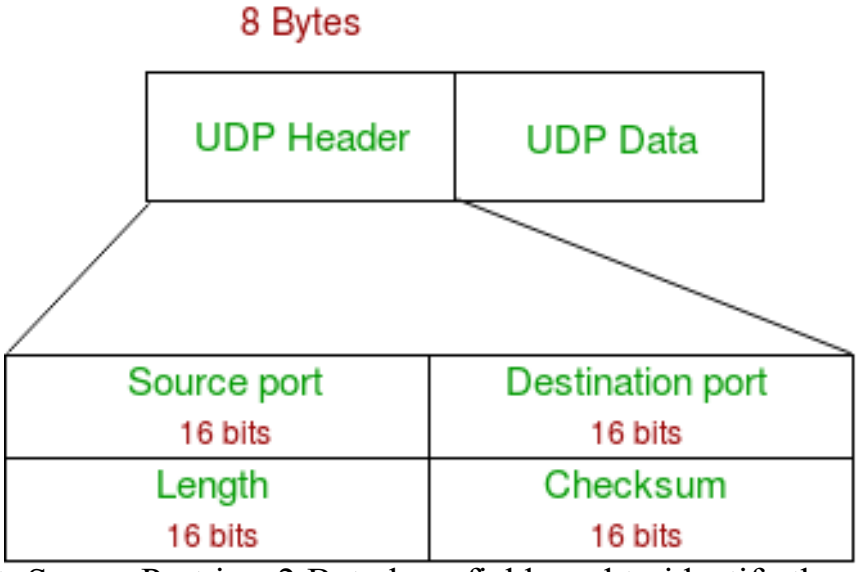
User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency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

Though Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services; provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency. Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth.

User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

UDP Header –

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.



Source Port: Source Port is a 2 Byte long field used to identify the port number of the source.

Destination Port: It is a 2 Byte long field, used to identify the port of the destined packet.

Length: Length is the length of UDP including the header and the data. It is a 16-bits field.

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.

Notes – Unlike TCP, the Checksum calculation is not mandatory in UDP. No Error control or flow control is provided by UDP. Hence UDP depends on IP and ICMP for error reporting. Also UDP provides port numbers so that is can differentiate between users requests.

Applications of UDP:

Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.

It is a suitable protocol for multicasting as UDP supports packet switching.

UDP is used for some routing update protocols like RIP(Routing Information Protocol).

Normally used for real-time applications which can not tolerate uneven delays between sections of a received message.

UDP is widely used in online gaming, where low latency and high-speed communication is essential for a good gaming experience. Game servers often send small, frequent packets of data to clients, and UDP is well suited for this type of communication as it is fast and lightweight.

Streaming media applications, such as IPTV, online radio, and video conferencing, use UDP to transmit real-time audio and video data. The loss of some packets can be tolerated in these applications, as the data is continuously flowing and does not require retransmission.

VoIP (Voice over Internet Protocol) services, such as Skype and WhatsApp, use UDP for real-time voice communication. The delay in voice communication can be noticeable if packets are delayed due to congestion control, so UDP is used to ensure fast and efficient data transmission.

The application layer can do some of the tasks through UDP-

- Trace Route
- Record Route
- Timestamp
- UDP takes a datagram from Network Layer, attaches its header, and sends it to the user. So, it works fast.
- Actually, UDP is a null protocol if you remove the checksum field.
- Reduce the requirement of computer resources.
- When using the Multicast or Broadcast to transfer.
- The transmission of Real-time packets, mainly in multimedia applications.

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.

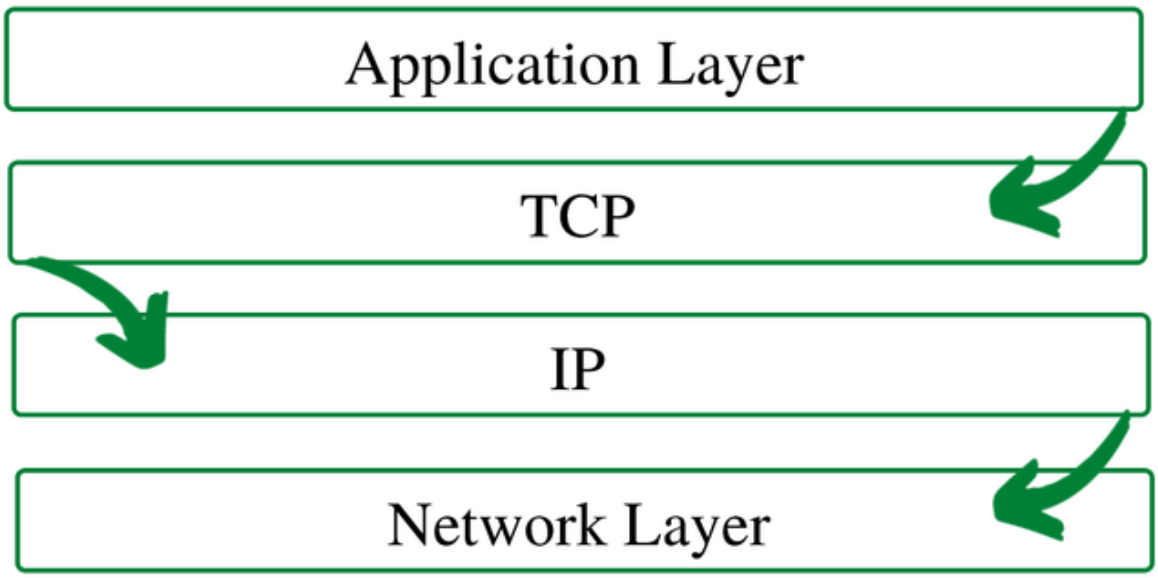
3. No flow control: UDP does not have flow control, which means that it can overwhelm the receiver with packets that it cannot handle.

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.

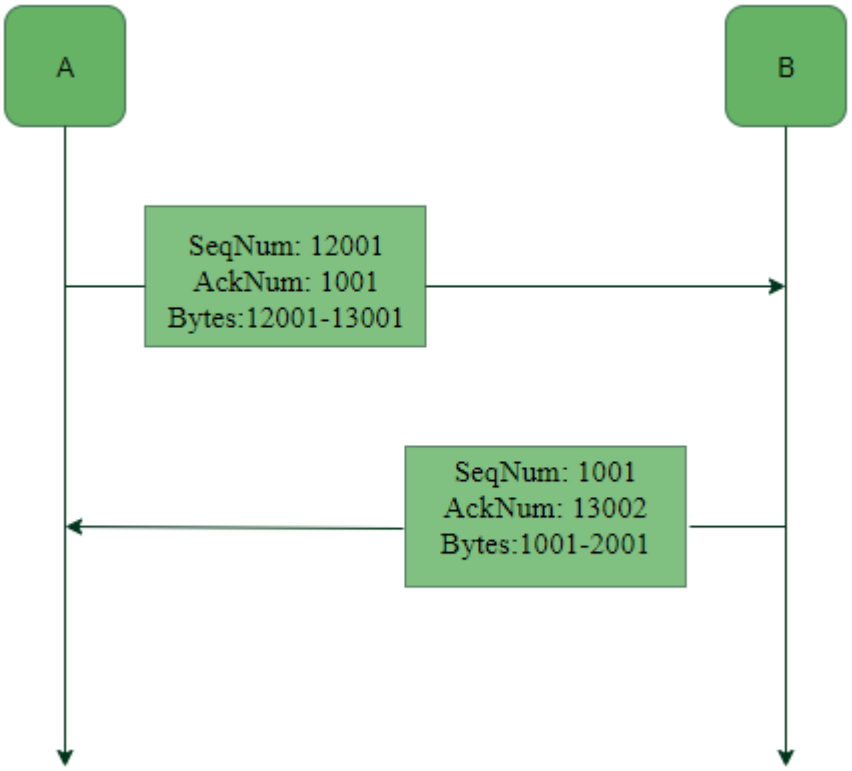
TCP (Transmission Control Protocol)

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.



Byte number, Sequence number and Acknowledgement number:

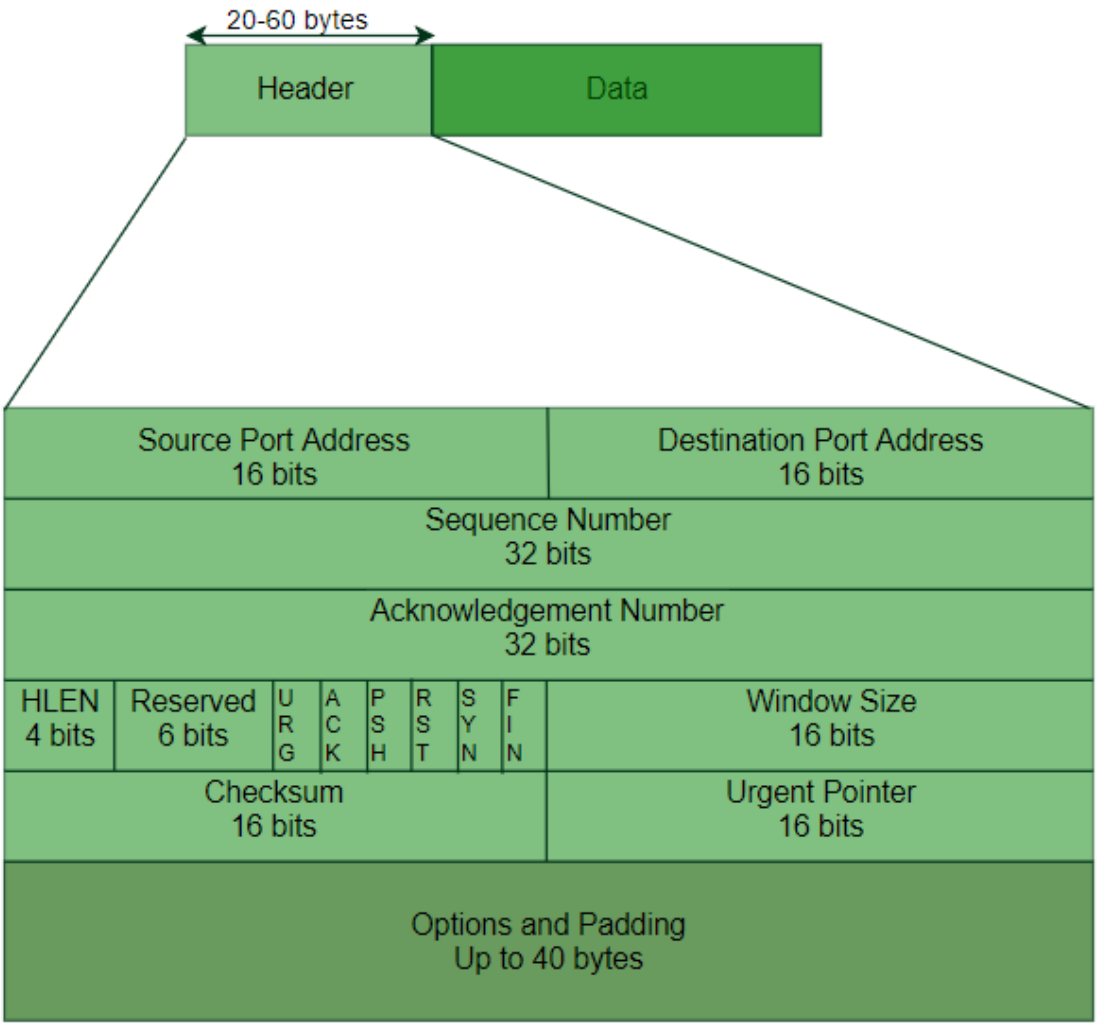
All the data bytes that are to be transmitted are numbered and the beginning of this numbering is arbitrary. Sequence numbers are given to the segments so as to reassemble the bytes at the receiver end even if they arrive in a different order. The sequence number of a segment is the byte number of the first byte that is being sent. The acknowledgement number is required since TCP provides full-duplex service. The acknowledgement number is the next byte number that the receiver expects to receive which also provides acknowledgement for receiving the previous bytes.



In this example we see that A sends acknowledgement number1001, which means that it has received data bytes till byte number 1000 and expects to receive 1001 next, hence B next sends data bytes starting from 1001. Similarly, since B has received data bytes till byte number 13001 after the first data transfer from A to B, therefore B sends acknowledgement number 13002, the byte number that it expects to receive from A next.

TCP Segment structure –

A TCP segment consists of data bytes to be sent and a header that is added to the data by TCP as shown:



The header of a TCP segment can range from 20-60 bytes. 40 bytes are for options. If there are no options, a header is 20 bytes else it can be of upmost 60 bytes.

Header fields:

Source Port Address –

A 16-bit field that holds the port address of the application that is sending the data segment.

Destination Port Address –

A 16-bit field that holds the port address of the application in the host that is receiving the data segment.

Sequence Number –

A 32-bit field that holds the sequence number, i.e, the byte number of the first byte that is sent in that particular segment. It is used to reassemble the message at the receiving end of the segments that are received out of order.

Acknowledgement Number –

A 32-bit field that holds the acknowledgement number, i.e, the byte number that the receiver expects to receive next. It is an acknowledgement for the previous bytes being received successfully.

Header Length (HLEN) –

This is a 4-bit field that indicates the length of the TCP header by a number of 4-byte words in the header, i.e., if the header is 20 bytes (min length of TCP header), then this field will hold 5 (because 5 x 4 = 20) and the maximum length: 60 bytes, then it'll hold the value 15(because 15 x 4 = 60). Hence, the value of this field is always between 5 and 15.

Control flags –

These are 6 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc. Their function is:

URG: Urgent pointer is valid

ACK: Acknowledgement number is valid(used in case of cumulative acknowledgement)

PSH: Request for push

RST: Reset the connection

SYN: Synchronize sequence numbers

FIN: Terminate the connection

Window size –

This field tells the window size of the sending TCP in bytes.

Checksum –

This field holds the checksum for error control. It is mandatory in TCP as opposed to UDP.

Urgent pointer –

This field (valid only if the URG control flag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. The value of this field is added to the sequence number to get the byte number of the last urgent byte.

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.

SCTP stands for Stream Control Transmission Protocol

SCTP stands for Stream Control Transmission Protocol. It is a new reliable, messageoriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced.

SCTP combines the best features of UDP and TCP. SCTP is a reliable message-oriented protocol. It preserves the message boundaries, and at the same time, detects lost data, duplicate data, and out-of-order data. It also has congestion control and flows control mechanisms.

Features of SCTP

There are various features of SCTP, which are as follows −

Transmission Sequence Number

The unit of data in TCP is a byte. Data transfer in TCP is controlled by numbering bytes by using a sequence number. On the other hand, the unit of data in SCTP is a DATA chunk that may or may not have a one-to-one relationship with the message coming from the process because of fragmentation.

Stream Identifier

In TCP, there is only one stream in each connection. In SCTP, there may be several streams in each association. Each stream in SCTP needs to be identified by using a stream identifier (SI). Each data chunk must carry the SI in its header so that when it arrives at the destination, it can be properly placed in its stream. The 51 is a 16-bit number starting from O.

Stream Sequence Number

When a data chunk arrives at the destination SCTP, it is delivered to the appropriate stream and in the proper order. This means that, in addition to an SI, SCTP defines each data chunk in each stream with a stream sequence number (SSN).

Packets

In TCP, a segment carries data and control information. Data is carried as a collection of bytes; control information is defined by six control flags in the header. The design of SCTP is totally different: data is carried as data chunks; control information is carried as control chunks.

Flow Control

Like TCP, SCTP implements flow control to avoid overwhelming the receiver.

Error Control

Like TCP, SCTP implements error control to provide reliability. TSN numbers and acknowledgement numbers are used for error control.

Congestion Control

Like TCP, SCTP implements congestion control to determine how many data chunks can be injected into the network.

What is Network Traffic?

In computing, network traffic is the amount of data sent and received by a computer over a network. It can be measured in terms of bitrate or bandwidth. Network traffic is often classified into different categories, such as web traffic, email traffic, file transfer traffic, and so on.

Some common causes of high network traffic include downloading large files, streaming video or audio, and visiting websites with lots of images. Traffic can also be caused by malicious software, such as viruses and worms, that propagate themselves by sending copies to other computers on the network.

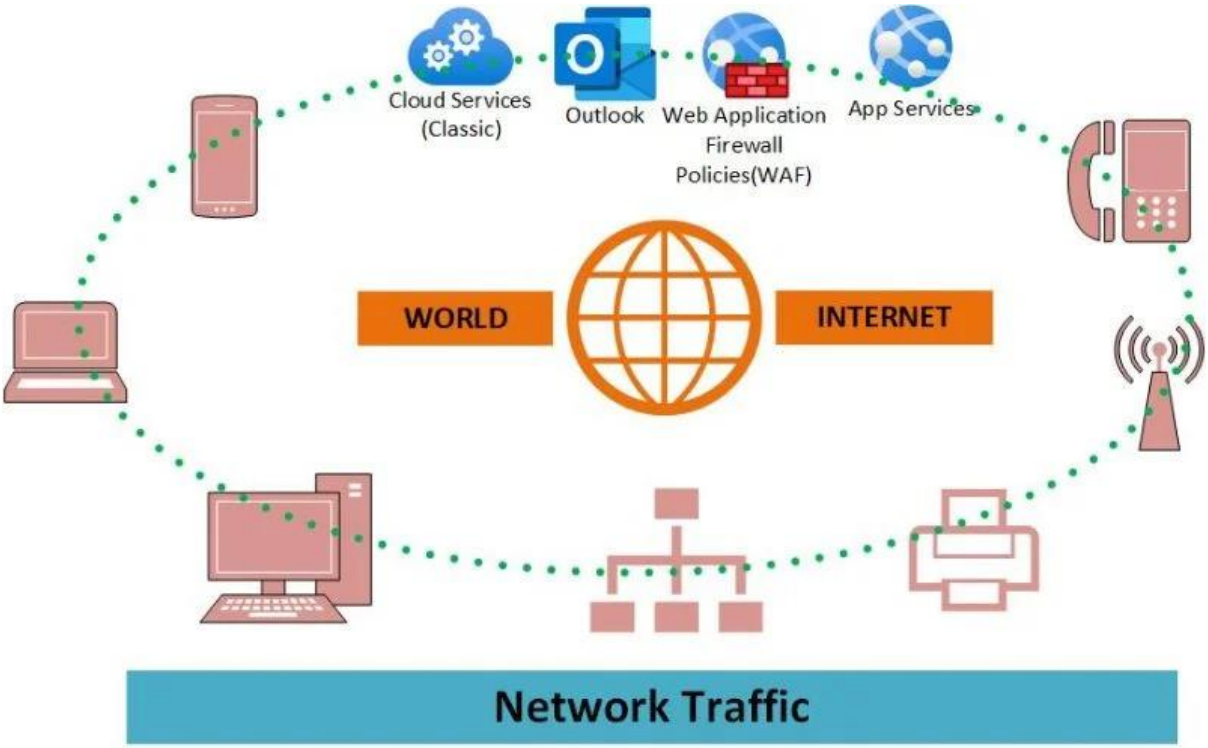
Types of Network Traffic

There are two main types of network traffic: unicast and multicast.

Unicast traffic is sent from one computer to another specific computer. This is the most common type of traffic on the Internet and includes things like web browsing, email, and file transfers.

Multicast traffic is sent from one computer to a group of computers. This is often used for streaming media, such as audio or video, and can be inefficient if there are a lot of members in the group who are not interested in the data being sent.

How does Network Traffic Flow?



Network traffic typically flows in one of two directions: ingress or egress.

Ingress traffic is data that is coming into a network from the outside.

For example, when you visit a website, the data that makes up the website (known as web traffic) flows into your computer from the server that hosts the website.

Egress traffic is data that is flowing out of a network to the outside.

For example, when you send an email, the data that makes up the email (known as email traffic) flows out of your computer and onto the Internet to the recipient's computer.

Some types of traffic can flow in both directions at the same time. For example, when you are browsing the web, you are both downloading data (ingress traffic) and sending data (egress traffic).

Congestion

Congestion causes choking of the communication channel. When too many packets are displayed in a part of the subnet, the subnet's performance degrades. Hence, the network's communication channel is called congested if packets are traversing the path experience primarily over the path propagation delay.

It is known as heavily congested when the packets never reach the destination, denoting the delay method infinity. When the input traffic rate exceeds the output lines capacity, the subnet's input part gets choked and generates congestion. It also happens when the routers are too slow to execute queuing buffers, refreshing tables, etc. The loss of capacity of the routers' buffer is also one of the many factors for congestion. However, enhancing the memory of the router may be helpful up to a certain point.

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.

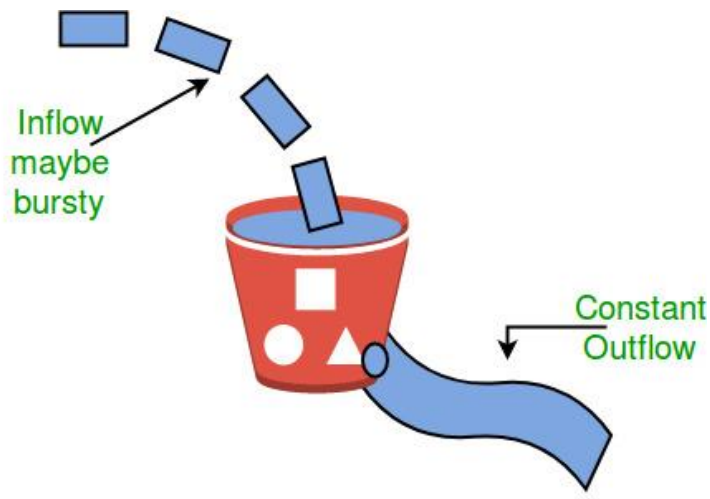
There are two congestion control algorithm which are as follows:

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:

- When host wants to send packet, packet is thrown into the bucket.
- The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
- Bursty traffic is converted to a uniform traffic by the leaky bucket.
- 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:

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

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.