DCN - TRANSMISSION CONTROL PROTOCOL

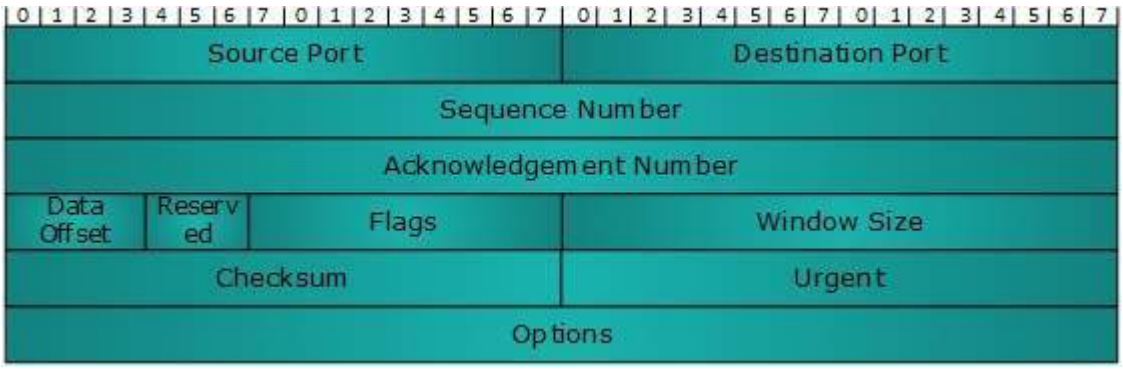
The transmission Control Protocol *TCP* is one of the most important protocols of Internet Protocols suite. It is most widely used protocol for data transmission in communication network such as internet.

Features

- TCP is reliable protocol. That is, the receiver always sends either positive or negative acknowledgement about the data packet to the sender, so that the sender always has bright clue about whether the data packet is reached the destination or it needs to resend it.
- TCP ensures that the data reaches intended destination in the same order it was sent.
- TCP is connection oriented. TCP requires that connection between two remote points be established before sending actual data.
- TCP provides error-checking and recovery mechanism.
- TCP provides end-to-end communication.
- TCP provides flow control and quality of service.
- TCP operates in Client/Server point-to-point mode.
- TCP provides full duplex server, i.e. it can perform roles of both receiver and sender.

Header

The length of TCP header is minimum 20 bytes long and maximum 60 bytes.



- **Source Port** 16 – *bits* - It identifies source port of the application process on the sending device.
- **Destination Port** 16 – *bits* - It identifies destination port of the application process on the receiving device.
- **Sequence Number** 32 – *bits* - Sequence number of data bytes of a segment in a session.
- **Acknowledgement Number** 32 – *bits* - When ACK flag is set, this number contains the next sequence number of the data byte expected and works as acknowledgement of the previous data received.
- **Data Offset** 4 – *bits* - This field implies both, the size of TCP header 32 – *bitwords* and the offset of data in current packet in the whole TCP segment.
- **Reserved** 3 – *bits* - Reserved for future use and all are set zero by default.
- **Flags** 1 – *biteach*

- **NS** - Nonce Sum bit is used by Explicit Congestion Notification signaling process.
- **CWR** - When a host receives packet with ECE bit set, it sets Congestion Windows Reduced to acknowledge that ECE received.
- **ECE** -It has two meanings:
  - If SYN bit is clear to 0, then ECE means that the IP packet has its CE *congestionexperience* bit set.
  - If SYN bit is set to 1, ECE means that the device is ECT capable.
- **URG** - It indicates that Urgent Pointer field has significant data and should be processed.
- **ACK** - It indicates that Acknowledgement field has significance. If ACK is cleared to 0, it indicates that packet does not contain any acknowledgement.
- **PSH** - When set, it is a request to the receiving station to PUSH data *assoonasitcomes* to the receiving application without buffering it.
- **RST** - Reset flag has the following features:
  - It is used to refuse an incoming connection.
  - It is used to reject a segment.
  - It is used to restart a connection.
- **SYN** - This flag is used to set up a connection between hosts.
- **FIN** - This flag is used to release a connection and no more data is exchanged thereafter. Because packets with SYN and FIN flags have sequence numbers, they are processed in correct order.

- **Windows Size** - This field is used for flow control between two stations and indicates the amount of buffer *inbytes* the receiver has allocated for a segment, i.e. how much data is the receiver expecting.
- **Checksum** - This field contains the checksum of Header, Data and Pseudo Headers.
- **Urgent Pointer** - It points to the urgent data byte if URG flag is set to 1.
- **Options** - It facilitates additional options which are not covered by the regular header. Option field is always described in 32-bit words. If this field contains data less than 32-bit, padding is used to cover the remaining bits to reach 32-bit boundary.

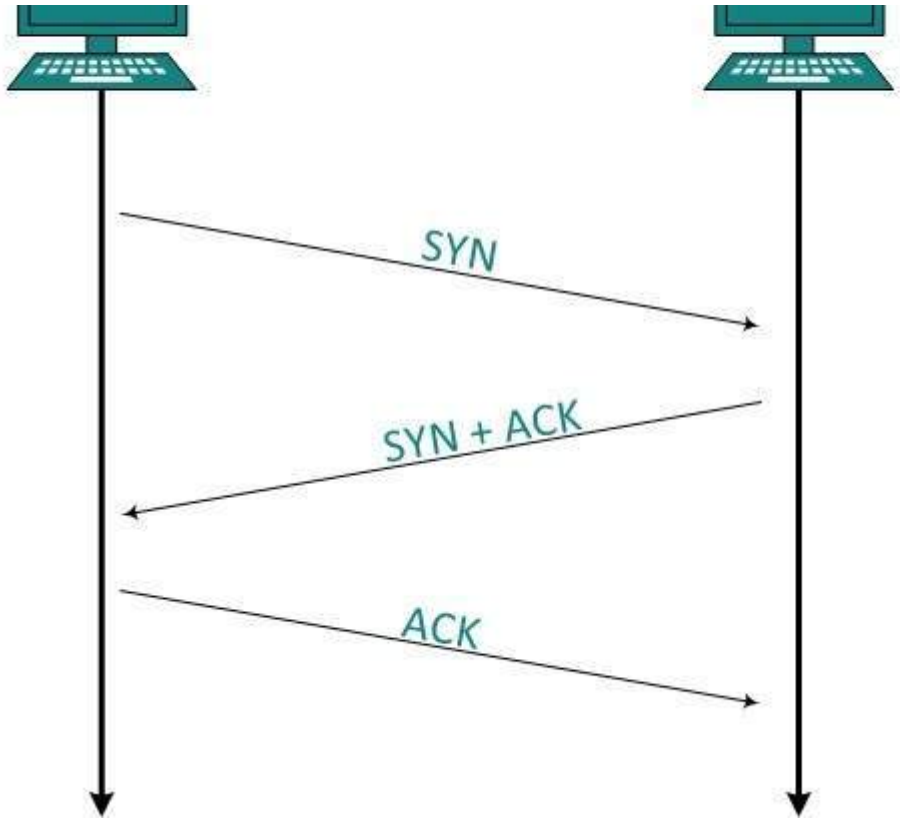
Addressing

TCP communication between two remote hosts is done by means of port numbers *TSAPs*. Ports numbers can range from 0 – 65535 which are divided as:

- System Ports 0– 1023
- User Ports 1024– 49151
- Private/Dynamic Ports 49152– 65535

Connection Management

TCP communication works in Server/Client model. The client initiates the connection and the server either accepts or rejects it. Three-way handshaking is used for connection management.



Establishment

Client initiates the connection and sends the segment with a Sequence number. Server acknowledges it back with its own Sequence number and ACK of client's segment which is one more than client's Sequence number. Client after receiving ACK of its segment sends an acknowledgement of Server's response.

Release

Either of server and client can send TCP segment with FIN flag set to 1. When the receiving end responds it back by ACKnowledging FIN, that direction of TCP communication is closed and connection is released.

Bandwidth Management

TCP uses the concept of window size to accommodate the need of Bandwidth management. Window size tells the sender at the remote end, the number of data byte segments the receiver at this end can receive. TCP uses slow start phase by using window size 1 and increases the window size exponentially after each successful communication.

For example, the client uses windows size 2 and sends 2 bytes of data. When the acknowledgement of this segment received the windows size is doubled to 4 and next sent the segment sent will be 4 data bytes long. When the acknowledgement of 4-byte data segment is received, the client sets windows size to 8 and so on.

If an acknowledgement is missed, i.e. data lost in transit network or it received NACK, then the window size is reduced to half and slow start phase starts again.

Error Control ∧ Flow Control

TCP uses port numbers to know what application process it needs to handover the data segment. Along with that, it uses sequence numbers to synchronize itself with the remote host. All data segments are sent and received with sequence numbers. The Sender knows which last data segment was received by the Receiver when it gets ACK. The Receiver knows about the last segment sent by the Sender by referring to the sequence number of recently received packet.

If the sequence number of a segment recently received does not match with the sequence number the receiver was expecting, then it is discarded and NACK is sent back. If two segments arrive with

the same sequence number, the TCP timestamp value is compared to make a decision.

Multiplexing

The technique to combine two or more data streams in one session is called Multiplexing. When a TCP client initializes a connection with Server, it always refers to a well-defined port number which indicates the application process. The client itself uses a randomly generated port number from private port number pools.

Using TCP Multiplexing, a client can communicate with a number of different application process in a single session. For example, a client requests a web page which in turn contains different types of data $HTTP$, $SMTP$, $FTP etc.$ the TCP session timeout is increased and the session is kept open for longer time so that the three-way handshake overhead can be avoided.

This enables the client system to receive multiple connection over single virtual connection. These virtual connections are not good for Servers if the timeout is too long.

Congestion Control

When large amount of data is fed to system which is not capable of handling it, congestion occurs. TCP controls congestion by means of Window mechanism. TCP sets a window size telling the other end how much data segment to send. TCP may use three algorithms for congestion control:

- Additive increase, Multiplicative Decrease
- Slow Start
- Timeout React

Timer Management

TCP uses different types of timer to control and management various tasks:

Keep-alive timer:

- This timer is used to check the integrity and validity of a connection.
- When keep-alive time expires, the host sends a probe to check if the connection still exists.

Retransmission timer:

- This timer maintains stateful session of data sent.
- If the acknowledgement of sent data does not receive within the Retransmission time, the data segment is sent again.

Persist timer:

- TCP session can be paused by either host by sending Window Size 0.
- To resume the session a host needs to send Window Size with some larger value.
- If this segment never reaches the other end, both ends may wait for each other for infinite time.
- When the Persist timer expires, the host re-sends its window size to let the other end know.
- Persist Timer helps avoid deadlocks in communication.

Timed-Wait:

- After releasing a connection, either of the hosts waits for a Timed-Wait time to terminate the connection completely.
- This is in order to make sure that the other end has received the acknowledgement of its

connection termination request.

- Timed-out can be a maximum of 240 seconds $4minutes$.

Crash Recovery

TCP is very reliable protocol. It provides sequence number to each of byte sent in segment. It provides the feedback mechanism i.e. when a host receives a packet, it is bound to ACK that packet having the next sequence number expected $ifitisnotthelastsegment$.

When a TCP Server crashes mid-way communication and re-starts its process it sends TPDU broadcast to all its hosts. The hosts can then send the last data segment which was never unacknowledged and carry onwards.