Transport Layer Introduction

Introduction

Next Layer in OSI Model is recognized as Transport Layer *Layer*−4. All modules and procedures pertaining to transportation of data or data stream categorized into this layer. As all other layers, this layer speaks to its peer Transport layer of the remote host.

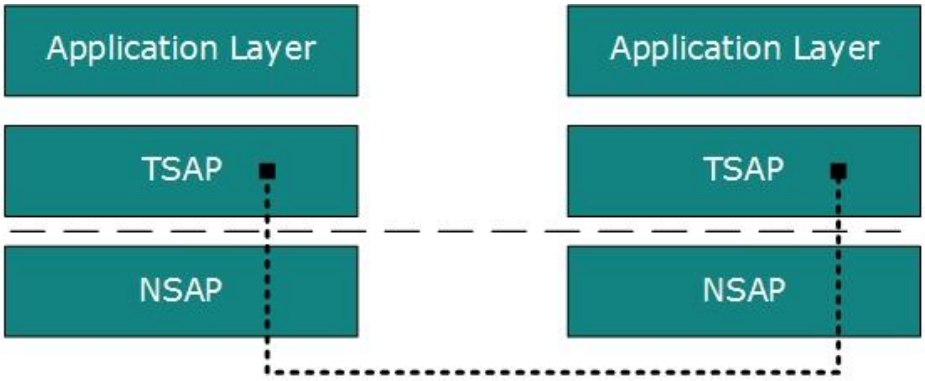
Transport layer offers peer-to-peer and end-to-end connection between two processes on remote hosts. Transport layer takes data from upper layer *i.e. Application layer* and then breaks it into smaller size segments numbers each byte and hands over to lower layer *Network Layer* for delivery.

Functions

- This Layer is the first one which breaks the information data, supplied by Application layer in to smaller units called segments. It numbers every byte in the segment and maintains their accounting.
- This layer ensures that data must be received in the same sequence in which it was sent.
- This layer provides end-to-end delivery of data between host which may or may not belong to the same subnet.
- All server processes intend to communicate over the network are equipped with well-known TSAPs *Transport Service Access Point* also known as port numbers.

End-to-end communication

A process on one host identifies its peer host on remote host by means of Transport Service Access Points, also known as Port numbers. TSAPs *Ports* are very well defined and a process which is trying to communicate with its peer knows this in advance.



[*Image: Transport Layer | TSAP*]

For example, when a DHCP client wants to communicate with remote DHCP server, it always request on port number 67. When a DNS client wants to communicate with remote DNS server it always requests on port number 53 *UDP*.

Two main Transport layer protocols are:

- **Transmission Control Protocol**

  Provides reliable communication between two hosts.

- **User Datagram Protocol**

  Provides unreliable communication between two hosts.

Transmission Control Protocol

Introduction

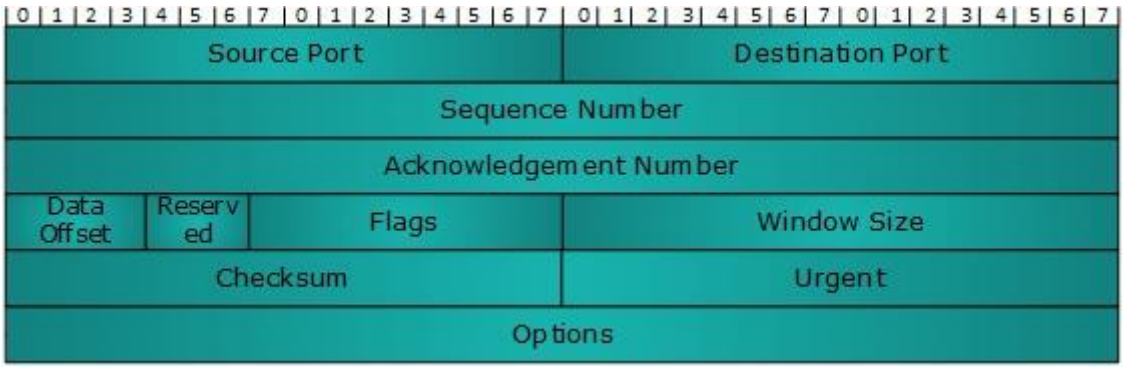
TCP is one of the most important protocols of Internet Protocols suite. It is most widely used protocol for data transmission in communication network such as Internet.

Features

- TCP is reliable protocol, that is, the receiver sends an acknowledgement back to the sender, of each packet it receives. Sender is now confirmed that packet has been received and can process further packets in its queue.
- TCP ensures that data has been received in the 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 act like receiver and sender.

Header

TCP header at minimum is 20 bytes long and maximum 60 bytes.



[*Image: TCP Header*]

- **Source Port 16−*bits*:** Identifies source port of the application process on the sending device.
- **Destination Port 16−*bits*:** 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 expect and works as acknowledgement of the previous data received.
- **Data Offset 4−*bits*:** This field contains two meaning. First, it tells the size of TCP header 32−*bitwords* Secondly, it indicates 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:** has two meaning:
        - If SYN bit is clear to 0, then ECE means that the IP packet has its CE *congestion experience* bit set.
        - If SYN bit is set to 1, ECE means that the device is ECT capable
    - **URG:** indicates that Urgent Pointer field has significant data and should be processed.
    - **ACK:** 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 *as soon as it comes* to the receiving application without buffering it.
    - **RST:** Reset flag has many 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 *in bytes* 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:** Points to the urgent data byte if URG flag is set to 1.
- **Options:** 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.

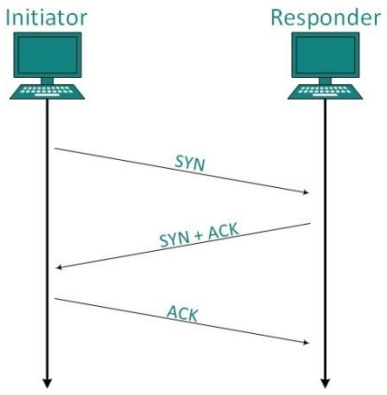
Addressing:

TCP communication between two remote hosts is done by means of port numbers *Transport Service Access Points*. Ports numbers can range from 0 – 65535 which are known 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.



[*Image: Three-way handshake*]

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 *client'sSequencenumber*+1. 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 *the remote end*, how much data byte segment the receiver *this end* can receive. TCP uses slow start phase by using window size 1 increases the window size exponentially after each successful communication.

For example: 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 segment will be sent of 4 data bytes. When the acknowledgement of 4-byte data segment is received client sets windows size to 8 and so on.

If an acknowledgement is missed, i.e. data lost in transit network or it receives NACK 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 what last segment was sent by the Sender looking at 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 it is discarded and NACK is sent back. If two segments arrive with same sequence number, the TCP timestamp value is compared to make a decision.

Multiplexing:

The technique to combine two or more data stream 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 processes in a single session. For example, a client requests a web page which in turn contains different type of data *HTTP*, *SMTP*, *FTPetc*. 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 connections 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 such amount of data, 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 different type of 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 state-ful 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 host 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 4 *minutes*.

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 *if it is not the last segment*.

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 lasts data segment which was never unacknowledged and carry onwards.

User Datagram Protocol

Introduction

UDP is simplest Transport Layer communication protocol available of the TCP/IP protocol suite. It involves minimum amount of communication mechanism. UDP is said to be an unreliable transport protocol but it uses IP services which provides best effort delivery mechanism.

In UDP, the receiver does not generate an acknowledgement of packet received and in turn, the sender does not wait for any acknowledgement of packet sent. This feature makes this unreliable as well as easier on processing.

Requirement:

Why do we need an unreliable protocol to transport data? We deploy UDP where the acknowledgement packets share significant amount of bandwidth with the actual data. Say for example, in Video streaming thousands of packets are forwarded towards its users. Acknowledging all the packets is troublesome and may contain huge amount of bandwidth wastage. The best delivery mechanism of underlying IP protocol ensures best efforts to deliver

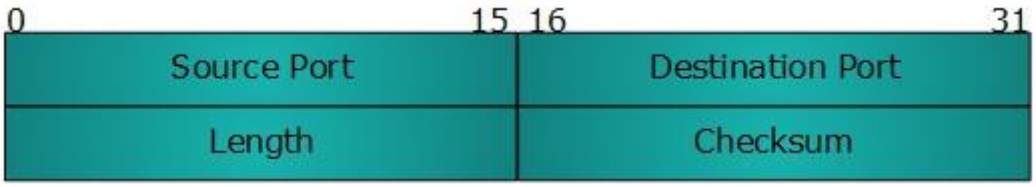
its packets, but even if some packets in video streaming get lost, the impact is not huge and can be ignored easily. Loss of few packets in video and voice traffic sometime goes unnoticed.

Features:

- UDP is used when acknowledgement of data does not hold any significance.
- UDP is good protocol for data flowing in one direction.
- UDP is simple and suitable for query based communications.
- UDP is not connection oriented.
- UDP does not provide congestion control mechanism.
- UDP does not guarantee ordered delivery of data.
- UDP is stateless.
- UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming.

UDP Header:

UDP header is as simple as its function



[*Image: UDP Header*]

UDP header contains four main parameters:

- **Source Port:** This 16 bits information is used to identify the source port of the packet.
- **Destination Port:** This is also 16 bits information, which is used identify application level service on destination machine.
- **Length:** Length field specifies the entire length of UDP packet (including header). It is 16-bits field and minimum value is 8-byte, i.e. the size of UDP header itself.
- **Checksum:** This field stores the checksum value generated by the sender before sending. IPv4 has this field as optional so when checksum field does not contain any value is made 0 and all its bits are set to zero.

UDP application: Here are few applications as example, which uses UDP to transmit data:

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