Chapter 6

Transport Layer

Transport layer provides services of session management, connection establishment release, multiplexing and flow control. Transport layer is responsible for end to end delivery of data. TCP and UDP protocols are transport level protocols responsible for delivery of a message from a process to another process. A new transport layer protocol, SCTP has been devised for some new applications such as IP telephony. In this chapter, we take a overview of these protocols and analyze their loopholes for security purpose.

6.1 Transport Control Protocol(TCP)

TCP provides a connection oriented, reliable, byte stream service. The term connection-oriented means the two applications using TCP must establish a TCP connection with each other before they can exchange data and transmit it on the complition of data exchange. It is a full duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction. TCP includes a flow-control mechanism for each of these byte streams that allow the receiver to limit how much data the sender can transmit. TCP also implements a congestion-control mechanism for smooth flow of byte streams.

6.1.1 TCP Services

The task of transport layer is to provide reliable, cost effective transport of data from source to destination. TCP provides the following services to the layer:

Stream Data Transfer: From the application's viewpoint, TCP transfers a continuous stream of bytes by grouping the bytes in TCP segments, which are passed to IP for transmission to the destination. TCP itself decides how to segment the data and it may forward the data at its own convenience.

Reliability: TCP assigns a sequence number to each byte transmitted and expects a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received

within a timeout interval, the data is retransmitted. The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order and to eliminate duplicate segments.

Flow Control: The receiving TCP, when sending an ACK back to the sender, indicates to the sender the number of bytes it can receive beyond the last received TCP segment, without causing overflow in its internal buffers. This ACK is sent in the form of the highest sequence number it can receive without problems.

Multiplexing: To allow many processes within a single host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenation with the network and host addresses from the internet communication layer, forms a socket. A pair of sockets uniquely identifies each connection.

Logical Connections: The reliability and flow control mechanisms described above require that TCP initializes and maintains certain status information for each data stream. The combination of this status, including sockets, sequence numbers and window sizes, is called a logical connection. Each connection is uniquely identified by the pair of sockets used by the sending and receiving processes.

Full Duplex: TCP provides concurrent data streams in both directions.

6.1.2 TCP Packet Format

Transport layer provides a reliable, connection-oriented transport service to the upper layer protocols. The TCP header is shown as in figure 6.1

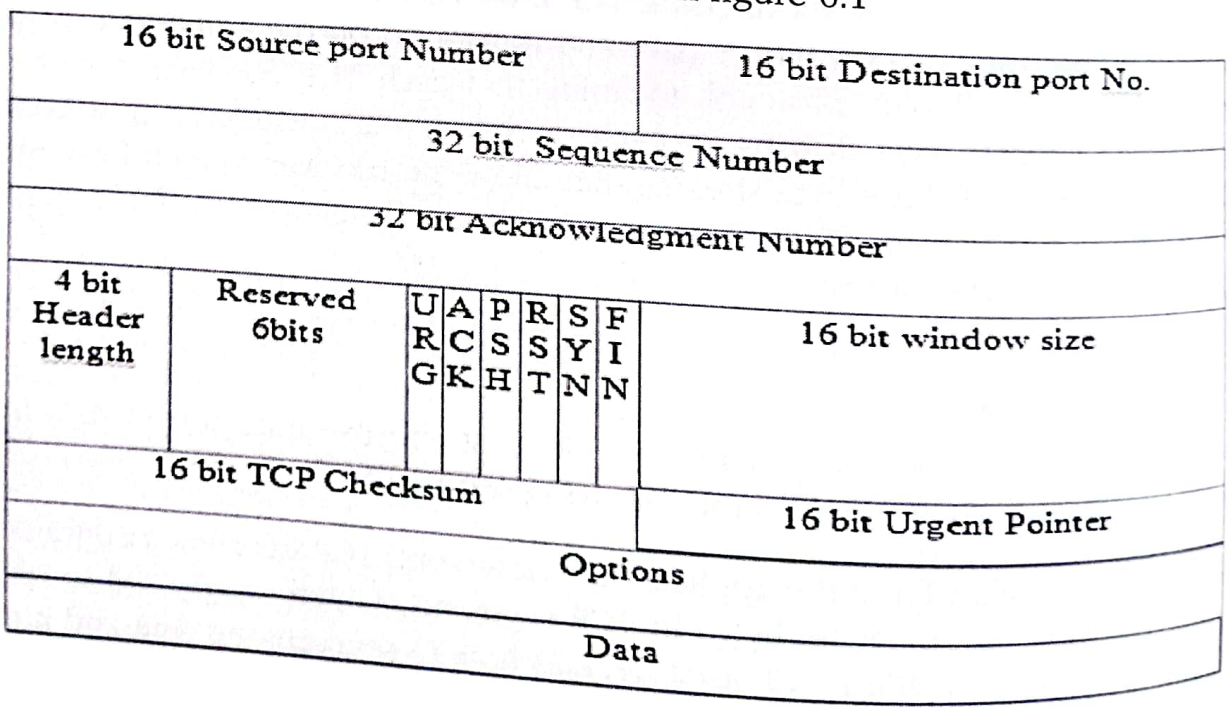

Figure 6.1 TCP Header

TCP headers should perform various tasks as follows:

Source Port: On the sending system, a process is being bounded by the source port. A hash between the IP addresses, destination and source ports is used for uniqueness for binding single application or program.

Destination Port: Destination port was directly bounded to a process at the receiving system.

Sequence Number: A number on every packet of TCP which facilitates the TCP stream properly sequenced. The port can then return an acknowledgment after the packet is properly received.

Acknowledge Number: This number is used when a packet at the host is received.

Data Offset: The distance of the TCP header and the location of data part of the packet is indicated by data offset.

Reserved bit: The reserved bits are reserved for future usage.

CWR bit: An added bit to RFC 3268 which is used by ECN. CWR bit is used for sending data to inform the receiving part when the congestion window reduced.

ECE bit: An added bit to RFC 3268 which is used by ECN. TCP/IP stack uses this bit on the receiver host for sending the host that is has received a CE packet.

URG bit: URG bit is used to determine the usage of Urgent Pointer Field. 1 is set as to use Urgent pointer and 0 is set not to use Urgent pointer.

ACK bit: A packet is set by this bit that indicates the reply to another packet that data is received.

PSH bit: To communicate any intermediate hosts for sending data on to the actual user.

RST bit: To indicate to the other end for tearing down the TCP connection, Reset bit is used.

SYN bit: When the connection is initially established, SYN is used. The initial packet and the reply SYN are the two instances of the connection.

FIN bit: The host sends FIN bit which indicates that no more data is left for sending. The other end will respond a FIN when there is no data left.

Window bit: Window bit information is used by the host for informing the sender the volume of data the receiver permits a given point of time.

Checksum bit: This performs checksum on the whole TCP header. It is a one's complement of the one's complement sum of every 16 bit word available in the header. The checksum field is set to zero.

Urgent Pointer bit: A pointer bit that point to the end of the data which is considered as urgent.

Options bit: A variable length field which contains optional headers which may be used. It contains an initial field – the length of options field, second field - informs which options are used.

Padding bit: Until the header ends at a 32-bit boundary, the padding of TCP header takes place. The padding always consists of only zeros

Example 6.1

following is dump of TCP header in Hex

05 cf 01 bb a0 a7 be 3b 50 80 8e b6 50 10 fe 6e 01 fe 00 00

i) What is the source port no?

ii) What is the destination port no?

iii) What is the sequence no?

iv) What is the acknowledgement no?

v) What is the length of the header?

vi) What is the type of the segment?

vii) What is the window size?

Solution

Given TCP segment is

05 cf 01 bb a0 a7 be 3b 50 80 8e b6 50 10 fe 6e 01 fe 00 00

(i) Source port number is given by the first two bytes of the TCP segment.

In our segment, Source port number = $(05cf)_{16}$

= $(1487)_{10}$

(ii) Destination port number is given by the 3rd & 4th bytes of the TCP segment.

In our segment, Destination port number = $(01bb)_{16}$

= $(443)_{10}$

(iii) The sequence number is given by the four bytes from 5th to 8th in the TCP segment.

In our segment, Sequence number = a0 a7 be 3b

∴ Sequence number = 10100000 10100111 10111110 00111011= 886

(iv) The acknowldgement number is given by the 9th to 12th bytes of the TCP segment.

In our segment, Acknowledgment number = $(50\ 80\ 8e\ b6)_{16}$
=01010000 10000000 1001110 10110110

=1853

(v) The length of the header is given by the first half of 13th byte of the TCP segment.

In our segment, ... HLEN = (5)16

=5*4

∴ Length of header=20 bytes

(vi) The type of segment is given by the six flags in the last six bits of the 14th byte TCP segment. Each flag defines its own meaning.

In our segment the 14th bytes is $(02)_{16}$

$=(0000\ 0010)_{16}$

∴ The last second bit is set which means that the synchronization flag is set. The type of service is synchronization.

vii) The window size is given by the 15th & 16th byte of the TCP segment.

In our segment,

∴ Window = $(fe\ 6e)_{16}$

$=(1111\ 1110\ 0110\ 1110)$

$=(65134)_{bytes}$

∴ Window size = 65134 bytes

6.1.3 TCP operation

The main purpose of TCP is error recovery and flow control.

6.1.3.1 Three Way Handshake Algorithm

During connection establishment server and client agree upon sequence and acknowledgment numbers. Implicitly client also notifies server of its source port. Sequence is a characteristic of TCP data segment. Sequence starts with a random number and then each time a new packet is sent, sequence is incremented by the number of bytes sent in the previous TCP segment. Acknowledgment segment is almost the same but from the receiver side. It does not contain data and is equal to sender's sequence number incremented by the number of bytes received. ACK segment acknowledges that host has received sent data.

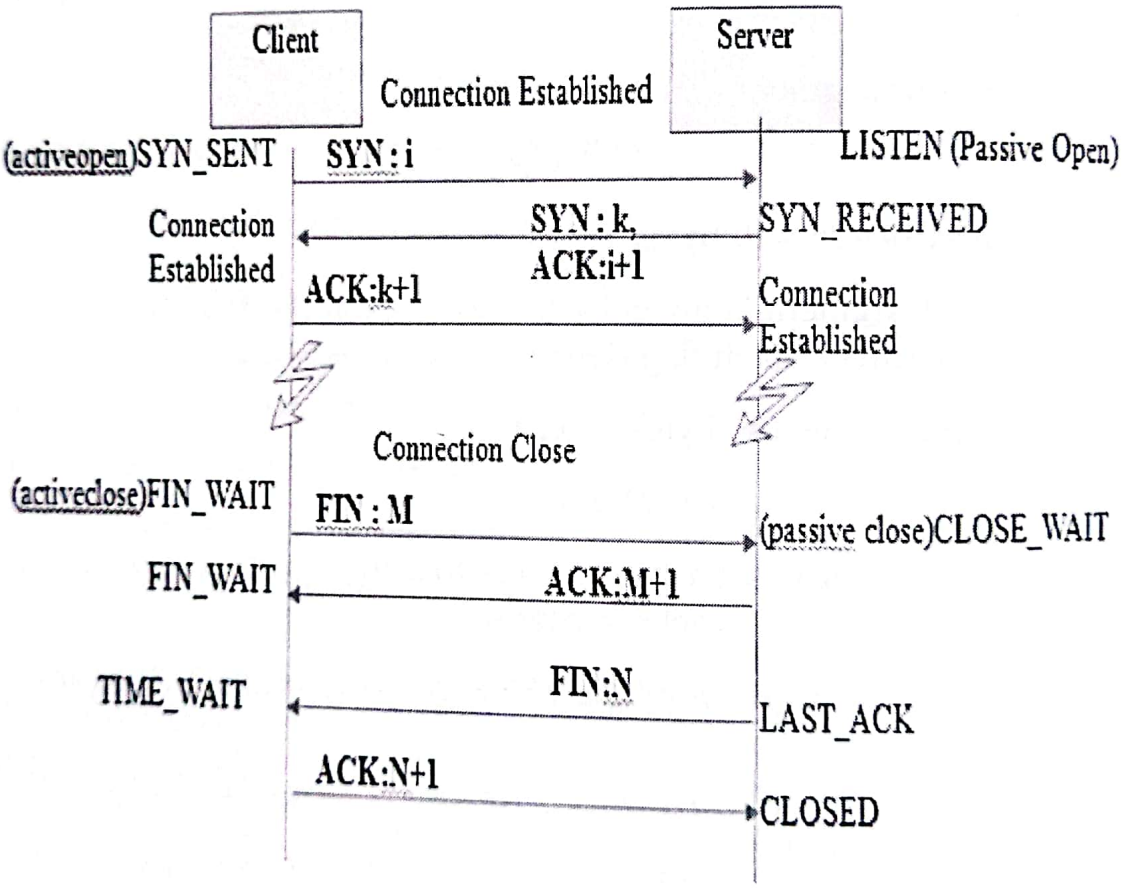

Figure 6.2 Three Way Handshake

As shown in figure 6.2 client-server handshake is performed in three steps:

1. Client sends packet to the server with the SYN flag set, indicating that it is willing to establish a connection. Client sets its sequence to a random number and sends the segment to the server.
2. Server acknowledges that it agrees to establish connection, sets its sequence to a random number, acknowledgment to the client sequence + 1 and send them to the client.

3. In the third message client sets its acknowledgment to the server's sequence + 1 and send back to the server.

Now when both client and server know each other's sequence and acknowledgment numbers, they can start sending data.

A TCP connection is normally terminated using a special procedure where each side independently closes its end of the link. It normally begins with one of the application processes signaling to its TCP layer that the session is no longer needed. In the normal case, each side terminates its end of the connection by sending a special message with the FIN (finish) bit set. This message, sometimes called a FIN, serves as a connection termination request to the other device, while also possibly carrying data like a regular segment. The device receiving the FIN responds with an acknowledgment to the FIN to indicate that it was received. The connection as a whole is not considered terminated until both sides have finished the shut down procedure by sending a FIN and receiving an ACK.

6.1.3.2 Congestion Avoidance

Congestion can occur when data arrives on a big pipe (a fast LAN) and sent out on a smaller pipe (a slower WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs. Congestion avoidance is a way to deal with lost packets. The loss of a packet signals congestion somewhere in the network between the source and destination. There are two indications of packet loss: a timeout occurring and the receipt of duplicate ACKs. Congestion avoidance and slow start are two algorithms to deal with congestion. But when congestion occurs TCP must slow down its transmission rate of packets into the network, and then invoke slow start to get things going again.

6.1.3.3 Fast Retransmit

TCP may generate an immediate acknowledgment (a duplicate ACK) when an out- of-order segment is received. This duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order and to tell it what sequence number is expected.

Since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of of the missing segment, without waiting for a retransmission timer to expire.

6.1.3.4 Fast Recovery

After fast retransmit sends missing segment, congestion avoidance, but not slow start is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows.

The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start.

6.2 User Datagram Protocol

UDP (user datagram protocol), sit on top of internet layer protocols (such as IP), and are typically responsible for establishing and dissolving communication between two specific devices. This type of communication is referred to as point-to-point communication.

6.2.1 UDP Services

User datagram protocol provides unreliable, connectionless service of transport layer. Protocols at this layer allow for multiple higher-layer applications running on the device to connect point-to-point to other devices. UDP provides services such as multiplexing connections, connectionless services. A number of UDP's attributes make it especially suited for certain applications.

- It is transaction-oriented, suitable for simple query-response protocols such as the Domain Name System or the Network Time Protocol.
- It provides datagrams, suitable for modeling other protocols such as in IP tunneling or Remote Procedure Call and the Network File System.
- It is simple, suitable for bootstrapping or other purposes without a full protocol stack, such as the DHCP and Trivial File Transfer Protocol.
- It is stateless, suitable for very large numbers of clients, such as in streaming media applications for example IPTV.
- The lack of retransmission delays makes it suitable for real-time applications such as Voice over IP, online games, and many protocols built on top of the Real Time Streaming Protocol.
- Works well in unidirectional communication, suitable for broadcast information such as in many kinds of service discovery and shared information such as broadcast time or Routing Information Protocol.

6.2.2 UDP Operation

There are several types of sockets that a transport protocol can use, such as stream, datagram, raw, and sequenced packet, to name a few. UDP uses datagram sockets, a message oriented socket handling data one message at a time.

There is a socket on each end of a point-to-point communication channel, and every application on a device want to establish communication to another device by establishing a socket. Sockets are bound to specific ports on that device, where the port number determines the application incoming data is intended for. Ports are 16-bit unsigned integers, meaning each device has 65536 (0-65535) ports. Some ports are assigned to particular applications (i.e., FTP = ports 20-21, HTTP = port 80, etc.). The client and server then send and receive data via their sockets.

In general, on the server side a server application is running, listening to the socket, and waiting for a client to request a connection. UDP essentially includes the destination IP address and port number in the transmitted packet, there is no handshaking to verify the data is received in the correct order, or even at all. The server determines if the received data is for one of its own applications by extracting the IP address and port number from the received packet. After the connection is successfully established, the client application establishes a socket for communication, and the server then establishes a new socket to listen the incoming requests from other clients.

6.2.3 UDP Packet Format

As shown in figure 6.3 the UDP header consists of 4 fields, each of which is 2 bytes (16 bits).

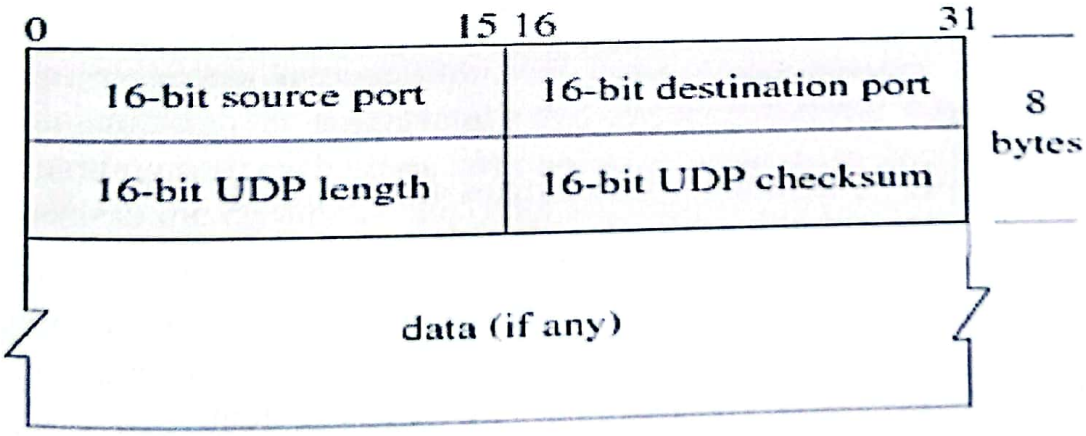

Figure 6.3 UDP Packet Format

- Source port number: This 16 bit field identifies the sender's port and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.

- Destination port number: This 16 bit field identifies the receiver's port. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.
- Length: A 16 bit field that specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes since that's the length of the header. The field size sets a limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes (65,535 – 8 byte UDP header – 20 byte IP header).
- Checksum: The 16 bit checksum field is used for error-checking of the header and data. If no checksum is generated by the transmitter, the field uses the value all-zeros.

Example 6.2

Determine

(i) Source port

(ii) Destination port

(iii) Header Length

(iv) Checksum

from the UDP segment as below

1e 2d 1e 2d 00 48 74 57

Solution

(i) Source port is given by first two bytes in segment

Here we can observe the first two bytes are 1e and 2d

Source port= $(1e\ 2d)_{16}$

=7725

(ii) Destination port is given by 3rd and 4th byte in segment.

Destination port=$(1e\ 2d)_{16}$ =7725

(iii) Header Length is given by 5th and 6th bytes

Here Header length= $(00\ 48)_{16}$ = 0000 0000 0100 1000

= 72

(iv) Checksum is given by 7th and 8th byets

Checksum= $(74\ 57)_{16}$= 0111 0100 0101 0111= 0x7457

6.3 Vulnerabilities and Attacks in TCP, UDP

In this section, we identify vulnerabilities and attacks in TCP and UDP protocol.

Vulnerabilities in TCP

- Insufficient transport layer protection allows communication to be exposed to untrusted third parties, providing an attack vector to compromise a web application and/or steal sensitive information, unless the website is configured to use SSL/TLS and configured to use SSL/TLS properly, the website may be vulnerable to traffic interception and modification.
- When the transport layer is not encrypted, all communication between the website and client is sent in clear-text which leaves it open to interception, injection and redirection also known as a man-in-the-middle/MITM attack. An attacker may passively intercept the communication, giving them access to any sensitive data that is being transmitted such as usernames and passwords.
- An attacker may also actively inject content from the communication, allowing the attacker to forge and omit information, inject malicious scripting, or cause the client to access remote untrusted content. An attacker may also redirect the communication in such a way that the website and client are no longer communicating with each other, but instead are unknowingly communicating with the attacker in the context of the other trusted party.
- TCP Timers: Connection Establishment Timer is started when the SYN is sent during the initial connection setup. Typical value of this timer is 75 seconds. If a time-out occurs, the connection is aborted.
- Connection Establishment and Termination: TCP exploits are typically based on IP spoofing and sequence number prediction. In establishing a TCP connection, both the server and the client generate an initial sequence number from which they will start counting the bytes transmitted. This sequence number is generated at random and should be hard to predict. However, some implementations of the TCP/IP protocol make it rather easy to predict this sequence number. The attacker either sniffs the current SEQ/ACK of the connection, or can algorithmically predict them.

- Simultaneous Connections: Occasionally, it is possible that hosts XX and YY both wish to establish a connection and both of them simultaneously initiate the handshake. This is called simultaneous connection establishment shown in figure 6.4.

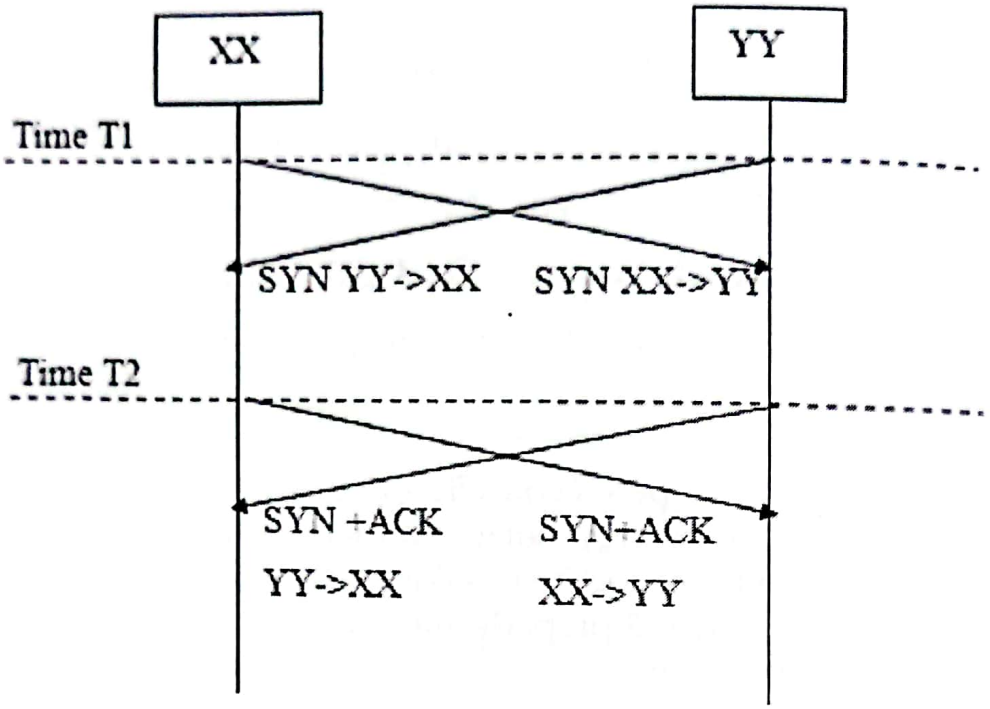

Figure 6.4 Simultaneous Connection

Both hosts XX and YY send out SYN's to each other. When the SYN's are received, each receiver sends out a SYN+ACK. Both hosts XX and YY must detect that the SYN and SYN+ACK actually refer to the same connection. If both hosts XX and YY detect that the SYN+ACK belongs to the SYN that was recently sent, they switch off the connection establishment timer and move directly to the SYN_RECVD state. This flaw could be used to steal a port on a host, using protocols such as FTP where the server initiates a connection to the client.

- The TCP specification does not specify clearly certain transitions for example suppose an attacker sends a TCP segment with both the SYN and the FIN bit set. Depending on the TCP implementation, victim host processes the SYN flag first, generates a reply packet with the corresponding ACK flag set and perform a state-transition to the state SYN_RCVD. Victim host then processes the FIN flag, performs a transition to the state CLOSE_WAIT, and sends the ACK packet back to attacker. The attacking host does not send any other packet to victim. TCP state machine in the victim for this connection is in CLOSE_WAIT state. The victim connection gets stuck in this state until the expiry of the keep-alive timer.
- In TCP/IP, there are a number of methods through which data can be surreptitiously passed between hosts. A number of fields in the TCP/IP header are unused or optional, and these can be used for covert data transmission. These packets either contain no actual data, or contain data designed to look innocent.

- Attackers can exploit many attacks by predicting sequence numbers which results in SYN flood attack known as Connection hijacking attack.

Vulnerabilities in UDP

- UDP lacks access and bandwidth control which causes denial of service and session hijacking.
- UDP traffic is unencrypted so anyone can capture the packets and decode it.
- In spite of a UDP header checksum and unencrypted data, most of the UDP header fields are easily manipulated or reproduced.
- The ability of attacker to frame malformed UDP packet, increases the chances of UDP session or application data hampering.
- Difficulty in finding manipulated packets which cause denial of service is in inspection of UDP traffics at packet inspecting devices as UDP is connectionless.

Attacks in TCP

The above mentioned vulnerabilities are exposed by many attackers to carry out wide variety of attacks. Some of the attacks are discussed in detail below.

- **TCP SYN Attacks:** The Transfer Control Protocol (TCP) includes a full handshake between sender and receiver, before data packets are sent. The TCP SYN attack exploits the three-way handshake between the sending system and the receiving system by sending large volumes of TCP SYN packets to the victim system with spoofed source IP addresses. The victim system responds at the source address of SYN request with the ACK+SYN which is actually a non-requesting host. When a large volume of SYN requests are being processed by a server and none of the ACK+SYN responses are returned, the server begins to run out of processor and memory resources. Eventually, if the volume of TCP SYN attack requests is large and they continue over time, the victim system will run out of resources and be unable to respond to any legitimate users. Nevertheless, if the SYN flood is strong enough and coming from multiple sources (especially if the source IP addresses in the incoming SYN packets are being spoofed), there may be no protection against such an attack.
- **PUSH + ACK Attacks:** In the TCP protocols, packets that are sent to a destination are buffered within the TCP stack and when the stack is full, the packets are sent to the receiving system. However, the sender can request the receiving system to unload the contents of the buffer before the buffer becomes full by sending a packet with the PUSH bit set to one. PUSH is a one-bit flag within the TCP header. The PUSH + ACK attack is similar to a TCP SYN attack in that its goal is to consume the resources of the victim system. The attacking agents send TCP packets with the PUSH and ACK bits set to one. These packets instruct the victim system to unload all data in the TCP buffer and send an acknowledgment when complete. If this

process is repeated with multiple agents, the receiving system cannot process the large volume of incoming packets and it will crash.

- **TCP Connection Hijacking:** Connection Hijacking is the attack where attacker takes control of the connection between two legitimate users. The attack is explained below:

  Any packet is legitimate or correct if it have correct SEQ+ACK numbers. So if someone change the sequence number of packet, the receiver would ingnore that packet and attacker could then impersonate to be legitimate user, but using correct SEQ+ACK numbers from the perspective of receiver. Attacker then send the packets to the receiver by the name of original sender. This is known as session hijacking attack.

Attacks on UDP

The various attacks on UDP are discussed below:

- **ICMP Tunneling:** This occurs by triggering a response from the ICMP protocol when it responds to a seemingly legitimate request. Ping for instance, that uses the ICMP protocol. sPing is a good example of this type of attack, which overloads the server with more bytes than it can handle, larger connections. ICMP can contain data about timing and routes. A packet can be used to hold information that is different from the intended information. This allows an ICMP packet to be used as a communications channel between two systems. The channel can be used to send a Trojan horse or other malicious packet.
- **Covert UDP:** There is no encryption techniques applied to data part of UDP. The UDP header fields are easily manipulated. As the data part in UDP is not protected, it can facilitate the covert data tunneling through the network.
- **Denial-of-Service:** A denial-of-service (DoS) attack refers to an attack on a network geared toward making system resources unavailable to intended users. A DoS attack involves prevention of a service or website from functioning properly. A UDP flood involves such an attack.
- **SMURF Attack:** This attack uses IP spoofing and broadcasting to send a ping to a group of hosts on a network. When a host is pinged it sends back ICMP message traffic information indicating status to the originator. If a broadcast is sent to network, all hosts will answer back to the ping. The result is an overload of network and the target system.
- **UDP Flood Attack:** UDP flooding is an event that occurs when an attacker sends out IP packets containing UDP datagrams, to random ports on a remote host. The host check for applications listed at the port, determines no applications are listed and replies with a "Destination Unreachable" packet. Ultimately the host sends out so many packets that the system becomes flooded, and thus unattainable to other clients.

6.4 Defense Mechanisms

The following counter measure mechanisms could help in defending the system, and one can build a more effective overall defense by combining several of them. Using a layered approach that combines several types of defenses, at several different locations, can be more flexible and harder for the hacker to completely bypass. These mechanisms include filtering, monitoring, port blocking and other adequate resources. Here we discuss various defense mechanisms for TCP and UDP.

Defense Mechanism for TCP Attacks

- **Packet Filtering Firewall:** For SYN Flood attack to work, the ability to send packets with spoofed source IP addresses is required. Removing an attacker's ability to send spoofed IP packets is an effective solution that requires no modifications to TCP. The packet Filtering firewall may stop or drop packets by inspecting them.
- **Network based IDS:** NIDS is designed to receive all packets on a network by using methods like tap, port monitoring etc. Most of the NIDS systems are pattern based which requires signatures to alert any intrusion. The logging and reporting of attacks by IDS systems can be used to do much more than detect specific, isolated, and unrelated attacks. By combining the data from all internal IDS systems, system administrators can identify attack trends and patterns and fine tune the IDS.
- **Better Sequence Numbers:** The choice of initial sequence numbers for a connection should not be random. Rather, it must be chosen so as to minimize the probability of old stale packets being accepted by new manifestations of the same connection.
- **Ingress – Inbound filtering Ingress filtering may work in this case is explained below**

  1. Block packets destined for services that are not being offered to the Internet.
  2. Block addresses that have a source IP address of:
     a. Illegal addresses – e.g. 0.0.0.0/8 (CIDR notation)
     b. Broadcast address – 255.255.255.255/32
     c. RFC1918 reserved addresses – Private networks use these.

There should not be any traffic attempting to access your network with these as source addresses. The IANA reserved blocks are:

A. 10.0.0.0 - 10.255.255.255

B. 172.16.0.0 - 172.31.255.255

C. 192.168.0.0 - 192.168.255.255

D. Multicast, if multicast is not being used.

E. Loopback – 127.0.0.0/8

F. ICMP broadcast per RFC 2644 – This will keep a site from being used as a **smurf amplifier.**

G. UDP echo

Defense Mechanisms for UDP Attacks

DoS attacks with UDP flooding are one of the many techniques hacker's uses to make the attack. Such attacks are made to make the network and related services non-operational and thereby restricting the legitimate user from using the system. UDP flooding can be prevented or managed via firewalls. UDP flooding is done using specific or random ports of the victims system. Therefore, it is recommended that: All unused UDP services on hosts be disabled. Firewall is configured with all UDP ports less than 900 be blocked except some specific services, such as DNS (port 53). Firewalls are system or network software designed to prevent unauthorized access while permitting authorized data transmissions. Firewalls can be positioned at key points in a system or network to filter out unwanted traffic.

Preventive methods should not have the effect of spoiling other forms of network traffic. Reactive methods should be activated only when a DoS attack is under way. False positives may cause indirect damage in many cases, but there are other undesirable methods of high false positive rates. For instance, when a reactive system detects and responds to a DoS attack, it can send a signal to the system administrator of the targeted system that it is taking action.

In case, UDP services are accessible from external network, it is recommended that proper packet and flow monitoring is in place to learn which systems are using the UDP services and to monitor for any misuse by the external systems. Monitoring could be done using Snort, tcpdump, and netlog, etc. While monitoring with the Snort, suspicious packet may be detected using packet sniffer (we used wireshark) and filtered accordingly. Threshold for network flow may be fixed and alert is generated accordingly.

The counter measure is to deny ICMP traffic on your network. The only way to prevent this attack is to prohibit ICMP traffic on the router.

6.5 Conclusion

Transport layer contains all the functions and mechanisms needed to make up for the best-effort connectionless delivery provided by the IP layer. Packets could arrive at a host with errors, out of their correct sequence, duplicated, or with gaps in sequence due to lost (or discarded) packets. TCP must guarantee that the data stream is delivered to the destination application error-free, with all data in sequence and complete.When applications use UDP instead of TCP, there is no need to establish, maintain, or tear down a connection between a source and destination before sending data. Connection management adds overhead and some initial delay to the network. In this chapter we have seen the various flaws in the design of these protocols and attacks exploiting them and the defense techniques against the TCP and UDP