Chapter 6
Transport Layer and Protocols

Transport Layer:

The transport layer provides a logical connection between a source host and a destination host. Transport protocols segment and reassemble data sent by upper-layer applications into the same data stream, or logical connection, between end points.

- Creates packet from bytes stream received from the application layer.
- Uses port number to create process to process communication.
- Uses a sliding window protocol to achieve flow control.
- Uses acknowledgement packet, timeout and retransmission to achieve error control.

The primary duty of the transport layer is to provide end-to-end control and reliability as data travels through this cloud. This is accomplished through the use of sliding windows, sequence numbers, and acknowledgments. The transport layer also defines end-to-end connectivity between host applications. Transport layer protocols include TCP and UDP.

TCP

TCP is a connection-oriented transport layer protocol that provides reliable full-duplex data transmission. TCP is part of the TCP/IP protocol stack. In a connection-oriented environment, a connection is established between both ends before the transfer of information can begin. TCP breaks messages into segments, reassembles them at the destination, and resends anything that is not received. TCP supplies a virtual circuit between end-user applications.

It exhibits the following key features:

- Transmission Control Protocol (TCP) corresponds to the Transport Layer of OSI Model.
- TCP is a reliable and connection oriented protocol.
- TCP offers:
    - Stream Data Transfer.
    - Reliability.
    - Efficient Flow Control
    - Full-duplex operation.
    - Multiplexing.
- TCP offers connection oriented end-to-end packet delivery.
- TCP ensures reliability by sequencing bytes with a forwarding acknowledgement number that indicates to the destination the next byte the source expect to receive.
- It retransmits the bytes not acknowledged with in specified time period.

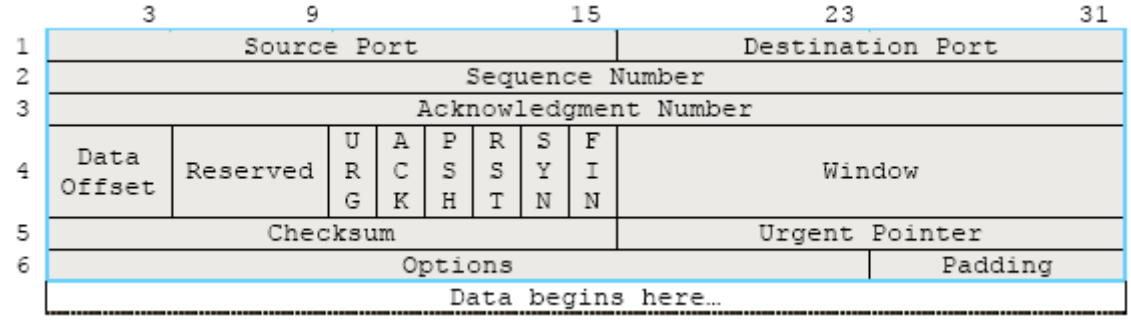
TCP Services

TCP offers following services to the processes at the application layer:

- Stream Delivery Service
- Sending and Receiving Buffers
- Bytes and Segments
- Full Duplex Service
- Connection Oriented Service
- Reliable Service

TCP Header Format:

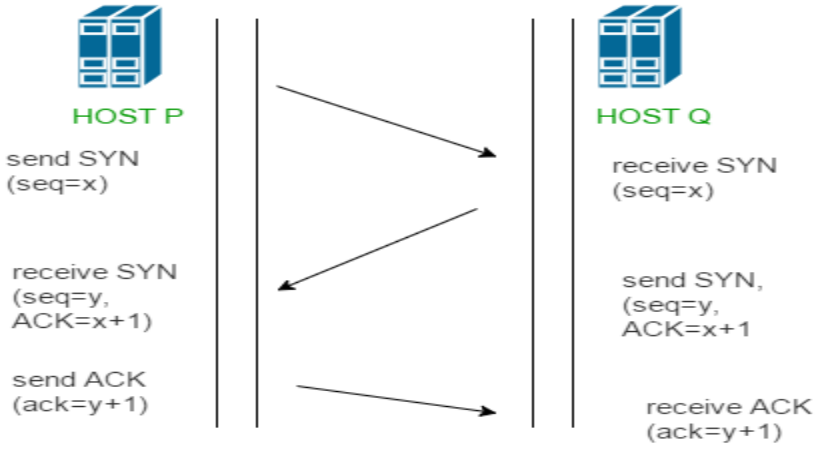
TCP uses only a single type of protocol data unit, called a TCP segment. The header is shown in Figure. Because one header must serve to perform all protocol mechanisms, it is rather large, with a minimum length of 20 octets.



The following are the definitions of the fields in the TCP segment:

- Source port – Number of the port that sends data
- Destination port – Number of the port that receives data
- Sequence number – Number used to ensure the data arrives in the correct order
- Acknowledgment number – Next expected TCP octet
- HLEN – Number of 32-bit words in the header
- Reserved – Set to zero
- Code bits – Control functions, such as setup and termination of a session
- Window – Number of octets that the sender will accept
- Checksum – Calculated checksum of the header and data fields
- Urgent pointer – Indicates the end of the urgent data
- Option – One option currently defined, maximum TCP segment size
- Data – Upper-layer protocol data
- Code Bits or Flags (6 bits).
    - URG: Urgent pointer field significant.
    - ACK: Acknowledgment field significant.
    - PSH: Push function.
    - RST: Reset the connection.
    - SYN: Synchronize the sequence numbers.
    - FIN: No more data from sender.

TCP Handshake



- **Step 1 (SYN) :** In the first step, client wants to establish a connection with server, so it sends a segment with SYN(Synchronize Sequence Number) which informs server that client is likely to start communication and with what sequence number it starts segments with
- **Step 2 (SYN + ACK):** Server responds to the client request with SYN-ACK signal bits set. Acknowledgement(ACK) signifies the response of segment it received and SYN signifies with what sequence number it is likely to start the segments with
- **Step 3 (ACK) :** In the final part client acknowledges the response of server and they both establish a reliable connection with which they will start eh actual data transfer

The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established.

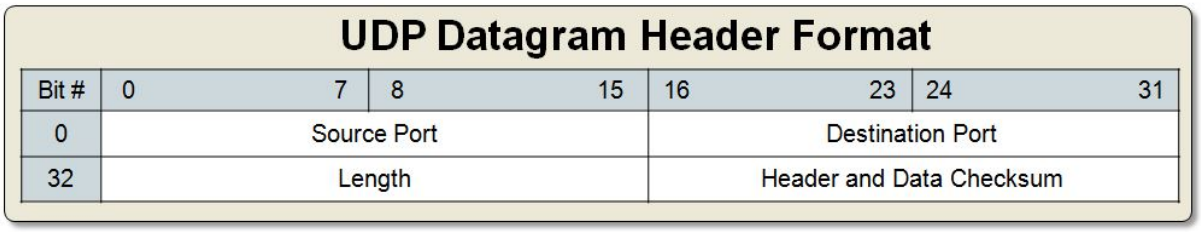
UDP (User Datagram Protocol):

UDP is the connectionless transport protocol in the TCP/IP protocol stack. UDP is a simple protocol that exchanges datagrams without guaranteed delivery. It relies on higher-layer protocols to handle errors and retransmit data.

Features of UDP:

- Provides connectionless, unreliable service.
- So UDP faster than TCP.
- Adds only checksum and process-to-process addressing to IP.
- Used for DNS and NFS.
- Used when socket is opened in datagram mode.
- It sends bulk quantity of packets.
- No acknowledgment.
- Good for video streaming it is an unreliable protocol.
- It does not care about the delivery of the packets or the sequence of delivery.
- No flow control /congestion control, sender can overrun receiver's buffer.
- Real time application like video conferencing needs (Because it is faster).
- An UDP datagram is used in Network File System (NFS), DNS, SNMP, TFTP etc.
- It has no handshaking or flow control.
- It not even has windowing capability.
- It is a fire and forget type protocol.
- An application can use a UDP port number and another application can use the same port number for a TCP session from the same IP address.
- UDP and IP are on different levels of the OSI stack and correspond to other protocols like TCP and ICMP.
- No connection establishment tear down; data is just sent right away.
- For data transfer with UDP a lock-step protocol is required (to be implemented by the application).
- No error control; corrupted data is not retransmitted (even though UDP header has a checksum to detect errors and report these to the application).

UDP Header Format



The following are the definitions of the fields in the UDP segment:

Source port number

- This is the port number used by the process running on the source host.
- It is 16 bits long, which means that the port number can range from 0 to 65,535.
- If the source host is the client, the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host.
- If the source host is the server, the port number, in most cases, is a well-known port number.

Destination port number

- This is the port number used by the process running on the destination host. It is also 16 bits long.
- If the destination host is the server, the port number, in most cases, is a well-known port number.
- If the destination host is the client, the port number, in most cases, is an ephemeral port number.

Length

- This is a 16-bit field that defines the total length of the user datagram, header plus data.
- The 16 bits can define a total length of 0 to 65,535 bytes. The length field in a UDP user datagram is actually not necessary.

Checksum

- This field is used to detect errors over the entire user datagram (header plus data).

TCP VS UDP

Characteristics Description	UDP	TCP
Acronym for	User Datagram Protocol	Transmission Control Protocol
General Description	Simple High speed low functionality "wrapper" that interface applications to the network layer and does little else	Full-featured protocol that allows applications to send data reliably without worrying about network layer issues.
Protocol connection Setup	Connection less; data is sent without setup	Connection-oriented; Connection must be Established prior to transmission.
Data interface to application	Message base-based is sent in discrete packages by the application.	Stream-based; data is sent by the application with no particular structure
Reliability and Acknowledgements	Unreliable best-effort delivery without acknowledgements	Reliable delivery of message all data is acknowledged.
Retransmissions	Not performed. Application must detect lost data and retransmit if needed.	Delivery of all data is managed, and lost data is retransmitted automatically.
Overhead	Very Low	Low, but higher than UDP
Transmission speed	Very High	High but not as high as UDP
Data. Quantity Suitability	Small to moderate amounts of data	Small to very large amounts of data.

Socket Programming:

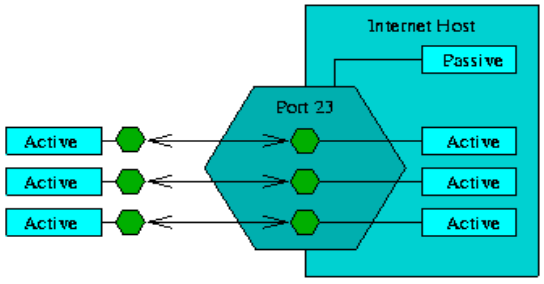
Sockets are the combination of IP address plus corresponding TCP/UDP port numbers. For applications to work with TCP/IP, Application Program Interface (API) is required. API serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers.

Sockets come in two primary flavors

An **active socket** is connected to a remote active socket via an open data connection. Closing the connection destroys the active sockets at each endpoint.

A **passive socket** is not connected, but rather awaits an incoming connection, which will spawn a new active socket.

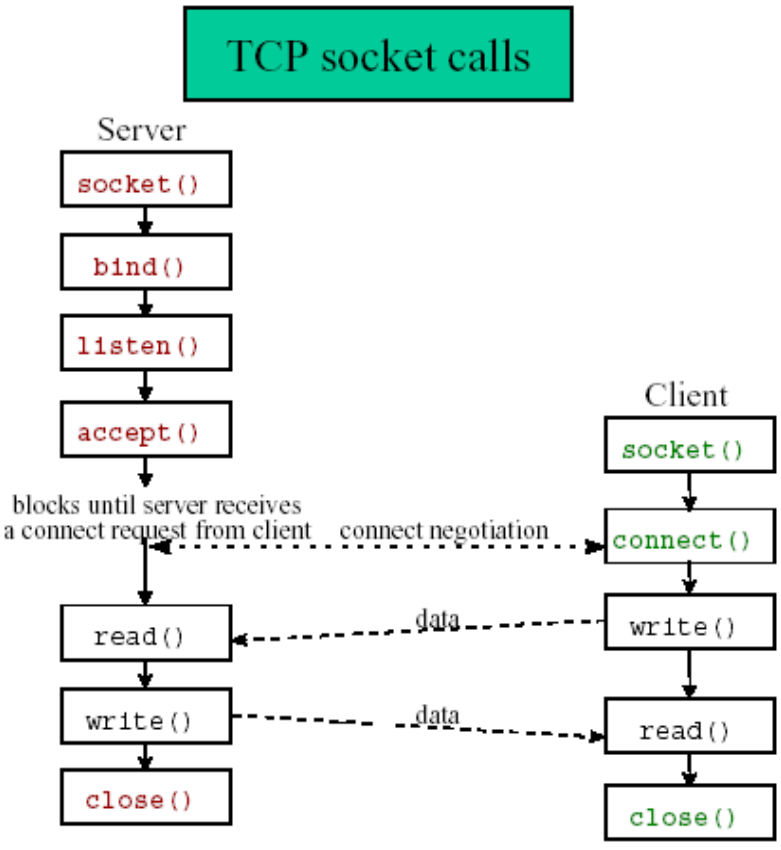
A socket is not a port, though there is a close relationship between them. A socket is associated with a port, though this is a many-to-one relationship. Each port can have a single passive socket, awaiting incoming connections, and multiple active sockets, each corresponding to an open connection on the port.



Sockets are the combination of IP address plus corresponding TCP/UDP port numbers. It is like PBX phone systems, where the IP address is the phone number, and the port is the extension. Every paired of connected socket has a source IP/port and a destination IP/port. Users of Internet applications are normally aware of all except the local port number, this is allocated when connection is established and is almost entirely arbitrary unlike the well-known port numbers associated with popular applications.

There are three types of sockets:

1. **Stream**
   Stream sockets provide reliable, connection-based communications. In connection-based communications, the two processes must establish a logical connection with each other. A stream of bytes is then sent without errors or duplication and is received in the order in which it was sent. Stream sockets correspond to the TCP protocol in TCP/IP.
2. **Datagram**
   Datagram sockets communicate via discrete messages, called datagrams, which are sent as packets. Datagram sockets are connectionless; that is, the communicating processes do not have a logical connection with each other. The delivery of their data is unreliable. The datagrams can be lost or duplicated, or they may not arrive in the order in which they were sent. Datagram sockets correspond to the UDP protocol in TCP/IP.
3. Raw
   Raw sockets provide direct access to the lower-layer protocols, for example, IP and the Internet Control Message Protocol (ICMP).



UDP socket calls