Transport Layer (Layer-4) refers to *transportation of data or data stream* are categorized into this layer. As all other layers, this layer *communicates with 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.*

Goal of the transport layer is to efficient, reliable and cost effective service to application layer. To provide the service, transport layer uses service provided by network layer. Hardware and software within the transport layer that provides service is called **transport layer entity**. Transport layer provides two types *service to application layer. (i ) Connection oriented service and (ii) Connectionless service*. The main transport *protocols are TCP and UDP.*

- **Transmission Control Protocol ( TCP ) :** It provides reliable communication between two hosts. *E.g. World Wide Web(HTTP), E-mail (SMTP TCP), File Transfer Protocol (FTP), Secure Shell (SSH).* ***Example*** *:***Email:** ***Reason:*** suppose if some packet(words/statement) is missing we cannot understand the content. It should be reliable.
- **User Datagram Protocol ( UDP ) :** It provides unreliable communication between two hosts. *E.g. Domain Name System (DNS), Streaming media applications such as movies, Online multiplayer games, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP).* **Example : Video streaming: Reason:** Suppose if some packet(frame/sequence) is missing we can understand the content. Because video is collection of frames. For 1 second video, there should be 25 frames(image). Even though we can understand some frames are missing due to our imagination skills.

## 4.1 Introduction to UDP (User Datagram Protocol)

This protocol is often referred to as Unreliable Datagram Protocol (UDP) because, unlike TCP streams, *there is no guarantee that data sent will reach its destination. When using a TCP socket, a connection is established between the end point before communication take s place and this connection is maintained until one end decides to actively close it. With a UDP socket a connection is NOT made, instead the sender just issues a message to its destination and hopes it gets there!* The message uses a datagram of fixed length, often termed a record. Since there is no connection between client and server the client can send a datagram to one server and then immediately send a datagram to another server using the same socket! UDP is a connectionless protocol.

### 4.1.1 Operation of UDP

***What UDP Does***

UDP's only *real task is to take data from higher-layer protocols and place it in UDP messages*, which are then passed down to the Internet Protocol for transmission. The basic steps for transmission using UDP are:

1. **Higher-Layer Data Transfer:** An application sends a message to the UDP software.
2. **UDP Message Encapsulation:** The higher-layer message is encapsulated into the *Data* field of a UDP message. The headers of the UDP message are filled in, including the *Source Port* of the application that sent the data to UDP, and the *Destination Port* of the intended recipient. The checksum value may also be calculated.
3. **Transfer Message To IP:** The UDP message is passed to IP for transmission.

And that's about it. Of course, on reception at the destination device this short procedure is reversed.

**How UDP Works**

UDP stands for User Datagram Protocol — *a datagram is the same thing as a packet of information*. The UDP protocol works similarly to TCP, but it throws all the *error-checking stuff out*. All the back-and-forth communication and deliverability guarantees slow things down. When using UDP, packets are just sent to the recipient. The sender won't wait to make sure the recipient received the packet — it will just continue sending the next packets. If you're the recipient and you miss some UDP packets, too bad — you can't ask for those packets again. There's no guarantee you're getting all the packets and there's no way to ask for a packet again if you miss it, but losing all this overhead means the computers can communicate more quickly.

UDP is used when speed is desirable and error correction isn't necessary. For example, UDP is frequently used for live broadcasts and online games.

The UDP messages will have the following parts:

- Source port - this is a 16 bit port number of the process that initially originated the UDP message on the source device.
- Destination port - this is the second part of the message. It is a 16 bit-port number of the process it is ultimate recipient of the message. This will generally be the port number of the client.
- Length -the length means the entire UDP datagram that includes data fields and the headers.
- Checksum - as we have already explained it before it is optional and generally 16 bit in size.
- Data - it is the variable part of the UDP. It encapsulates the higher layer messages that need to be sent.

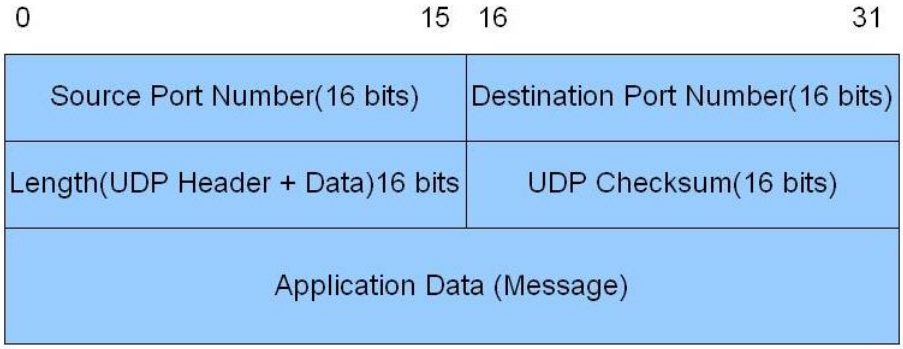



*Fig. UDP Header Format*

### 4.1.2 Characteristics of UDP

- **UDP is unreliable** – UDP does not guarantee delivery of packets. There is no error detection, flow control or re-transmission of lost packets. It just sends them and doesn't care whether they arrive or not.
- 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 is fast** – Because UDP doesn't have the additional overhead as TCP it is a faster protocol ideal for streaming

### 4.1.3 Application of UDP

- Playing a game video online
- Streaming movies online
- UDP is used with RTOS (Real time operating systems), which is a high availability control software in civil aircraft.
- Video conference

## 4.2 Introduction to TCP (Transmission Control Protocol)

The TCP is referred as the reliable protocol, which is responsible for *breaking up the messages into the TCP segments as well as reassembling it in a receiving side*. The major purpose of the TCP is to give the reliable and secure logical connection, service or circuit between the pairs of the processes. To offer this type of service on top of the less reliable internet communication system needs facilities in areas such as security, precedence, multiplexing, reliability, connections and basic data transfer. The main purpose of the TCP is flow control and error recovery. As it is connection based protocol, which means that before allowing any data it accomplishes connections and also terminates it upon completion.

The two main components of TCP are the two end systems which is a web browser and a web server, for instance. TCP offers the delivery of a stream of bytes from a program from one computer to another computer. TCP is also responsible for controlling size, flow control, the rate of data exchange, and network traffic congestion.

### 4.2.1 Operation of TCP.

- TCP is the connection oriented protocol, that means the *devices must open the connection before transferring data and must lose a connection gracefully after transferring the data.* It also assures the reliable data delivery to the destinations. This protocol offers the extensive error checking mechanisms, including the acknowledge of data and flow control as mentioned above. The TCP is relatively slow because of the extensive error checking mechanisms only. Demultiplexing as well as multiplexing is greatly possible in the *TCP by means of the TCP port numbers and also retransmission of the lost packets is merely possible in the TCP.*
- During the connection, accomplishment both server and client agree upon the sequence and also acknowledge numbers. The implicit client notifies the server of its source ports. *The sequence is the characteristic of the TCP data segment.* This sequence begins with the

random number and each time the new packet is sent, then the sequence is incremented by a number of bytes sent in the previous segment of the TCP. Acknowledge segment is moreover the same, but from a receiver side. This does not comprise data and are equal to the sender's sequence numbers increased by the number of the received bytes. The ACK segment acknowledges that the host has got the sent data.

How TCP Works

- When you load a web page, your computer sends TCP packets to the web server's address, *asking it to send the web page to you*. The web server responds by sending a stream of TCP packets, which your web browser stitches together to form the web page and display it to you. When you click a link, sign in, post a comment, or do anything else, your web browser sends TCP packets to the server and the server sends TCP packets back. TCP isn't just one way communication — the remote system sends packets back to acknowledge it's received your packets.
- TCP guarantees the recipient will receive the *packets in order by numbering* them. The recipient sends messages back to the sender saying it received the messages. If the sender doesn't get a correct response, it will *resend* the packets to ensure the recipient received them. Packets are also checked for errors. TCP is all about this reliability — packets sent with TCP are tracked so no data is lost or corrupted in transit. (*This is why file downloads don't become corrupted even if there are network hiccups. Of course, if the recipient is completely offline, your computer will give up and you'll see an error message saying it can't communicate with the remote host.)*

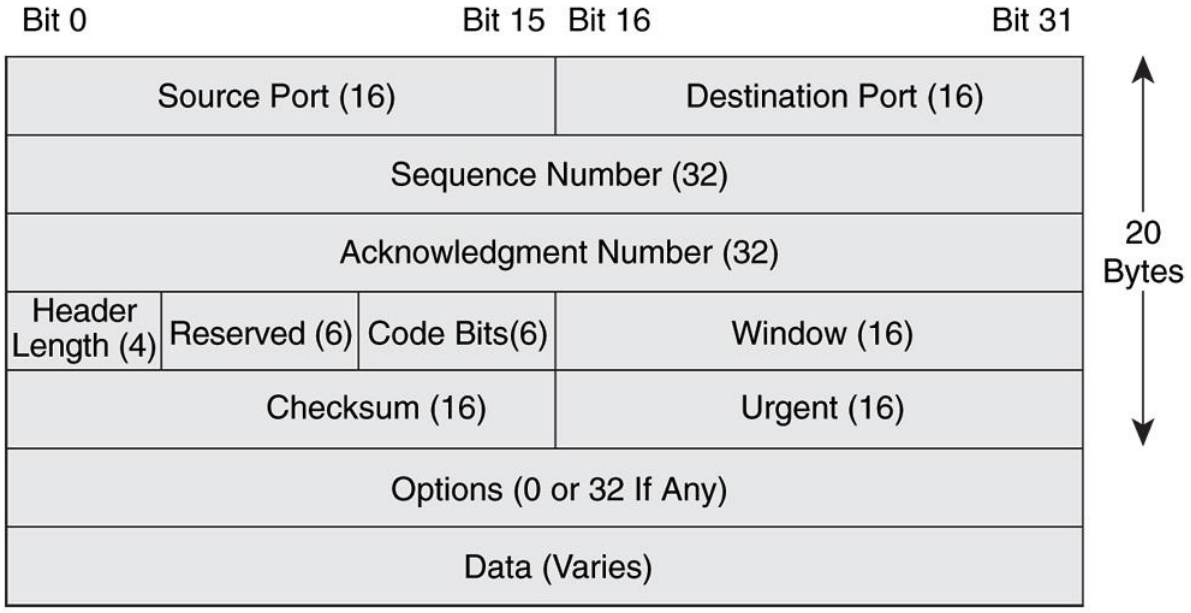



*Fig. TCP Header Format*

- **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 or Header Length (4-bits)** - This field implies both, the size of TCP header (32-bit words) 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.
- **Code Bits or Flags (1-bit each)**
  - **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 (congestion experience) 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 (as soon as it comes) 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.

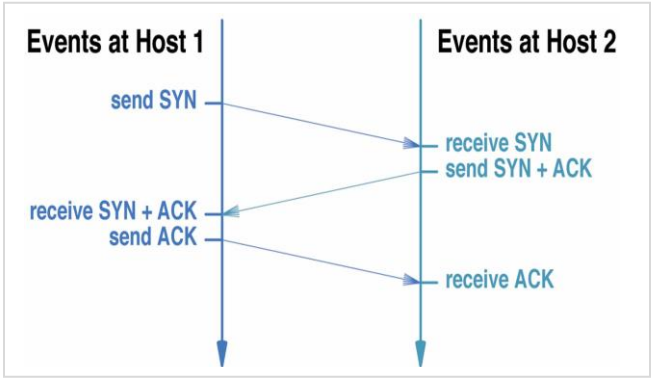
- **SYN** - This flag is used to set up a connection between hosts.
- **FIN** - This flag is used to release a connection and no more data is exchanged thereafter. Because packets with SYN and FIN flags have sequence numbers, they are processed in correct order.

- **Windows Size** - Bandwidth Management; 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** – Error control; This field contains the checksum of Header, Data and Pseudo Headers.
- **Urgent Data 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.
- **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

### 4.2.2 Characteristics of TCP

- **TCP is reliable** – This means it guarantees the delivery packets uncorrupted. This is all done by controlling the session with *flow control, error detection, congestion control and re-transmission* of lost packets.
- **TCP is a connection orientated protocol** – This means a connection or socket must first be established before data can flow. *Data travels both ways.*
- **TCP is ordered** – TCP uses sequence numbers to ensure that packets are re-constructed in the correct order.
- **TCP is slower than UDP** – *Because TCP does all the above there is additional overhead needed and processing time which makes TCP slower than UDP*

### 4.2.3 TCP three-way handshake process... SYN, SYN + ACK, ACK




Is used for establishing TCP connection for reliable communication. There are three messages (Sync, Sync + Ack, Ack) transmitted by TCP to negotiate and start a TCP session between two computers. A three-way handshake is also used to terminate a connection

Scenarios for establishing a connection using a three-way handshake. CR : CONNECTION REQUEST.

(a) Normal operation,

(b) Old CONNECTION REQUEST appearing out of nowhere.

(c) Connection Release : Connection at transport can be released in two way.

1. Asymmetric: if one of host terminates connection, then in both the direction, data communication will be terminated.
2. Symmetric: if one of the host disconnects connection, then it cannot send the data but it can receive it.

- Host A sends a TCP SYNchronize packet to Host B
- Host B receives A's SYN
- Host B sends a SYNchronize-ACKnowledgement
- Host A receives B's SYN-ACK
- Host A sends ACKnowledge
- Host B receives ACK.

*TCP socket connection is ESTABLISHED.*

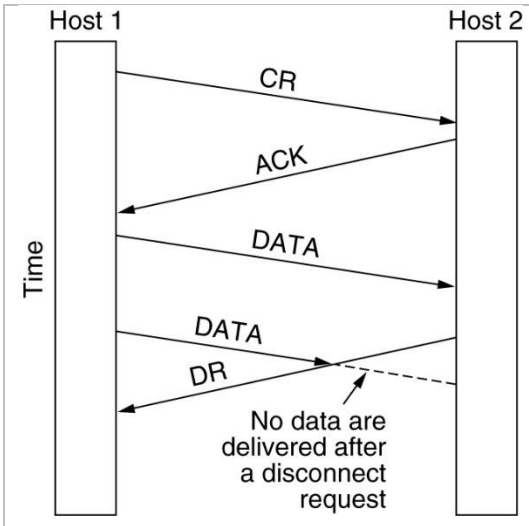
**Note : the SYN flag "consumes" one byte of sequence space so that it can be acknowledged unambiguously**

SYNchronize and ACKnowledge messages are indicated by a either the SYN bit, or the ACK bit inside the TCP header, and the SYN-ACK message has both the SYN and the ACK bits turned on (set to 1) in the TCP header.

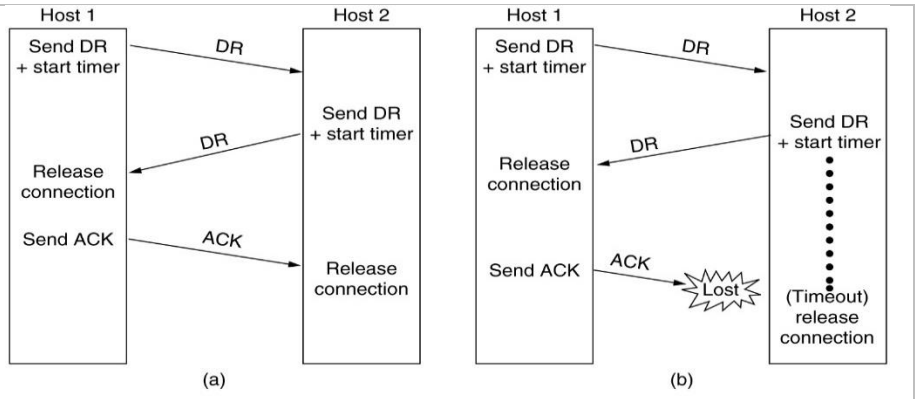
TCP knows whether the network TCP socket connection is opening, synchronizing, established by using the SYNchronize and ACKnowledge messages when establishing a network TCP socket connection.

When the communication between two computers ends, another 3-way communication is performed to tear down the TCP socket connection. This setup and teardown of a TCP socket connection is part of what qualifies TCP a reliable protocol. TCP also acknowledges that data is successfully received and guarantees the data is reassenbled in the correct order.

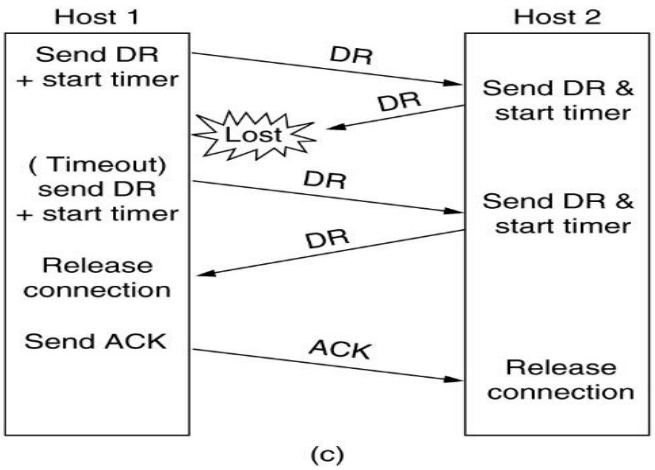
Note that UDP is connectionless. That means UDP doesn't establish connections as TCP does, so UDPdoes not perform this 3-way handshake and for this reason, it is referred to as an unreliable protocol. That doesn't mean UDP can't transfer data, it just doesn't negotiate how the conneciton will work, UDP just transmits and hopes for the best.



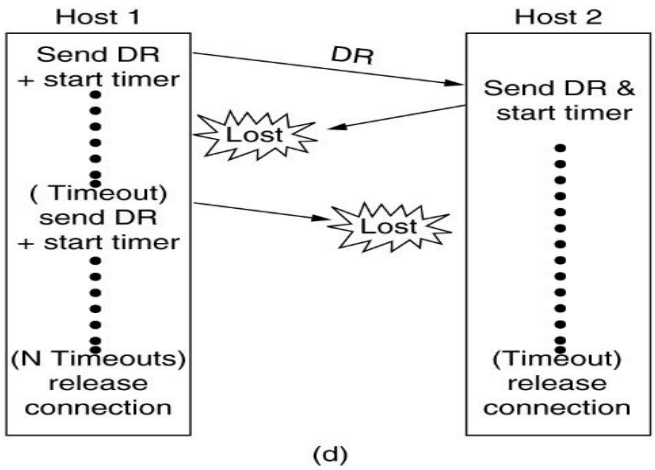

**Asymmetric Release :** Abrupt disconnection with loss of data.




(a) Normal case of a three-way handshake. (b) final ACK lost.




(c) Response DR lost




(d) Response lost and subsequent DRs lost.

TCP Connection Release uses **symmetric approach.** It is called Four Way handshaking for connection termination.

### 4.2.4 Application of TCP

- Sending and receiving an email
- Internet banking
- Remote access to your network devices
- Music streaming
- Sending sms

### 4.3 Relationship between TCP & IP

- TCP (Transmission Control Protocol) is a standard for defining the procedure *to set up and manage a network conversation via which application programs can share data among each other*. TCP works with the Internet Protocol (IP), which tells how computers transmit packets of data to each other. *We can say, the Internet is defined by two fundamental rules TCP and IP. TCP is defined by the Internet Engineering Task Force (IETF) in the Request for Comment (RFC) standards document number 793*.
- Transmission Control Protocol is one of the excessively used protocols in digital network communications and is part of the Internet protocol suite, which we generally called as the TCP/IP suite. *Principally, TCP makes sure the end-to-end delivery of data between*

*different nodes. TCP works in association with Internet Protocol. Internet Protocol defines the logical site of the remote node, whereas TCP routes and make sure that the data is reached to the right target.*

- Internet Protocol (IP) is the main set of digital message formats and rules that are used for exchanging messages *between computers across only one network or a series of unified networks, using the Internet Protocol Suite (often referred to as TCP/IP).* Messages are exchanged using datagrams, which is also known as data packets or simply packets.
- Transmission Control Protocol (TCP) is a principal protocol of the Internet Protocol Suite. *It runs at a upper level than its equivalent protocol, Internet Protocol (IP).*
- IP is a protocol that is used for data exchange across a packet switched internetwork in which all sent data is grouped as one). Like TCP, it also uses the Internet Protocol Suite. It is the important protocol in the Internet Layer of the Internet Protocol Suite. Its major job is to deliver notable protocol datagrams ( packets) from the source host to the target host only on the basis of their addresses. We can say that, IP deal about addressing techniques and structures for the packet encapsulation.
- TCP offers communication facilities at the middle level between an application program and the IP. This means when an application program desires to transmit a big piece of data across the internet by using the IP, instead of dividing the data into sizes that will suit the IP and using a series of requests from the IP, the software has capability of sending a single request to TCP, and allow this protocol to manage the information of the IP transfer. *TCP identifies problems that are seen in the IP, asks for the retransmission of the lost packets, rearranges the order of the packets in the original sequence, and helps in minimizing network congestion. Once all this has been performed and the proper replica of the data has been compiled, the packet is passed along to the application program.*
- *IP encapsulation is the process of shielding the data from the upper layer in the form of packets or datagrams. There is no actual requirement for circuit establishment before a host transmits packets to another host to which it has never before exchange data. As such, IP is considered as a protocol without a connection. It can be somehow compared with public switched telephone networks that need the establishment of a circuit in order for each phone call to go through. As a consequence of IP encapsulation, it can be used over a heterogeneous network to resolve IP addresses to data link addresses.*
- TCP/IP is a two-layer program. The *higher layer is TCP*, Transmission Control Protocol, which manages the assembling of a message or file into smaller packets that are sent over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer is Internet Protocol, manages the internet address part of each packet *so that it reaches to the correct target. Each gateway computer on the network checks this address to observe where to forward the message. Although some packets from the identical message are transported differently than others, they'll be reassembled at the target.*

### 4.3.1 Standard TCP / IP services

- **FTP and Anonymous FTP** - The File Transfer Protocol (FTP) transfers files to and from a remote network. FTP works even when the remote computer does not run a UNIX-based operating system. A user must log in to the remote computer to make an ftp connection unless it has been set up to allow anonymous FTP.
  *You can now obtain a wealth of materials from **anonymous FTP servers** connected to the Internet. These servers are set up by universities and other institutions to make certain software, research papers, and other information available to the public domain. When you log in to this type of server, you use the login name anonymous, hence the term "anonymous FTP servers."*
- **Telnet** - The Telnet protocol enables terminals and terminal-oriented processes to communicate on a network running TCP/IP. It is implemented as the program telnet (on local machines) and the daemon in.telnet (on remote machines). Telnet provides a user interface through which two hosts can communicate on a character-by-character or line-by-line basis. The application includes a set of commands that are fully documented in the telnet man page.
- **TFTP** - The trivial file transfer protocol (tftp) provides functions similar to ftp, but it does not establish ftp's interactive connection. As a result, users cannot list the contents of a directory or change directories. This means that a user must know the full name of the file to be copied.

### 4.4 Port numbers and socket address

*** PORT**

"Port" is a **number** used by a particular software to identify its data coming from internet. Each software like, skype, chrome, YouTube has its own port number and that how skype or YouTube knows which internet data is for itself.

- One of the circuit connection points on a front-end processor or local intelligent controller
- **Port** is represented by a positive (16-bit ) integer value between 0 and 65,535
- The TCP and UDP protocols use ports to map incoming data to a particular process running on a computer
- At the transport layer, an address is needed to choose among multiple processes running on the destination host called **Port Number**
  - Destination Port Number for delivery and Source Port Number for reply

Some **ports** have been reserved to support common / well known services
e.g. ftp 21/tcp, telnet 23/tcp, smtp 25/tcp, login 513/tcp

- User level process/services generally use **port number** value >= 1024

*** SOCKET**

"**IP address and Port** " together is called "Socket". It is used by another computer to send data to one particular computer's particular software. *IP address is used to identify the computer and Port is to identify the software such as IE, Chrome, Skype etc*.

- Provides an abstraction for interprocess communication
- The services provided (often by the operating system) that provide the interface between application and protocol software.
- A Socket is one endpoint of a two-way communication link between two processes running on the network.
- **Socket Address** → combination of **IP address** and a **Port number**
- A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent
- TCP connection can be uniquely identified by its two endpoints, multiple connections are possible between host and the server
- Sockets provide an interface for programming networks at the transport layer
- Process to Process delivery of data needs two identifiers, IP Address and Port Number at each endpoint
- Transport Layer Protocol needs a pair of Socket addresses
  - Client Socket Address : Uniquely defines the Client Process
  - Server Socket Address : Uniquely defines Server Process
  - *Both Socket Addresses contain IP Header and Transport Layer Protocol Header*
- IP Header *contains* IP Addresses. TCP & UDP Header *contains* the Port Numbers

**Functions**

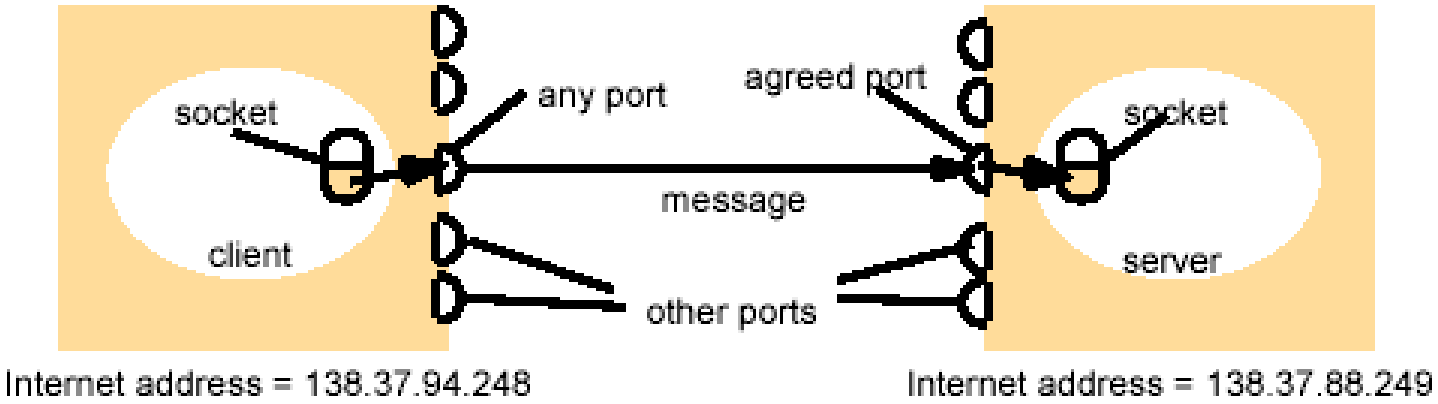
- Define an "end- point" for communication
- Initiate and accept a connection
- Send and receive data
- Terminate a connection gracefully
- Examples : Web Browsers, FTP, HTTP, Email ( SMTP, POP3)

**Types of Sockets**

I. Stream socket :( *a. k.* a. connection- oriented socket) :It provides reliable, connected networking service Error free; no out- of- order packets (uses TCP) applications: telnet/ ssh, http, …

II. Datagram socket :( *a. k.* a. connectionless socket) : It provides unreliable, best- effort networking service Packets may be lost; may arrive out of order (uses UDP) applications: streaming audio/ video (real player),

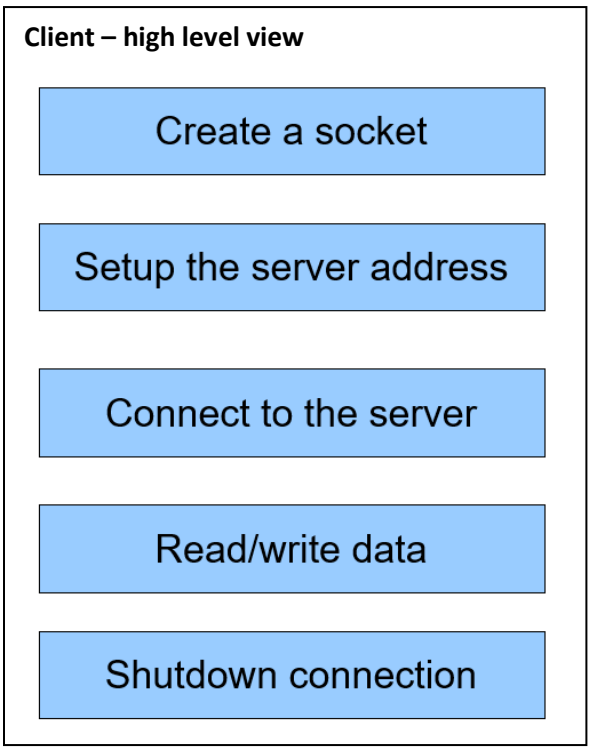
**Addressing**

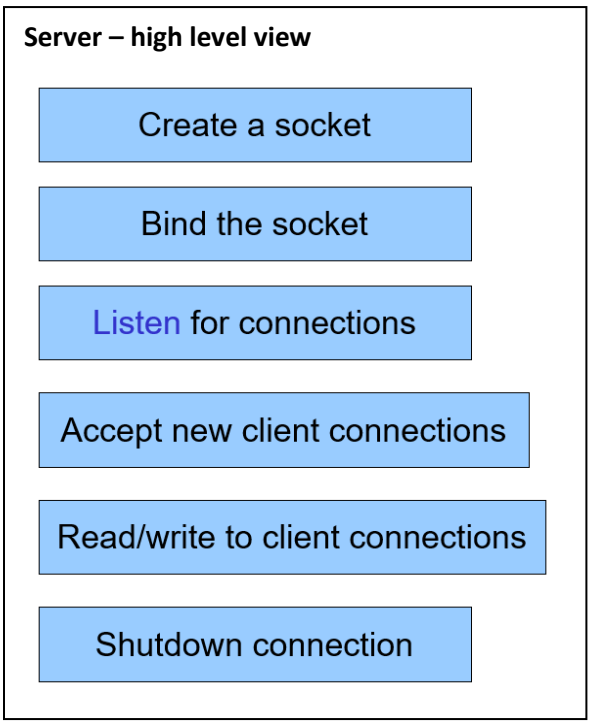




**Addresses, Ports and Sockets**

- Like apartments and mailboxes
- You are the application
- Your apartment building address is the address
- Your mailbox is the port
- The post-office is the network
- The socket is the key that gives you access to the right mailbox




**Comparison between TCP and UDP**

| Parameter | TCP | UDP |
|---|---|---|
| Acronym for | Transmission Control Protocol | User Datagram Protocol or Universal Datagram Protocol |
| Connection | TCP is a connection-oriented protocol. | UDP is a connectionless protocol. |
| Function | As a message makes its way across the internet from one computer to another. This is connection based. | UDP is also a protocol used in message transport or transfer. This is not connection based which means that one program can send a load of packets to another and that would be the end of the relationship. |
| Usage | TCP is suited for applications that require high reliability, and transmission time is relatively less critical. | UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients. |
| Use by other protocols | HTTP, HTTPs, FTP, SMTP, Telnet | DNS, DHCP, TFTP, SNMP, RIP, VOIP. |
| Ordering of data packets | TCP rearranges data packets in the order specified. | UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer. |
| Speed of transfer | The speed for TCP is slower than UDP. | UDP is faster because error recovery is not attempted. It is a "best effort" protocol. |
| Reliability | There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent. | There is no guarantee that the messages or packets sent would reach at all. |

| | | |
|---|---|---|
| **Header Size** | TCP header size is 20 bytes | UDP Header size is 8 bytes. |
| **Common Header Fields** | Source port, Destination port, Check Sum | Source port, Destination port, Check Sum |
| **Streaming of data** | Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries. | Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent. |
| **Weight** | TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control. | UDP is lightweight. There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP. |
| **Data Flow Control** | TCP does Flow Control. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control. | UDP does not have an option for flow control |
| **Error Checking** | TCP does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination. | UDP does error checking but simply discards erroneous packets. Error recovery is not attempted. |
| **Fields** | 1. Sequence Number, 2. AcK number, 3. Data offset, 4. Reserved, 5. Control bit, 6. Window, 7. Urgent Pointer 8. Options, 9. Padding, 10. Check Sum, 11. Source port, 12. Destination port | 1. Length, 2. Source port, 3. Destination port, 4. Check Sum |
| **Acknowledgement** | Acknowledgement segments | No Acknowledgment |
| **Handshake** | SYN, SYN-ACK, ACK | No handshake (connectionless protocol) |