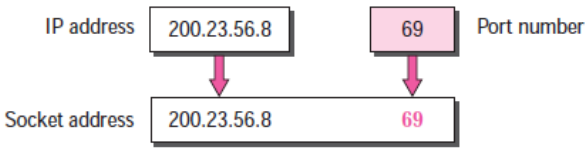
Transport Layer

The first duty of a transport-layer protocol is to provide process-to-process communication. A process is an application-layer entity (running program) that uses the services of the transport layer. Both processes are defined by identifiers called port numbers which are integers between 0 and 65,535. ICANN (Internet Corporation for Assigned Names and Numbers) has divided the port numbers into three ranges:

- **Well-known ports:** The ports ranging from 0 to 1,023 are assigned and controlled by ICANN.
- **Registered ports:** The ports ranging from 1,024 to 49,151 are not assigned or controlled by ICANN. They can only be registered with ICANN to prevent duplication.
- **Dynamic ports:** The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used as temporary or private port numbers.

The client program defines itself with a port number, called the ephemeral port number. The word ephemeral means short lived and is used because the life of a client is normally short. An ephemeral port number is recommended to be chosen randomly from the dynamic range. The server process must also define itself with a port number. This port number, however, cannot be chosen randomly. This port number is usually chosen from the well known region and sometimes from the registered region

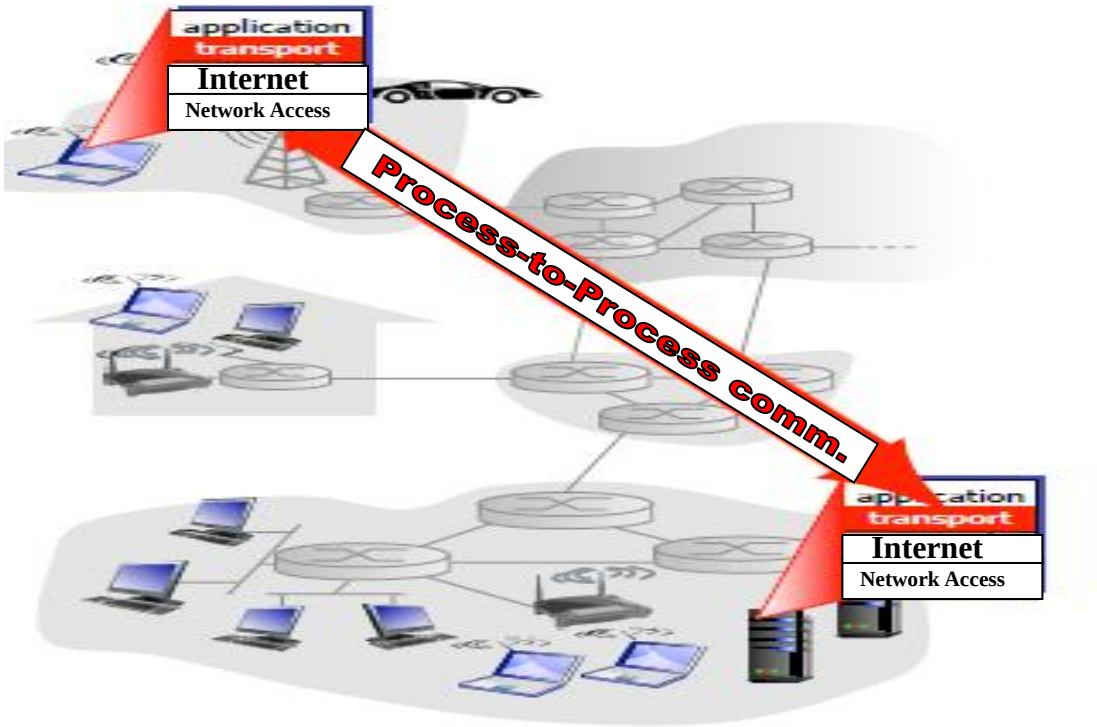
The combination of an IP address and a port number is called a socket address. The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely. Example (200.23.56.8:69) where this server socket address composed of two parts seperated by a column (:) the first division is the host IP address while the second portion refers to the server process.



These two socket addresses (client and server) are part of the network-layer packet header and the transport-layer packet header. The first header contains the IP addresses; the second header contains the port numbers.

Providing logical communication between application processes running on end hosts is the role of transport layer protocols. The most common of these are:

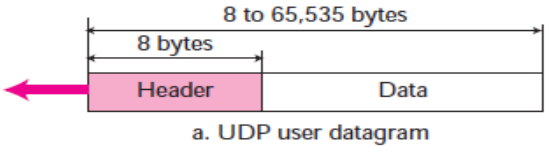
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)



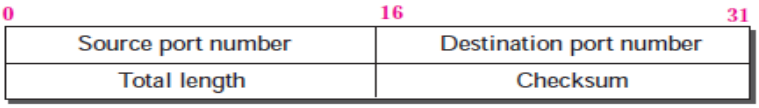
❖ UDP

○ Format:

UDP packet, called user datagram, has a fixed-size header of 8 bytes. This header consists of the following fields:



a. UDP user datagram



b. Header format

- Source port number (2B): This is the port number used by the process running on the source host.
- Destination port number (2B): This is the port number used by the process running on the destination host.
- Length (2B): This is a 16-bit field that defines the total length of the user datagram, header plus data.
- Checksum (2B): This field is used to detect errors over the entire user datagram (header plus data).

○ Properties:

➢ Process-to-Process Communication

UDP provides process-to-process communication using sockets, a combination of IP addresses and port numbers. Common port numbers used by UDP are :

Port number	Process	Brief description
7	Echo	Echoes received datagram back to the sender
13	Daytime	Returns the date and time
53	DNS	Resolves names into IP addresses
161,162	SNMP	Monitoring and management protocol
520	RIP	Routing protocol
1433	MS SQL	Database application
1812	RADIUS	Authentication Protocol
5060	SIP	Voice over IP protocol

➢ Connectionless Services

UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program. The user datagrams are not numbered.

UDP also considerd connectioless because sending data in UDP is so simple; there is no special or additional procecedures for connection establishment or connection terminatin.

➢ No Flow Control

UDP is a very simple protocol. There is no flow control, The receiver may overflow with incoming messages. The lack of flow control means that the process using UDP should provide for this service, if needed.

➢ No Error Control

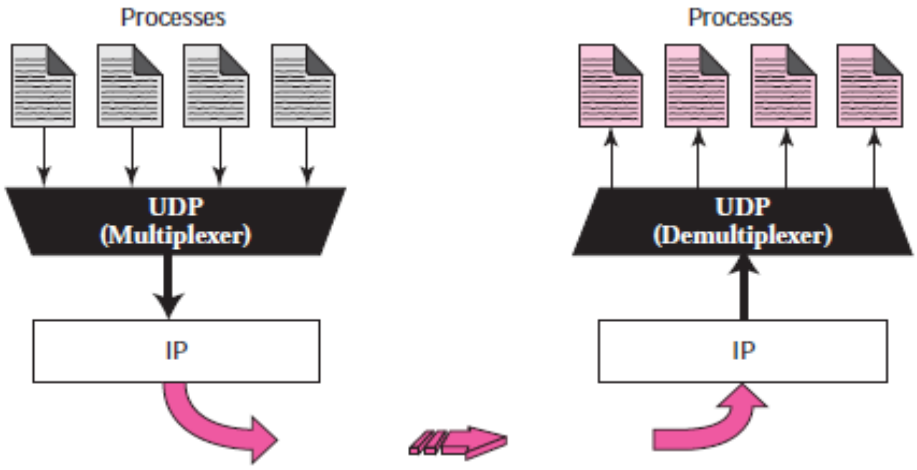
There is no error control mechanism in UDP except for the checksum which is optional in UDP. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded. The lack of error control means that the process using UDP should provide for this service if needed.

➢ Multiplexing and Demultiplexing

In a host running UDP , there is several processes that may want to use the services of UDP. To handle this situation, UDP multiplexes and demultiplexes the processes

At the sender site, This is a many-to-one relationship and requires multiplexing. UDP accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, UDP passes the user datagram to IP.

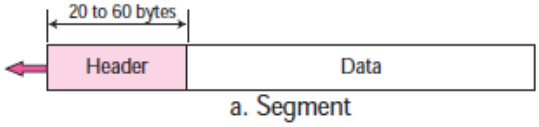
At the receiver there is a one-to-many relationship and requires demultiplexing. UDP receives user datagrams from IP. After error checking and dropping of the header, UDP delivers each message to the appropriate process based on the port numbers.



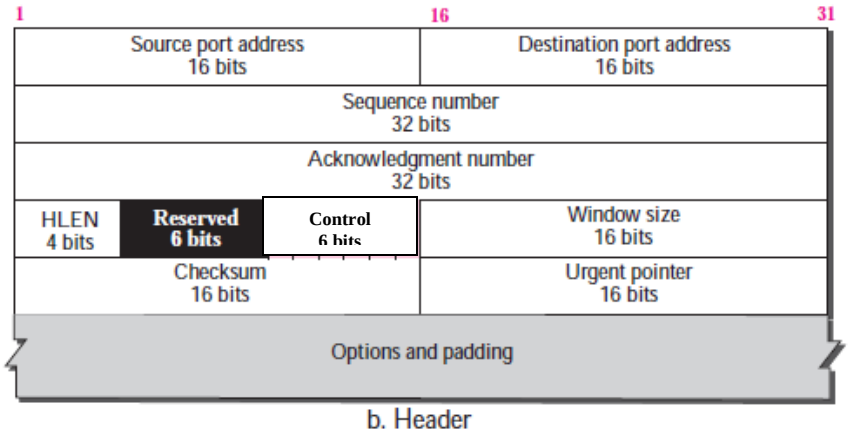
❖ TCP

○ Format:

The PDU of TCP - called segment- consists of a header of 20 to 60 bytes, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes with options. Haerder fields consists of:

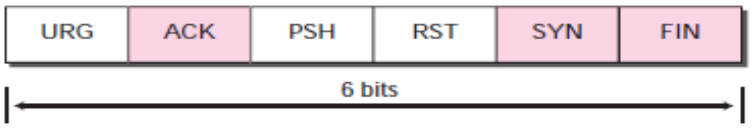


a. Segment



b. Header

- Source port address (2B): This field defines the port number of the process used by running on the source host.
- Destination port address (2B): This field defines the port number of the application program in the host that is receiving the segment.
- Sequence number (4B): This field defines the number assigned to the first byte of data contained in this segment. The sequence number tells the destination which byte in this sequence is the first byte in the segment. On the other hand; this field can indirectly give an idea of the order of the segment.
- Acknowledgment number (4B): it defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully. received byte number (x) from the other party, it returns (x +1) as the acknowledgment number.
- Header length (4b): This field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 and 15.
- Reserved (6b): This is a 6-bit field reserved for future use.
- Control (6b): This field defines 6 different control bits or flags. One or more of these bits can be set at a time. These flags are
    - URG - urgent pointer valid (set when sender wants the receiver to read a piece of data urgently and possibly out of order)
    - ACK - acknowledgement number valid.
    - PSH - receiver should immediately pass the data to the application.
    - RST - reset the connection
    - SYN - synchronize sequence numbers to initiate connection
    - FIN - sender is finished sending data



- Window size (2B): This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

- Checksum (2B): This 16-bit field contains the checksum. However, the use of the checksum in the UDP datagram is optional, whereas the use of the checksum for TCP is mandatory.
- Urgent pointer (2B): This 16-bit field, which is valid only if the urgent flag is set. It is used when the segment contains urgent data. It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
- Options: There can be up to 40 bytes of optional information in the TCP header.

○ Properties:

➢ Process-to-Process Communication

As with UDP, TCP provides process-to-process communication using port numbers. The table below lists some of the port numbers used by TCP.

Port number	Process	Brief description
7	Echo	Echoes received datagram back to the sender
13	Daytime	Returns the date and time
20,21	FTP	File Transfer Protocol
23	Telnet	Remote terminal connection
25	SMTP	e-mail protocol
53	DNS	Resolves names into IP addresses
80	HTTP	Transferring web HTML codes
1863	IM	MSN messenger
8008	HTTP	Alternative port for Transferring web HTML codes

➢ Multiplexing and Demultiplexing

Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver. However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes.

➢ Connection-Oriented Service

TCP, unlike UDP, is a connection-oriented protocol. when a process at a client wants to communicate with the process at the server, the following three phases occur:

1. The two TCPs establish a virtual connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

Note that this is a virtual connection, not a physical connection. The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost, or corrupted, and then resent. Each may be routed over a different path to reach the destination.

➢ Numbering System

Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the sequence number and the acknowledgment number. These two fields refer to a byte number and not a segment number.

▪ Byte Number

TCP numbers all data bytes (octets) that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, TCP stores them in the sending buffer and numbers them. The numbering does not necessarily start from 0. Instead, TCP chooses an arbitrary number between (0) and (2^{32} -1) for the number of the first byte.

▪ Sequence Number

After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte of data carried in that segment.

Example :

Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

Solution :

The following shows the sequence number for each segment:

Segment 1	→	Sequence Number:	10,001	Range:	10,001	to	11,000
Segment 2	→	Sequence Number:	11,001	Range:	11,001	to	12,000
Segment 3	→	Sequence Number:	12,001	Range:	12,001	to	13,000
Segment 4	→	Sequence Number:	13,001	Range:	13,001	to	14,000
Segment 5	→	Sequence Number:	14,001	Range:	14,001	to	15,000

- **Acknowledgment Number**

  The acknowledgment number defines the number of the next byte that the recipient expects to receive. In addition, the acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number. The term cumulative here means that if a party uses (n) as an acknowledgment number, it has received all bytes from the beginning up to (n-1).

➢ Flow Control

TCP, unlike UDP, provides flow control via the usage of window size field. The sending TCP controls how much data can be accepted from the sending process; the receiving TCP controls how much data can to be sent by the sending TCP . This is done to prevent the receiver from being overwhelmed with data.

➢ Error Control

To provide reliable service, TCP implements an error control mechanism. In addition to the checksum calculation process , TCP makes use of the numbering system that has been mentioned previously to overcome the problem of lost or duplicated data.

➢ Reliable Service

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

TCP Connection

TCP is connection-oriented protocol that establishes a virtual path between the source and destination. All of the segments belonging to a message are then sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.

➢ Connection Establishment

TCP transmits data virtually in full-duplex mode. This implies that each party must initialize communication and get approval from the other party before any data are transferred. In TCP this is achieved by a procedure called "Three way handshaking". This procdure occures in the foolwing three steps:

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. The client in our example chooses a random number as the first sequence number and sends this number to the server. This sequence number is called the initial sequence number (ISN). Note that this segment does not contain an acknowledgment number.

   Note:

   Although that the SYN segment is a control segment and carries no data, it consumes one sequence number. When the data transfer starts, the ISN is incremented by 1. We can say that the SYN segment carries no real data, but we can think of it as containing one imaginary byte.

2. The server sends the second segment, a SYN + ACK segment with two flag bits set: SYN and ACK. This segment has a dual purpose. First, it is a SYN segment for communication in the other direction. The server uses this segment to initialize a sequence number for numbering the bytes sent from the server to the client. The server also acknowledges the receipt of the SYN segment from the client by setting the ACK flag and displaying the next sequence number it expects to receive from the client.

3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

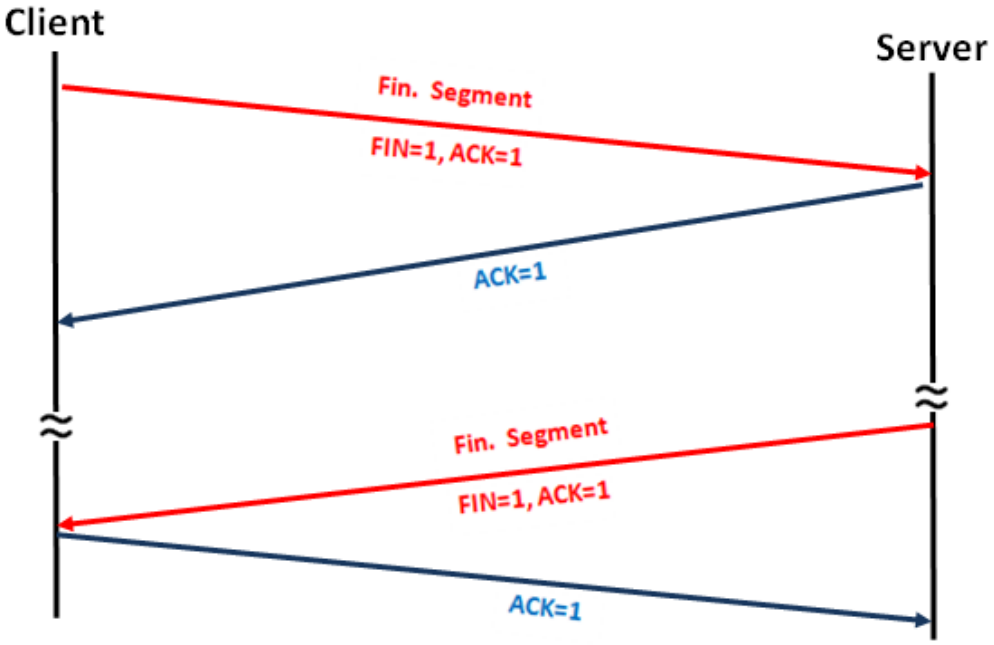
➢ Connection Termination

Any of the two parties involved in exchanging data (client or server) can close the connection, This usually done by a procedure called " half-close". In this procedure, one end can stop sending data while still receiving data. This is why it is called a half-close.

Either the server or the client can issue a half-close request. For example; if the client half-closes a connection. It sending a FIN segment. The server accepts the half-close by sending the ACK segment.

After half closing the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server. The client cannot send any more data to the server.

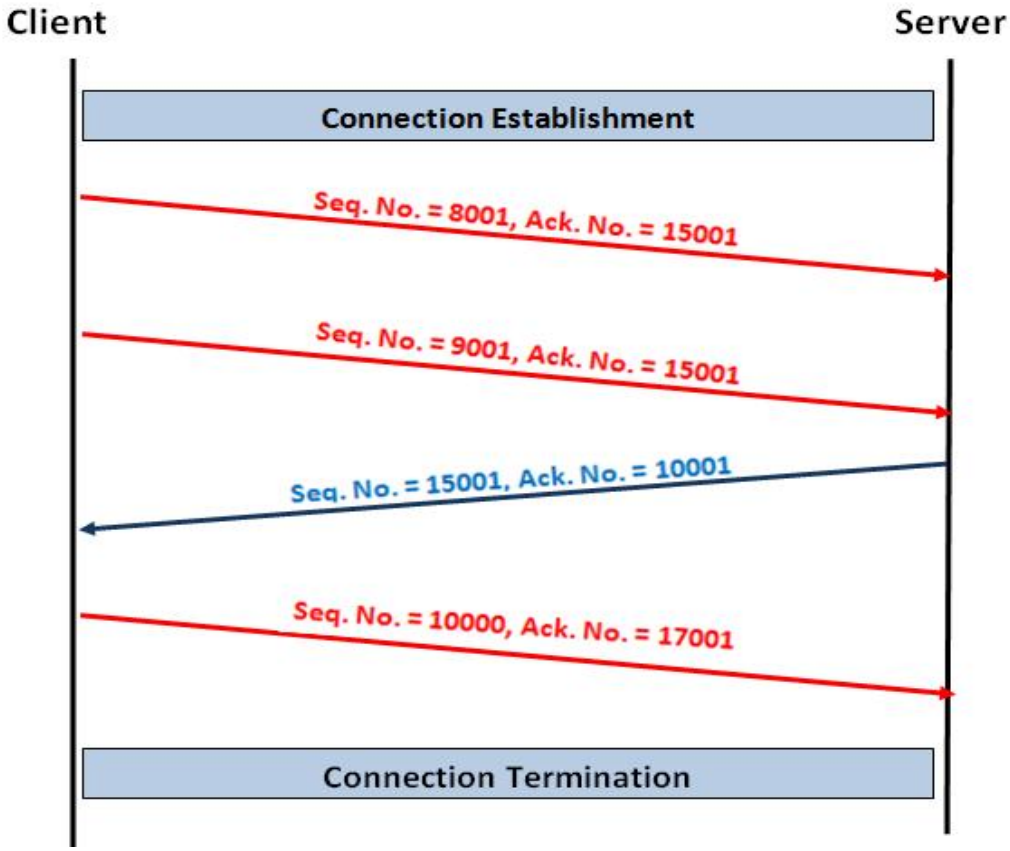
When the server has sent all of the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.



➢ Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can send data and acknowledgments in both directions. it is enough to know that data traveling in the same direction as an acknowledgment are carried on the same segment.

The following figure illustrates the process of data transfer. In this example, after a connection is established, the client sends 2,000 bytes of data in two segments. The server then sends 2,000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there is no more data to be sent.



Note

If a segment does not carry user data, it does not logically define a sequence number. The field is there, but the value is not valid. however, some segments, when carrying only control information, need a sequence number to allow an acknowledgment from the receiver. Each of these segments consume one sequence number as though it carries one byte, but there are no actual data.