UNIT 3 TCP : TRANSMISSION CONTROL PROTOCOL

Structure

3.0 INTRODUCTION

When two computers wish to exchange information over a network, there are several components that must be in place before the data can actually be sent and received. Of course, the physical hardware must exist, which is typically either a network interface card (NIC) or a serial communications port for dial-up networking connections. Beyond this physical connection, however, computers also need to use a protocol which defines the parameters of the communication between them. In short, a protocol defines the "rules of the road" that each computer must follow so that all of the systems in the network can exchange data. One of the most popular protocols in use today is TCP/IP, which stands for Transmission Control Protocol/Internet Protocol.

By convention, TCP/IP is used to refer to a suite of protocols, all based on the Internet Protocol (IP). Unlike a single local network, where every system is directly connected to each other, an internet is a collection of networks, combined into a single, virtual network. The Internet Protocol provides the means by which any system on any network can communicate with another as easily as if they were on the same physical network. Each system, commonly referred to as a host, is assigned a unique 32-bit number which can be used to identify it over the network. Typically, this address is broken into four 8-bit numbers separated by periods. This is called dot-notation, and looks something like "192.43.19.64". Some parts of the address are used to identify the network that the system is connected to, and the remainder identifies the system itself. There are three "classes" of addresses, referred to as "A", "B" and "C". The rule of thumb is that class "A" addresses are assigned to very large networks, class "B" addresses are assigned to medium sized networks, and class "C" addresses are assigned to smaller networks (networks with less than approximately 250 hosts).

When a system sends data over the network using the Internet Protocol, it is sent in discrete units called datagrams, also commonly referred to as packets. A datagram consists of a header followed by application-defined data. The header contains the addressing information which is used to deliver the datagram to its destination, much like an envelope is used to address and contain postal mail. And like postal mail, there is no guarantee that a datagram will actually arrive at its destination. In fact, datagrams may be lost, duplicated or delivered out of order during their travels over the network. Needless to say, this kind of unreliability can cause a lot of problems for software developers. Whats really needed is a reliable, straight-forward way to exchange data without having to worry about lost packets or jumbled data.

To fill this need, the Transmission Control Protocol (TCP) was developed. Built on top of IP, TCP offers a reliable, full-duplex byte stream which may be read and written to in a fashion similar to reading and writing a file. The advantages to this are obvious: the application programmer doesn't need to write code to handle dropped or out-of-order datagrams, and instead can focus on the application itself. And because the data is presented as a stream of bytes, existing code can be easily adopted and modified to use TCP.

TCP is known as a connection-oriented protocol. In other words, before two programs can begin to exchange data they must establish a "connection" with each other. This is done with a three-way handshake in which both sides exchange packets and establish the initial packet sequence numbers (the sequence number is important because, as mentioned above, datagrams can arrive out of order; this number is used to ensure that data is received in the order that it was sent). When establishing a connection, one program must assume the role of the client, and the other the server. The client is responsible for initiating the connection, while the server's responsibility is to wait, listen and respond to incoming connections. Once the connection has been established, both sides may send and receive data until the connection is closed.

3.1 OBJECTIVES

After going through the Unit, you will be able to understand

- Important Features of TCP
- the basic difference between TCP & UDP
- how they function;
- advantages and disadvantages of one over the other.

The end of the chapter emphasis on Domain Name System; Row names are given over internet in such a manner that they are unique.

3.2 BASIC TERMINOLOGY

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service one must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A use 8 bits for the network address with 24 bits left over for other addressing.
Class B uses 16 bit network addressing.
Class C uses 24 bit network addressing and class D uses all 32.

Host address

8 bits are finally used for host addresses within subnet (if it exists-typically for class B network addressing). This places a limit of 256 machines that can be on the subnet.

Total address

The 32 bit address is usually written as 4 integers separated by dots.

Symbolic names

Each host has a name. This can be found from the user level command on each terminal.

Hostname

A symbolic name for the network also exists that is used for access to the host.

TCP Header

TCP provides a reliable stream delivery and virtual connection service to applications through the use of sequenced acknowledgment with retransmission of packets when necessary. The following figure shows TCP header structure:

TCP header structure

32 bits

Source port	Destination port
Sequence number	
Acknowledgement number	
Offset Reserved U A P R S F	Window
Checksum	Urgent pointer
Option + Padding	
Data	

Figure 1: TCP Header Structure

Source Port: Source port number

Destination Port: Destination port number

Sequence Number: The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present, the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

Acknowledgement Number

If the ACK control bit is set, this field contains the value of the next sequence number which the sender of the segment is expecting to receive. Once a connection is established, this value is always sent. Both the sequence number and acknowledgement number are 32 bit long.

4 bits. The number of 32-bit words in the TCP header, which indicates where the data begins. The TCP header (even one including options) has a length which is an integral number of 32 bits.

Reserved
6 bits Reserved for future use. Must be zero.

Control Bits
6 bits. The control bits may be (from right to left):
U (URG)
Urgent pointer field significant.
A (ACK)
Acknowledgement field significant.
P (PSH)
Push function.
R (RST)
Reset the connection
S (SYN)

Synchronize sequence numbers.
F (FIN)
No more data from sender.

Window
16 bits. The number of data octets, which the sender of this segment is willing to accept, beginning with the octet indicated in the acknowledgment field.

Checksum
16 bits. The checksum field is the 16 bit one's complement of the one's complement sum of all 16-bit words in the header and text. If a segment contains an odd number of header and text octets to be check summed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

Urgent Pointer

16 bits. This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field can only be interpreted in segments for which the URG control bit has been set.

Options

Options may be transmitted at the end of the TCP header and always have a length which is a multiple of 8 bits. All options are included in the checksum. An option may begin on any octet boundary.

There are two possible formats for an option.

A single octet of option type

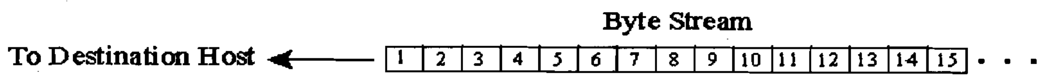
An octet of option type, an octet of option length, and the actual option data octets. The option length includes the option type and option length, as well as the option data octets. The list of options may be shorter than that designated by the data offset field because the contents of the header beyond the End-of-Option option must be header padding i.e., zero.

A TCP must implement all options.

3.3 IMPORTANT FEATURES OF TRANSMISSION CONTROL PROTOCOL (TCP)

TCP provides reliable, end-to-end data transmission with flow control

- Examples of TCP applications include Telnet, FTP, WWW, POP, IMAP, etc.
- Basic features of TCP transmission:
  - √ **Streamed Data:** Data from sender to receiver organized as a *stream* of bits divided into 8-bit bytes (data streams have no TCP imposed structure)



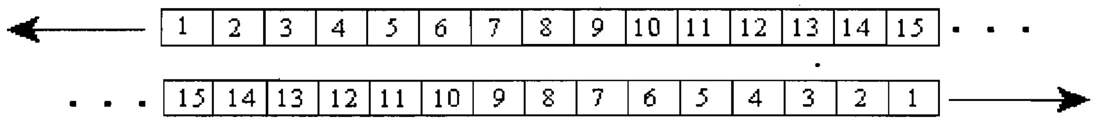
√ **Connection Oriented**

- Client host "calls" server host at a specific destination port;
- If receiver accepts call, a connection is established between the corresponding client and server applications;

- Information is transferred bi-directionally;
- Connection is closed ("call terminated") when client or server application finished or when certain communications errors detected.

  √ **Buffered Transfer:** Applications send bytes to TCP software that delivers the stream of bytes in exactly the order sent (not necessarily grouped the same way)

  √ **Full-duplex Transmission:** Both hosts can send and receive data and control information independently.



- The unit of transfer in TCP is called a **segment**
- TCP Segment Encapsulation.

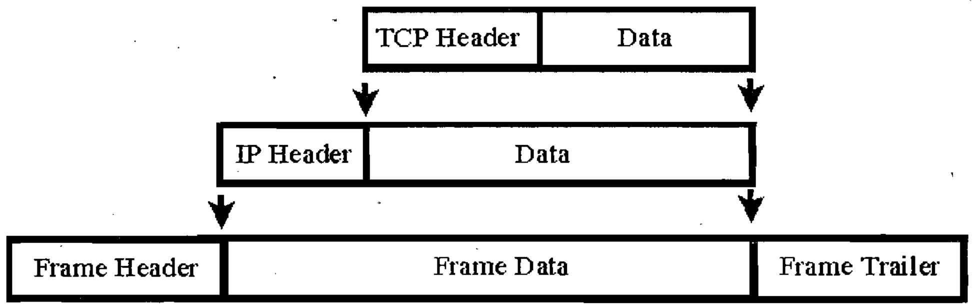


Figure 2: Encapsulation of Segment

- TCP Segment Format

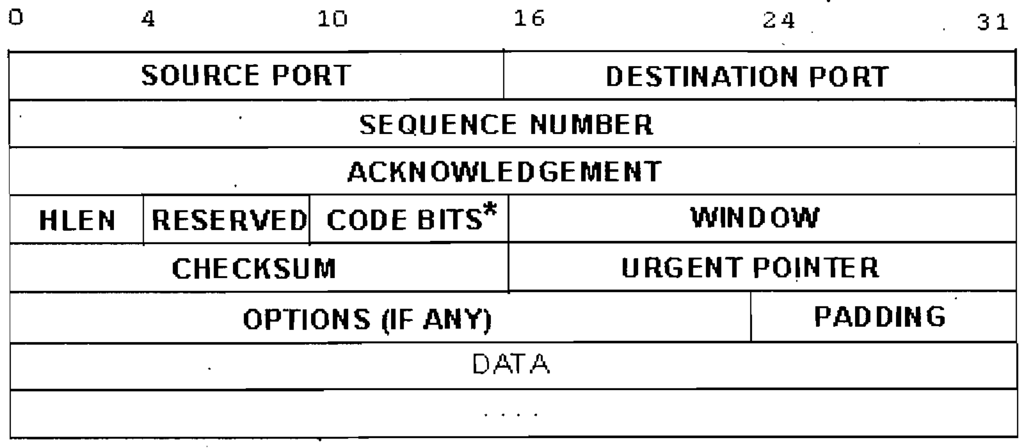


Figure 3 : Segment Format

- Connection defined by the pair of numbers (**source IP, source port**) and (**dest IP, dest port**).

- Different connections can use the same destination port on server host as long as the source ports or source IPs are different

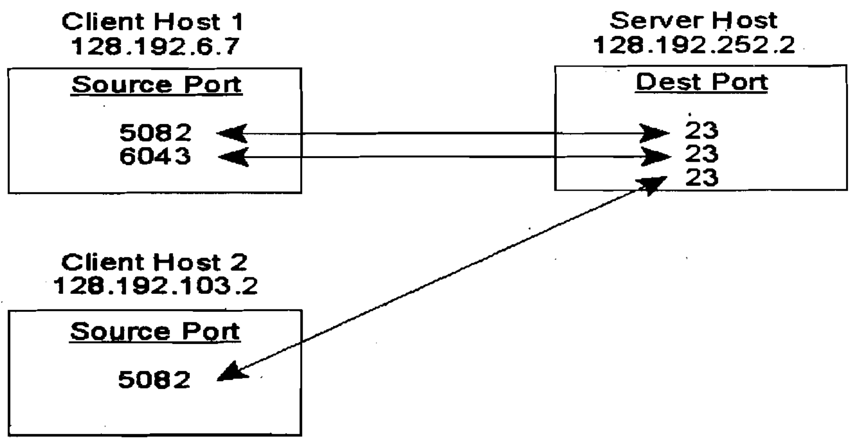


Figure 4: Use of same port by different connections

- TCP breaks data stream into segments

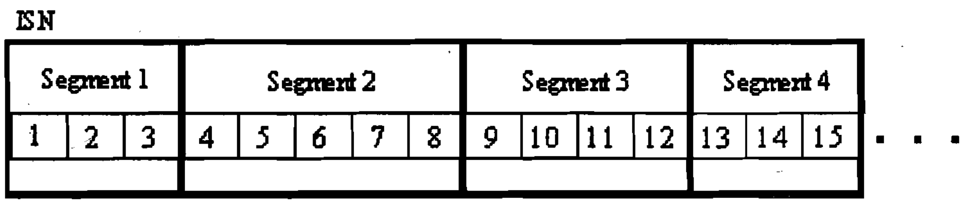


Figure 5: Breaking data into Segments

- Sequence numbers used to place received segment data in the correct order
  - √ Initial sequence number (ISN) marks the beginning of data stream
  - √ ISN is random and negotiated when connection is established
- Acknowledgement numbers tell sender that receiver expects next segment

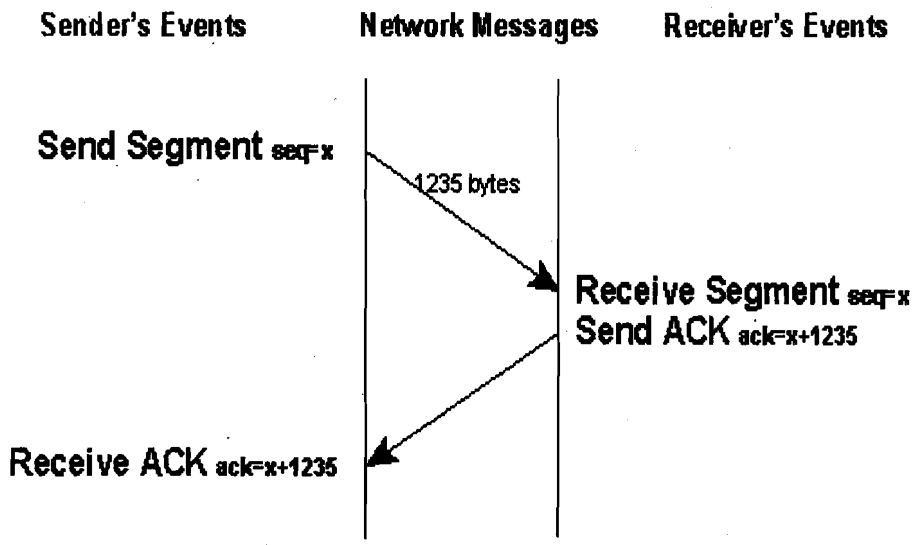


Figure 6: Exchange of Segment

- When a segment is sent, a timer is started; if an ACK has not been received when the timer expires, the segment is resent

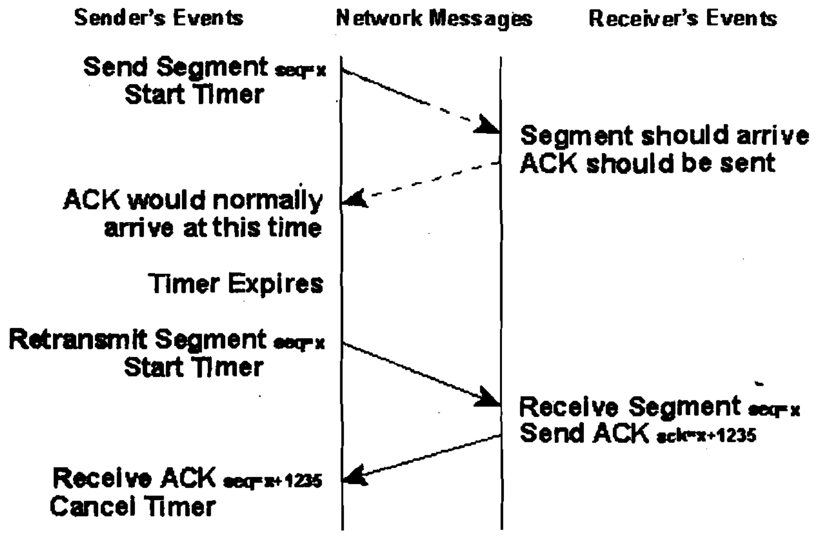


Figure 7: Retransmission of Segment on time-over.

- Sliding windows are used to transmit data stream efficiently and for flow control

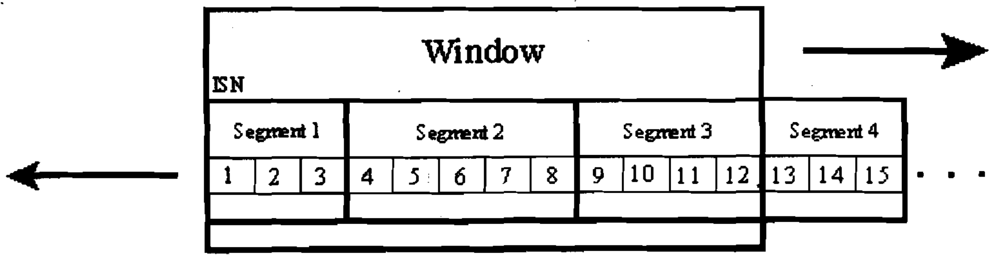


Figure 8: Sliding Window

- All segments within the window are sent without waiting for acknowledgement (efficient transmission)

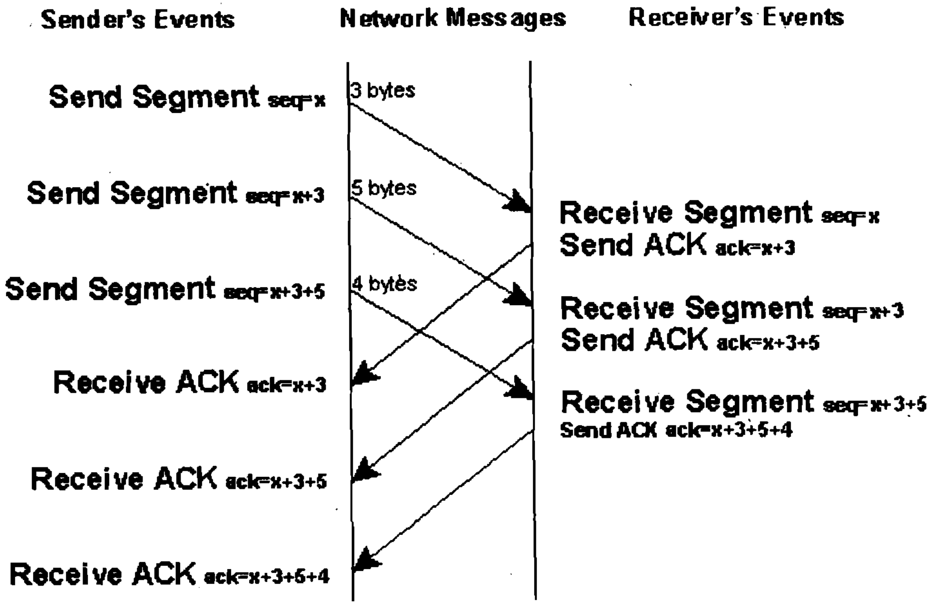


Figure 9: Dispatch of segment without any acknowledgement

- Window slides as acknowledgements received

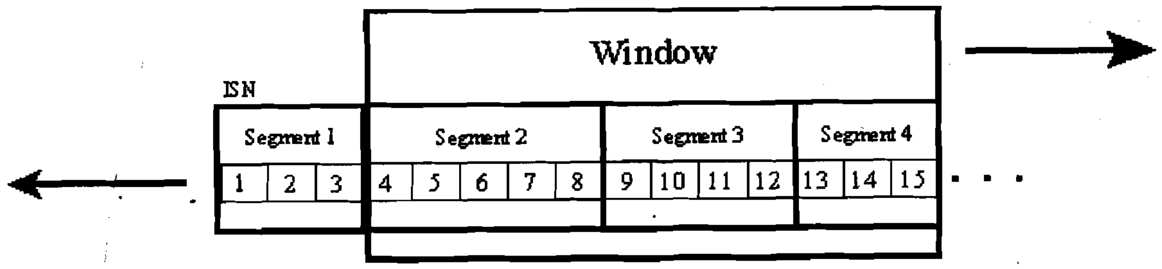
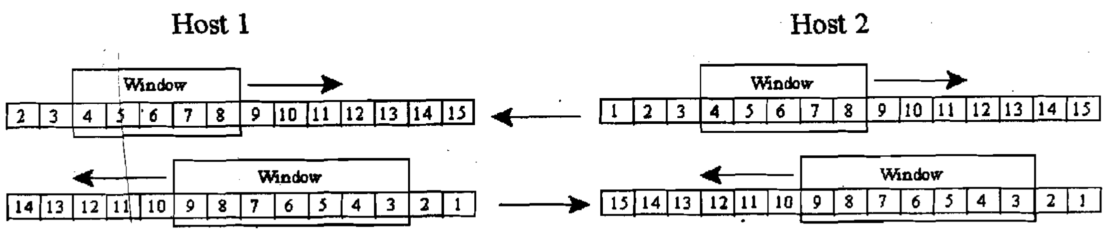


Figure 10: Status of Sliding Window

- Receiver sends acceptable window size to sender during each segment transmission (flow control)
  - ✓ if too much data being sent, acceptable window size is reduced
  - ✓ if more data can be handled, acceptable window size is increased
- Actually four windows used, i.e., send and receive windows in both directions (full-duplex)



- A TCP connection is established using a "three-way handshake"

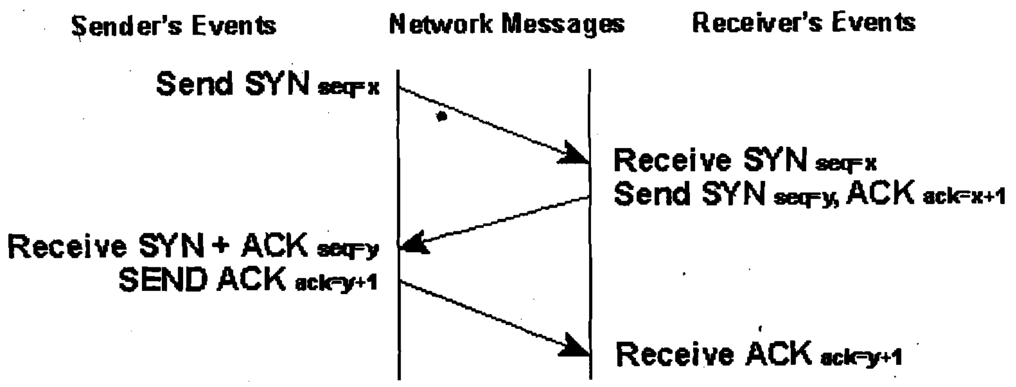
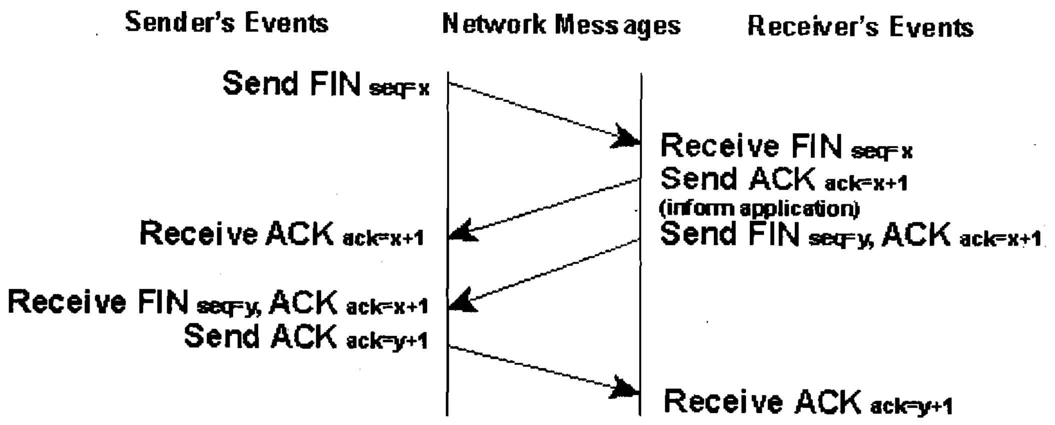


Figure 11: Three way Handshake

- A TCP connection is closed using a “modified three-way handshake”
- A TCP connection is aborted via a connection reset (RST bit set in the CODE field)



·Figure 12: Modified Three way handshake

3.4 USER DATAGRAM PROTOCOL (UDP)

Provides an unreliable datagram service to network applications.

Unlike TCP, the User Datagram Protocol (UDP) does not present data as a stream of bytes, nor does it require that you establish a connection with another program in order to exchange information. Data is exchanged in discrete units called datagrams, which are similar to IP datagrams. In fact, the only features that UDP offers over raw IP datagrams are port numbers and an optional checksum.

UDP is sometimes referred to as an unreliable protocol because when a program sends a UDP datagram over the network, there is no way for it to know that it actually arrived at its destination. This means that the sender and receiver must typically implement their own application protocol on top of UDP. Much of the work that TCP does transparently (such as generating checksums, acknowledging the receipt of packets, retransmitting lost packets and so on) must be performed by the application itself.

With the limitations of UDP, you might wonder why it's used at all. UDP has the advantage over TCP in two critical areas: speed and packet overhead. Because TCP is a reliable protocol, it goes through great lengths to insure that data arrives at its destination intact, and as a result it exchanges a fairly high number of packets over the network. UDP doesn't have this overhead, and is considerably faster than TCP. In those situations where speed is paramount, or the number of packets sent over the network must be kept to a minimum, UDP is the solution.

Now let us look at UDP header format

3.4.1 UDP Header

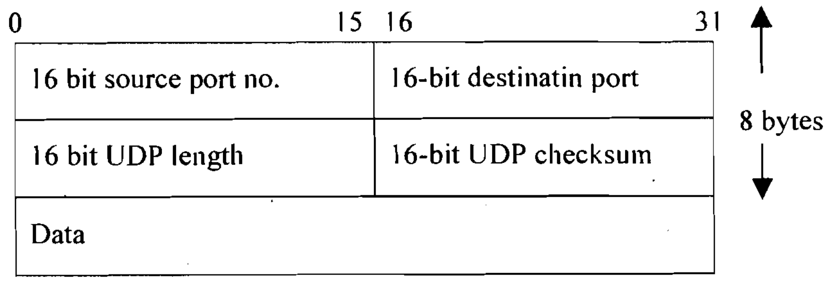


Figure 13: UDP Header format

Port nos. identify the sending process and receiving process. TCP & UDP use the destination port no. to demultiplex incoming data from IP.

The UDP length field is the length of UDP header and the UDP data in bytes. The min. value for this field is 8 bytes. UDP checksum covers the UDP header and the UDP data.

3.4.2 Characteristics of User Datagram Protocol (UDP)

The following are the important characteristics of UDP

- UDP provides an unreliable, connectionless delivery;
- Application programs using UDP are responsible for message loss, duplication, delay, out-of-order delivery, and loss of connectivity;
- Examples of applications that use UDP transport include Network Time Protocol (NTP), Sun's Network File System (NFS), and the Simple Network Management Protocol (SNMP);
- Each UDP message is called a **user datagram.**
- UDP Message Encapsulation.

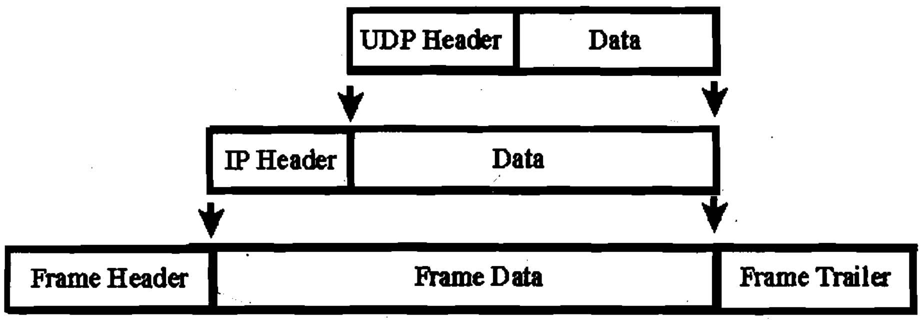


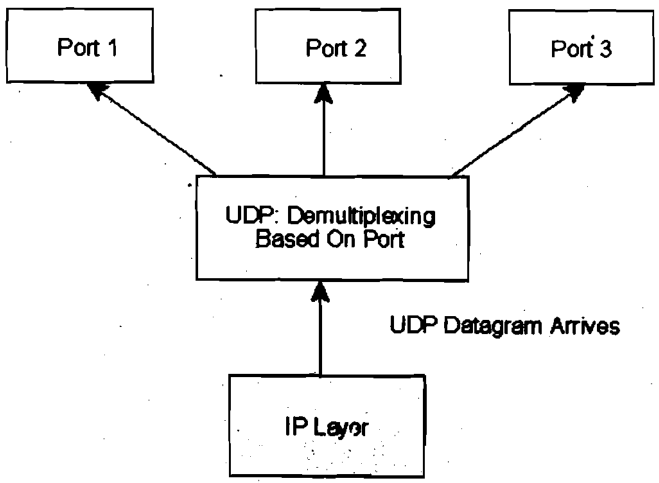
Figure 14: UDP Message Encapsulation

- UDP Message Format

0	16 31
UDP SOURCE PORT	UDP DESTINATION PORT
UDP MESSAGE LENGTH	UDP CHECKSUM
DATA	

Figure 15: UDP Message Format

- UDP messages are stored in queues on destination host, one queue per destination port



Check Your Progress

1) TCP offers a __________ __________byte stream which can easily be read or written

2) ______ is known as a connection oriented protocol

3) The unit of transfer in TCP is called a __________

4) Internet uses ______ bits integer for storing the IP address out or which

Class	A	uses	____	bits
Class	B	uses	___	bits and
Class	C	uses	___	bits

5) For network addressing UDP proves an ______ ________ delivery. Each UDP message is called a __________.

3.5 SUMMARY

TCP is a connection, oriented, reliable full duplex transmission protocol above IP (internet protocol). It provides a virtual circuit that two processes can use to communicate. It uses sequenced acknowledgement with retransmission of packets when necessary.

UDP provides an unreliable, connectionless, datagram service to network applications.

Domain Name System (DNS).provides mapping between names for Internet resources and their associated IP addresses.

In the end of the Unit the process of establishing on FTP session is explained.

3.6 MODEL ANSWERS

1. Reliable, Duplex

2. TCP

3. Segment

4. 32

   8, 16, 24

5. Unreliable, Connectionless, User datagram