Unit :- 2 TCP/IP

Protocol - II

Page
Date

* Internet protocol :- IP(v4)

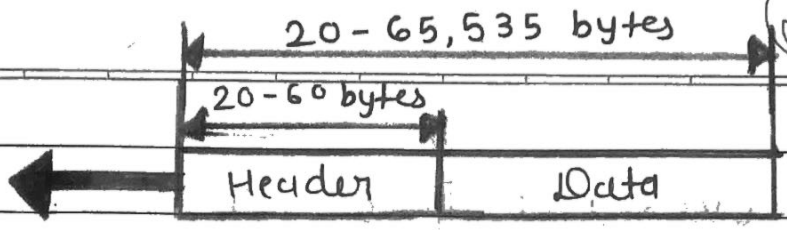
→ IP is an unrelible and connectionless datagram protocol -- a best - effort delivery service. The term best - effort means that IP packets can be corrupted, lost, arrive out of order or delayed and may create congestion for the network.

→ If reliability is important, IP must be paired with a reliable protocol such as TCP.

* Datagram :

→ Packets in the Network (internet) layer are called datagrams.

• IP Datagram format :-



0 3	4 7	8 15	16 31
VER 4 bits	HLEN 4 bits	Service type 8 bits	Total length 16 bits
Identification 16 bits		Flag 3 bits	Fragmentation offset 13 bits
Time to live 8 bits	Protocol 8 bits	Header checksum 16 bits	
Source IP address			
~~Option~~ Destination IP address			
Options + padding (0 to 40 bytes)			

b. Header Format

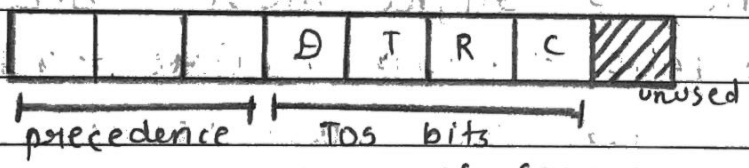
→ **Version (VER.)** : This 4-bit field defines the version of the IP protocol. currently the version 4. All fields must be interpreted as specified in the fourth version of the protocol. If the machine is using some other version of IP, the datagram is discarded rather than interpreted incorrectly.

→ **Header Length (HLEN.)** :- This 4-bit field defines total length of the diagram header in 4 byte words. This field is needed because the length of the header is variable (between 20 and 60 byte).

→ When there are no options, the header length is 20 byte, and the value of this field is 5 (5 x 4 = 20) when the option field is at its maximum size; the value of this field is 15 (15 x 4 = 60).

→ service type :- ~~The older~~ This field was changed from differenciated services to the service type by IETF (Internet engineering Task force)

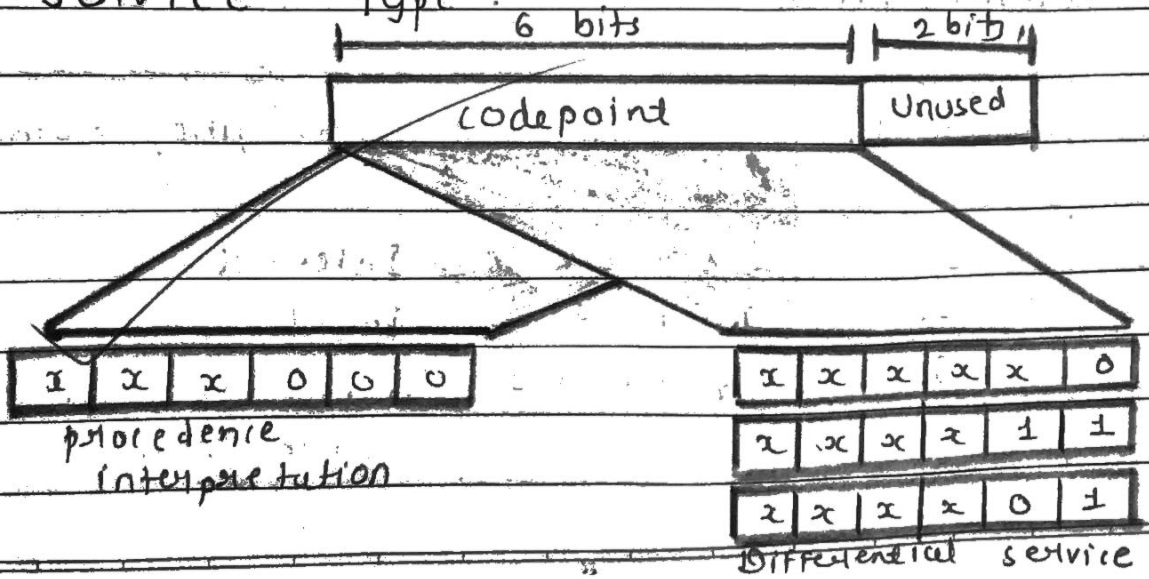
- DIFFerenciated services :-



imp (e)

D : Minimize Delay
T : Maximize Throughput
R : Maximize Reliability
C : Minimize cost

- service type :-



→ In service bit the first six bit make up the codepoint subfield and the last 2 bits are not used.

→ the codepoint sub field in two different ways:

(a) When the 3 right-most bit are 0s, the 3-left-most bits are interpreted the same as the precedence bits in the service type interpretation.

(b) When the 3 right-most bits are not all 0s, the 6 bit define 56(46-8) services based on the priority assignment by the internet or local authorities.

category	codepoint	Assigning Authority
1	x x x x x 0	Interned
2	x x x x 1 1	local
3	x x x x 0 1	Temporary or experimental

→ Total length :- This is a 16-bit field that defines the total length (header plus data) of the IP datagram in bytes.

Length of data = total length - header length

→ Time to live :-

Imp

A datagram has a limited lifetime in its travel through an internet. This field was originally designed to hold a timestamp, which was decremented by each visited router. The data gram was discarded when the value became zero.

→ However, for this scheme, all the machines must have synchronized clocks and must know how long it takes for a datagram to go from one machine to another. Today, this field is mostly used to control the maximum number of hops (routers) visited by the data-gram.

→ When a source host sends the datagram, it stores a number in this field. This value is approximately two times the maximum number of routes between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero the router discards the datagram.

→ This field is needed because routing table in the internet can become corrupted.

* **Protocol :-**

→ This 8-bit field defines the higher-level protocol that uses the services of IP layer such as TCP or UDP.

* **Header checksum :-**

19/12/22

→ The checksum is the error detection method.

5m

→ The checksum protects against the corruption that may occur during the transmission of a packet.

- For example :-

	At Sender side	At Receiver side
	1 0 1 0 1 1 0 0	1 0 1 0 1 1 0 0
	1 1 0 0 1 1 1 0	1 1 0 0 1 1 1 0
	1 0 0 0 0 0 1 0	1 0 0 0 0 0 1 0
sum	0 1 1 1 1 1 0 0	1 0 0 0 0 0 1 1
checksum	1 0 0 0 0 0 1 1	sum 1 1 1 1 1 1 1 1
		0 0 0 0 0 0 0 0

→ if the final checksum is all zeros, there is no error in the receive data.

* Source IP address :-

→ This 32-bit field defines the IP address of the source. This field must remain unchanged during the

time the IP datagram travels from the source host to the destination host.

* **Destination address:-**

→ This 32-bit field defines the IP address of the destination. This field must remain unchanged during the IP datagram travels from the source host to the destination host.

6m * **Fragmentation:-** Very Imp

→ A datagram can travel through different networks. Each router decapsulates the IP datagram from the frame it receives, processes it, and then encapsulates it in another frame.

→ The format and size of the received frame depend on the protocol used by the physical network through which the frame has just traveled.

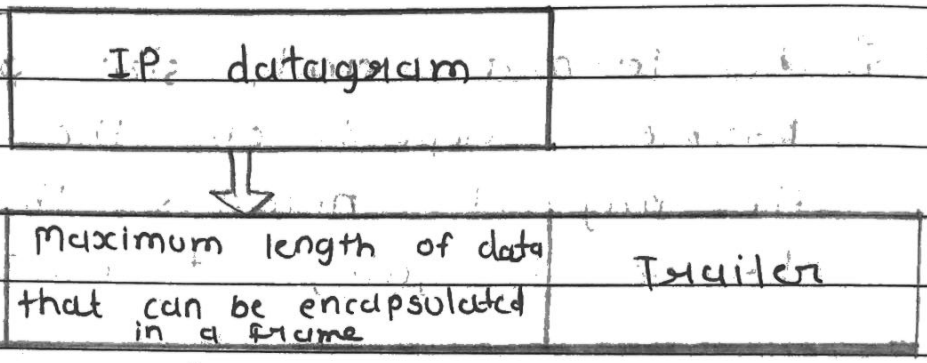
→ The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel.

→ for example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

* **Maximum Transfer Unit (MTU) :-**

→ Each data link layer protocol has its own frame format in most protocols. One of the fields defined in the format is maximum size of the data field.

→ In other words, when a datagram encapsulated in a frame, the total size of the datagram must be less than this maximum size.



→ The value of the MTU differs from one physical network protocol to another.

- For example, the value for

Ethernet LAN is 1500 bytes
FDDI LAN is 4352 bytes
PPP is 296 bytes.

→ Definition:

The division of a packet into smaller units to accomodate a protocol's ~~entry~~ MTU is called fragmentation.

→ Note: Only data in a datagram is fragmented.

* Fields Related to fragmentation :-

(1) Identification
(2) Flags
(3) Fragmentation offset.

* Identification :-

The combination of the identification and source IP address must uniquely define a datagram as it leaves the datagrams.

When the IP protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by one. As long as the counter is kept in the main memory, uniqueness is guaranted.

When a datagram is fragmented, the value in the identification field is copied into all fragmentation

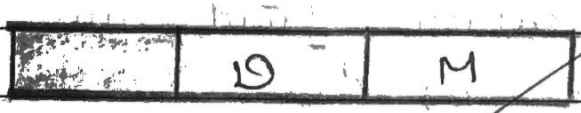
→ In other words, all fragments have the same identification number, ~~helps the destination~~ which is also the same as the original datagram. The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value should be assembled into one datagram.

* Flags:-

MC9

→ This is three-bit field.

D : Do not fragment
M : More fragment



• D : Do not fragment.

→ if D is 1, the machine must not fragment the datagram.

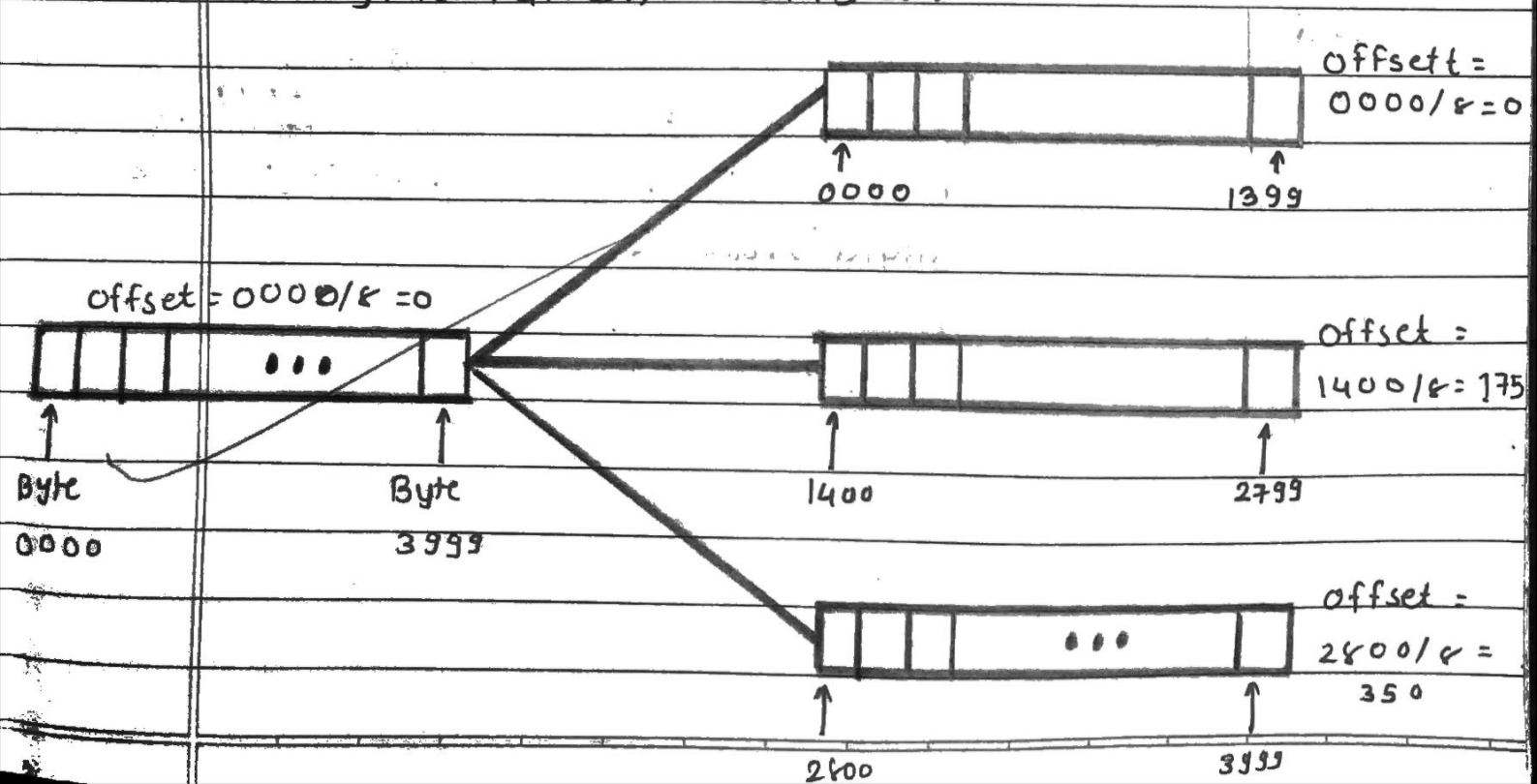
→ if D is 0, the datagram can be fragmented if necessary.

- M : More fragment.

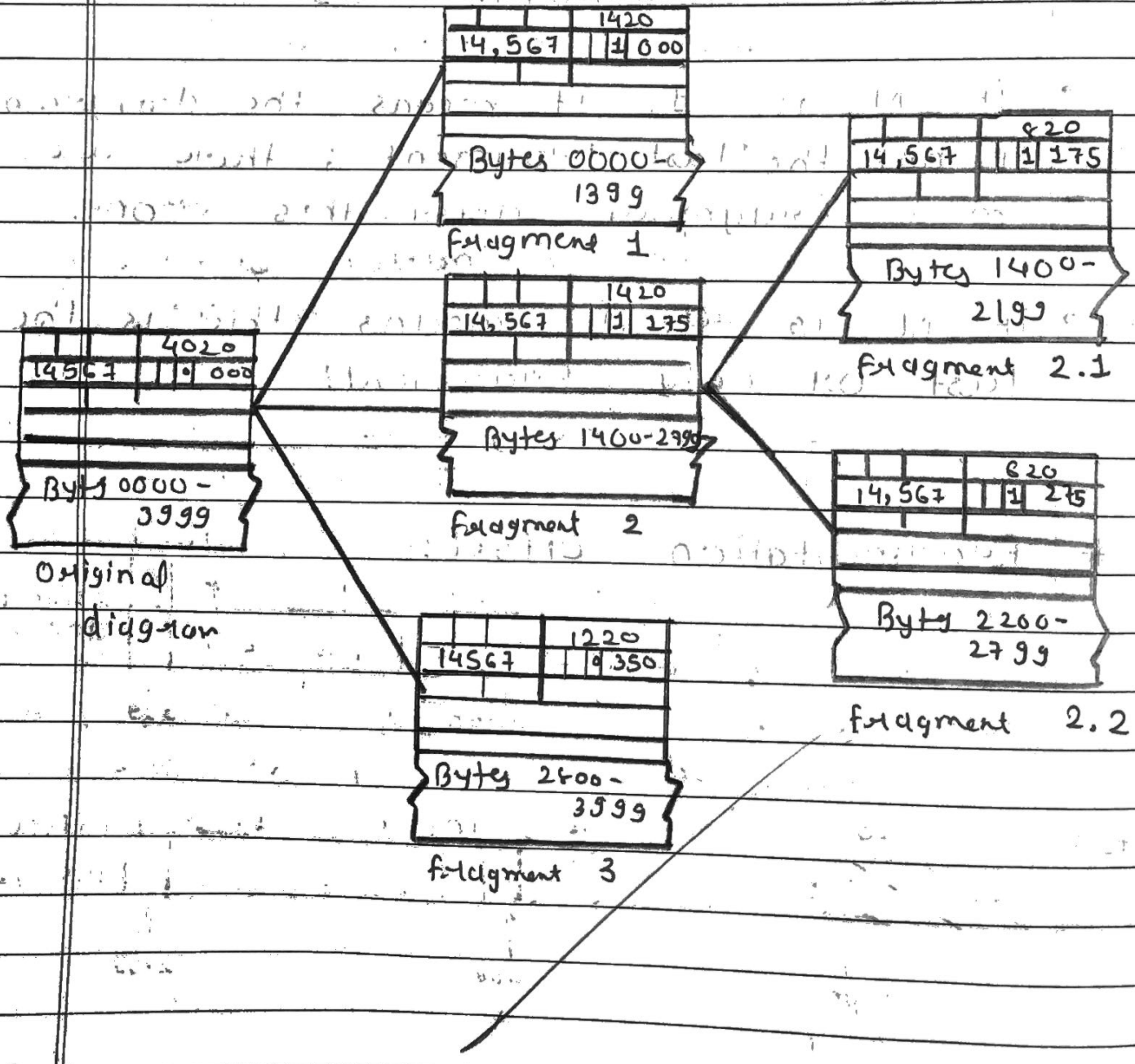
→ if M is 1, it means the datagram is not the last fragment ; there are more fragments after this one.

→ if M is 0, it means this is the last or only fragment.

* Fragmentation offset :-



→ This 13-bit field shows the relative position of fragment with respect to the whole datagram.



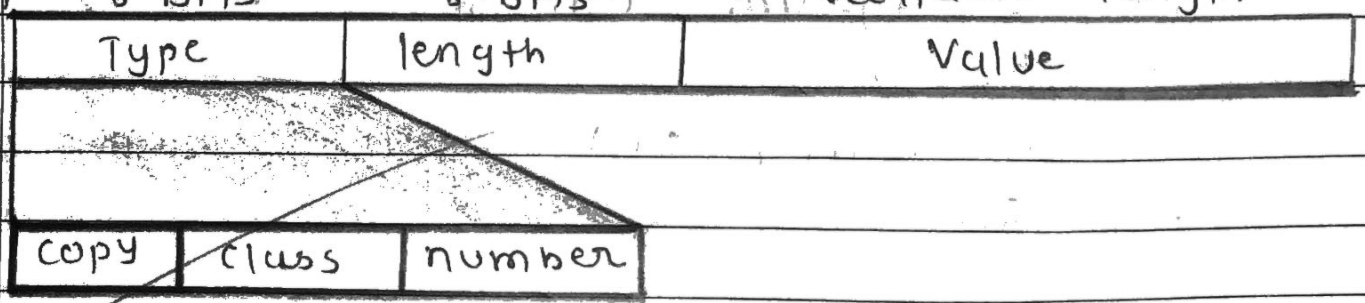
* Options :-

→ The header of IP datagram is made of two parts : a fixed part and a variable part. The fixed part is 20 bytes long.

→ The variable part comprises the options, which can be a maximum of 40 bytes

→ Options can be used for network testing and debugging.

* Option Format :-



* Type :-

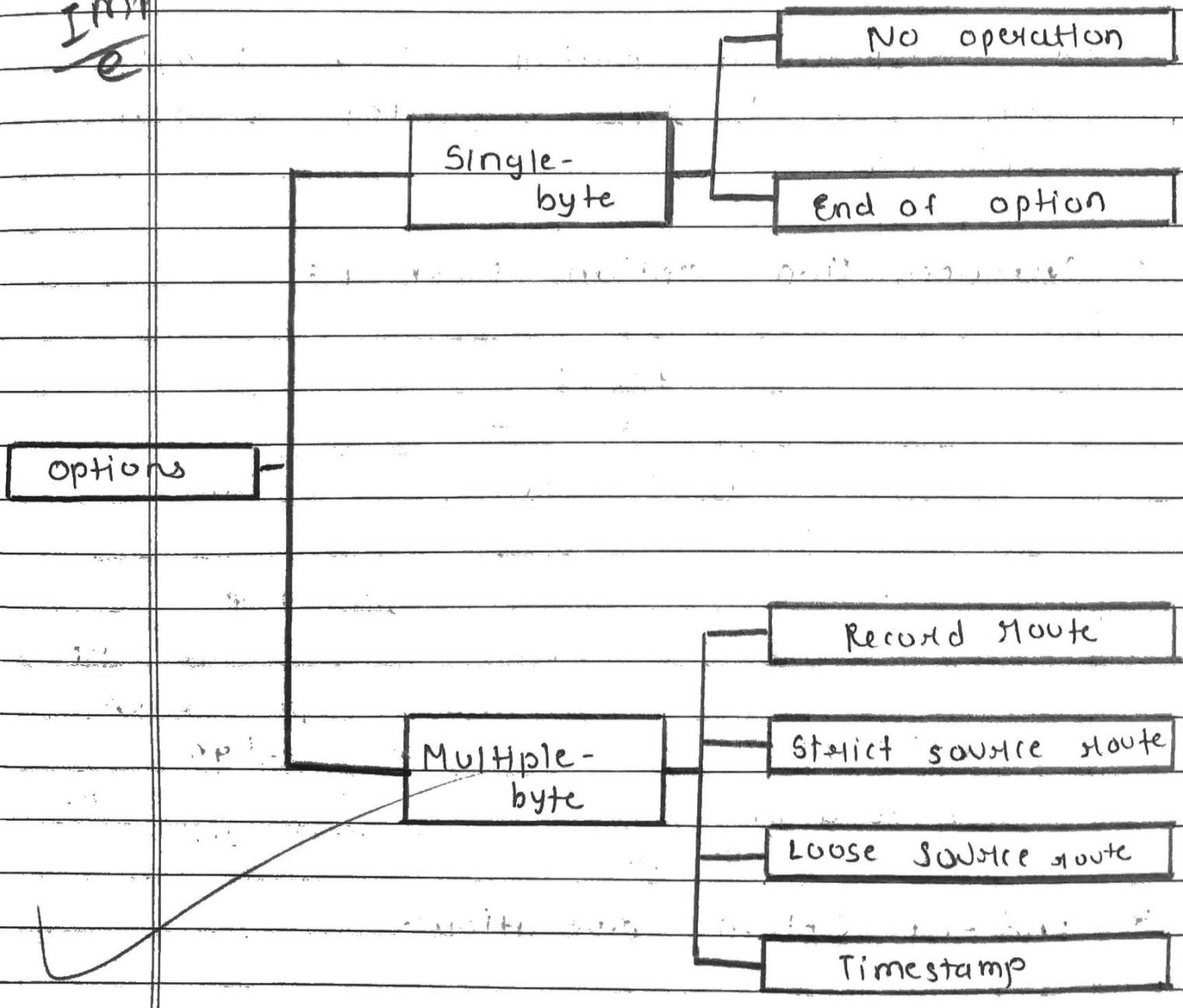
→ The type field is 8 bits long and contains three subfields : copy, class and number.

- copy. This 1-bit subfield controls the presence of the option in fragmentation
  - 0 copy only in first fragment
  - 1 copy into all fragments
- class. This 2-bit subfield defines the general purpose of the option.
  - 00 Datagram control
  - 01 Reserved
  - 10 Debugging and management
  - 11 Reserved
- Number. This 5-bit subfield defines the type of option.

Page
Date

- categories of option:

Imp



* No-operation option :-

→ A No-operation option is a 1-byte option used as a filler between options.

• No-operation option format :

Type : 1
00000001

a. No operation option

NO-OP
An 11-byte option

b. Used to align beginning of an option

A 7-byte option	
	NO-OP
An 8-byte option	

c. Used to align the next option

* End-of-Option operation :-

→ An end-of-option is also a 1-byte option used for padding at the end of the option field.

Page
Date

Type : 0 00000000

a. End of Option

Options END-OP
Data

b. Used for padding

* Record - route option :-

→ A Record - route option is used to record the internet routers that handle the datagram. It can list up to nine router IP address.

* Strict - Source - Route Option :-

→ A strict - source - route option is used by the source to predetermine a route for the datagram as it travels through the Internet.

→ Dictation of a route by the source can be useful for several purposes.

→ The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput.

* Loose - source - Route option :-

→ A loose - source - route option is similar to the strict source route, but it is more relaxed.

→ Each router in the list must be visited, but the datagram can visit other routers as well.

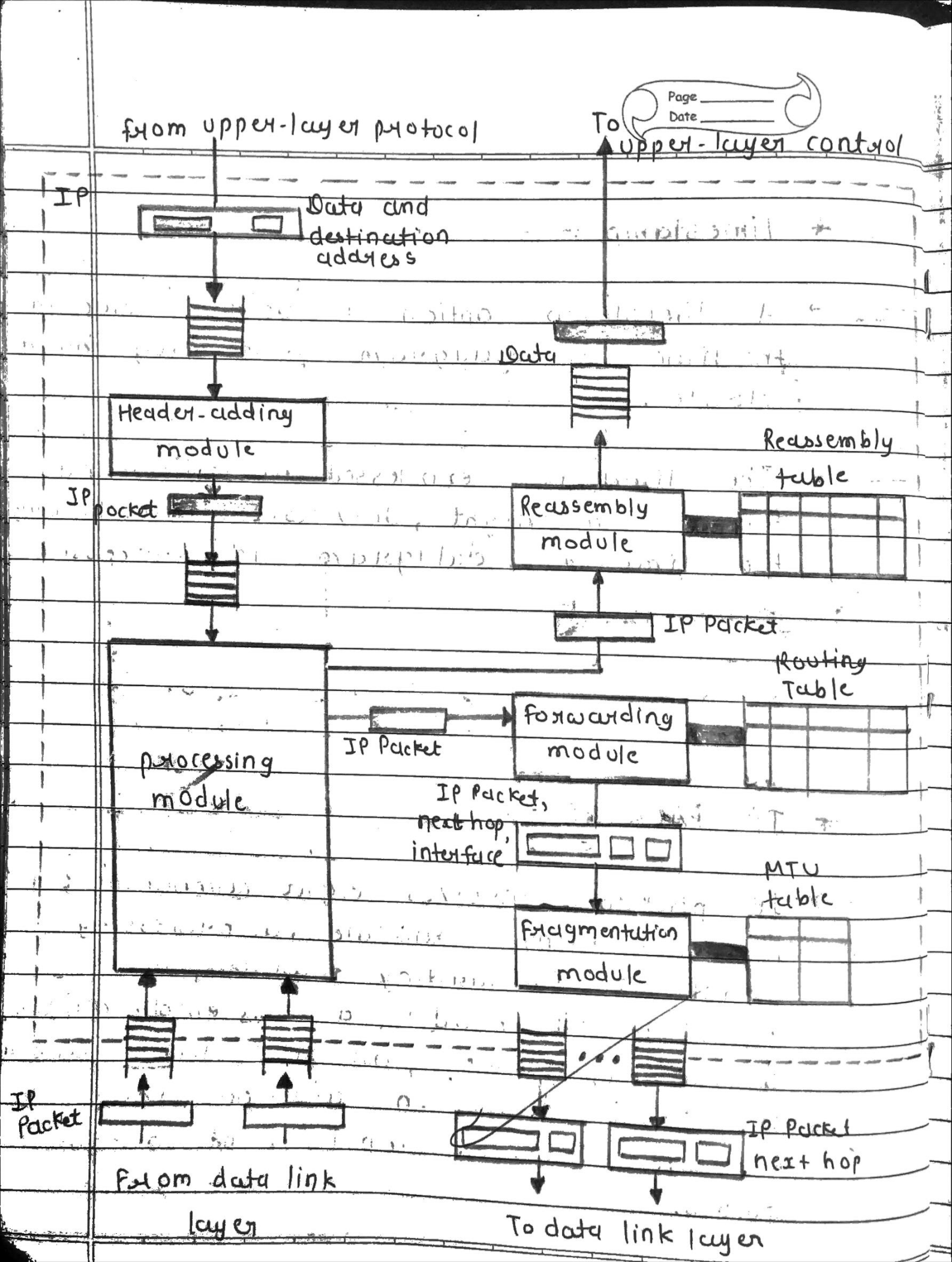
* Timestamp :-

→ A timestamp option is used to record the time of datagram processing by a router.

→ The time is expressed in milliseconds from midnight, Universal Time. Knowing the time a datagram is processed

* IP PACKAGE :-

→ IP package involves eight components; a header-adding module, a processing module, a forwarding module, a fragmentation module, a reassembly module, a routing table, an MTU table, and a reassembly table. In addition, the package includes input and output queues.


Page
Date
From upper-layer protocol
To upper-layer control
IP
Data and destination address
Data
Header-adding module
Reassembly table
IP Packet
Reassembly module
IP Packet
Routing Table
processing module
IP Packet
forwarding module
IP Packet, next hop, interface
MTU table
Fragmentation module
IP Packet
IP Packet next hop
From data link layer
To data link layer

* Header - Adding Module:

```
1  IP_Adding_Module (data, destination_address)
2  {
3      Encapsulate data in an IP datagram
4      Calculate checksum and insert it in
5                          the checksum field
6      Send data to the corresponding queue
7      Return
8  }
```

* Processing Module:

```
1  IP_Processing_Module (Datagram)
2  {
3    Remove one datagram from one of input queus.
4    If (destination address matches a local address)
5    {
6      send the datagram to the reassembly module
7      Return.
8    }
9    If (machine is a router)
10   {
11     Decrement TTL.
```

```
12  }
13  IFC TTL less than or equal to zero)
14  {
15      Discard the datagram
16      Send an ICMP error message
17      Return.
18  }
19  Send the datagram to the forwarding module
20  Return
21  }
```

* Queues :-

→ Our package uses two types of queues input ~~ques~~ ~~quess~~ queues and output queues.

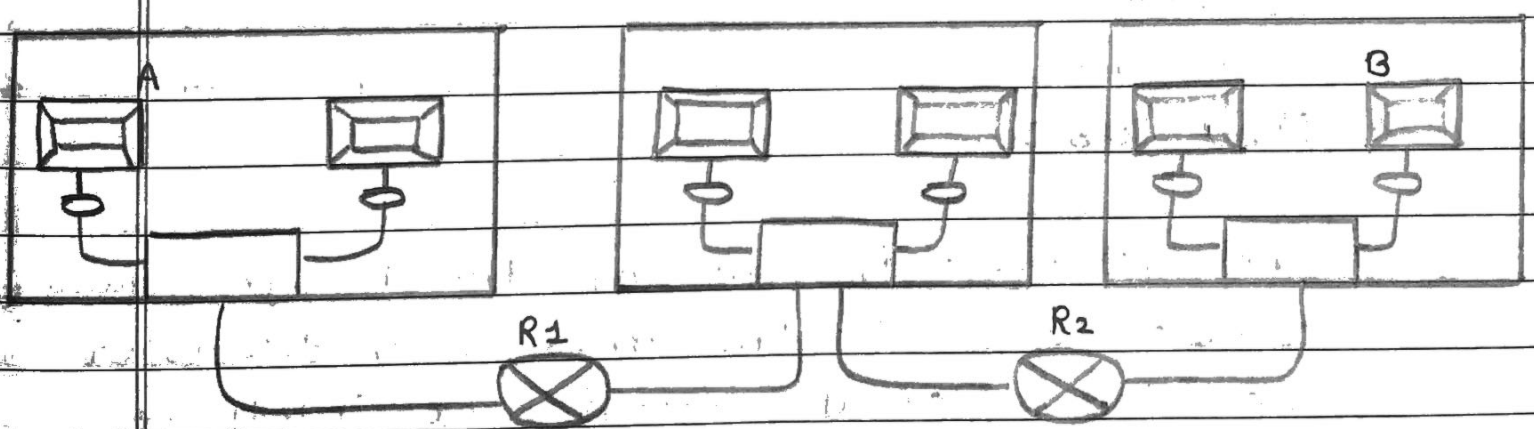
→ The input queues store the datagrams coming from the data link layer or the upper-layer protocols.

→ The output queues store the datagrams

going to the data link layer or the upper layer protocols.

* Routing Table :-

→ The routing table is used by the forwarding module to determine the next-hop address of the packet.



A

Destination	Route
Host B	R1, R2, Host B

R1

Destination	Route
Host B	R2, Host B

R2

Destination	Route
Host B	Host B.

a. Routing tables based on route

A

Destination	Next Hop
Host B	R1

Destination	next hop
Host B	R2

Destination	next hop
Host B	-

b. Routing tables based on next hop

* Forwarding Module :-

→ The forwarding module receives an IP packet from the processing module. The module finds the IP address of the next station along with their interface number to which the packet with this information to the fragmentation module.

* MTU Table :-

→ The MTU table is used by the fragmentation module to find the maximum transfer unit (MTU) of a particular interface. It can have only two columns : interface and MTU.

Interface no.	MTU
m1	1500
m2	500
x	5000

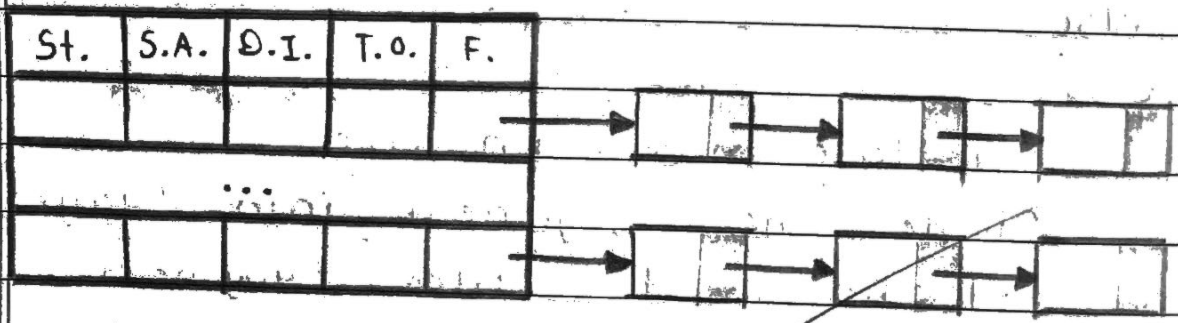
* Fragmentation Module :-

```
1  IP_fragmentation_module (datagram)
2  {
3    Extract the size of datagram
4    If (size > MTU of the corresponding network)
5    {
6      IF (D bit is set)
7      {
8        Discard datagram
9        Send an ICMP error message
10       return
11     }
12     else
13     {
14       calculate maximum size
15       Divide the segement into fragment
16       Add header to each fragment
17       Add regulered options to each fragment
18       Send fragmend
19
20       return
21     }
22   }
23   else
```

```
24  {
25      Send the datagram
26  }
27  Return
28  }
```

* **Reassembly Table :-** The reassembly table is used by the reassembly module.

St. : state
S.A. : Source address
D.I. : Datagram ID
T.O. : Time-out
F. : Fragments




```
1   IP- Reassembly_module (datagram)
2   {
3     IF (offset value = 0 AND M = 0)
4     {
5       Send datagram to the appropriate queue
6       queue
```

```
7  }
8  Search the reassembly table for the
9    ~~entry~~                          entry
10   IF (entry not found)
11 {
12     (reate a new entry
13 }
14   Insert datagram into the linked list
15   IF call fragments have arrived)
16 {
17   Reassemble the fragment
18   Deliever the fragment to upper-layer
19                                protocol
20   return
21 }
22 else
23 {
24     IF (time-out expired)
25   {
26     Discard all fragments
27     send an ICMP error message
28   }
29 }
30   Return
31 }
```

* **Source port number :** This is the port number used by the process running on the source host.

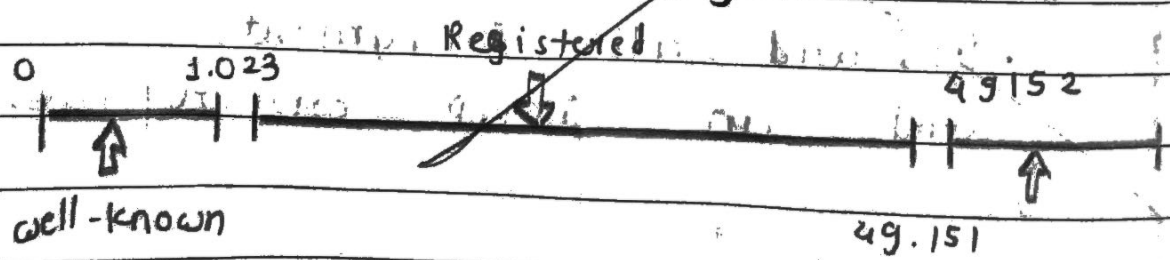
* **Destination port number :** This is the port number used by the process running on the destination host.

* **Length :** This is a 16-bit field that defines the total length of the user datagram, header plus data.

UDP length = IP length - IP header's length

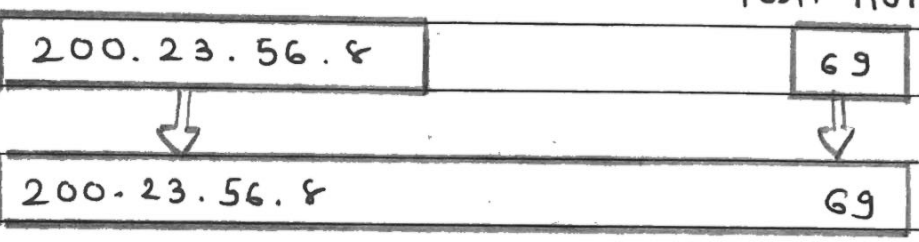
* **Checksum :-** This field is used to detect errors over the entire user datagram.

* **port numbers and range :-**



ICANN = Internet corporation for assigned names and numbers.

* **Socket address** :- the combination of IP address and port address is known as socket address.



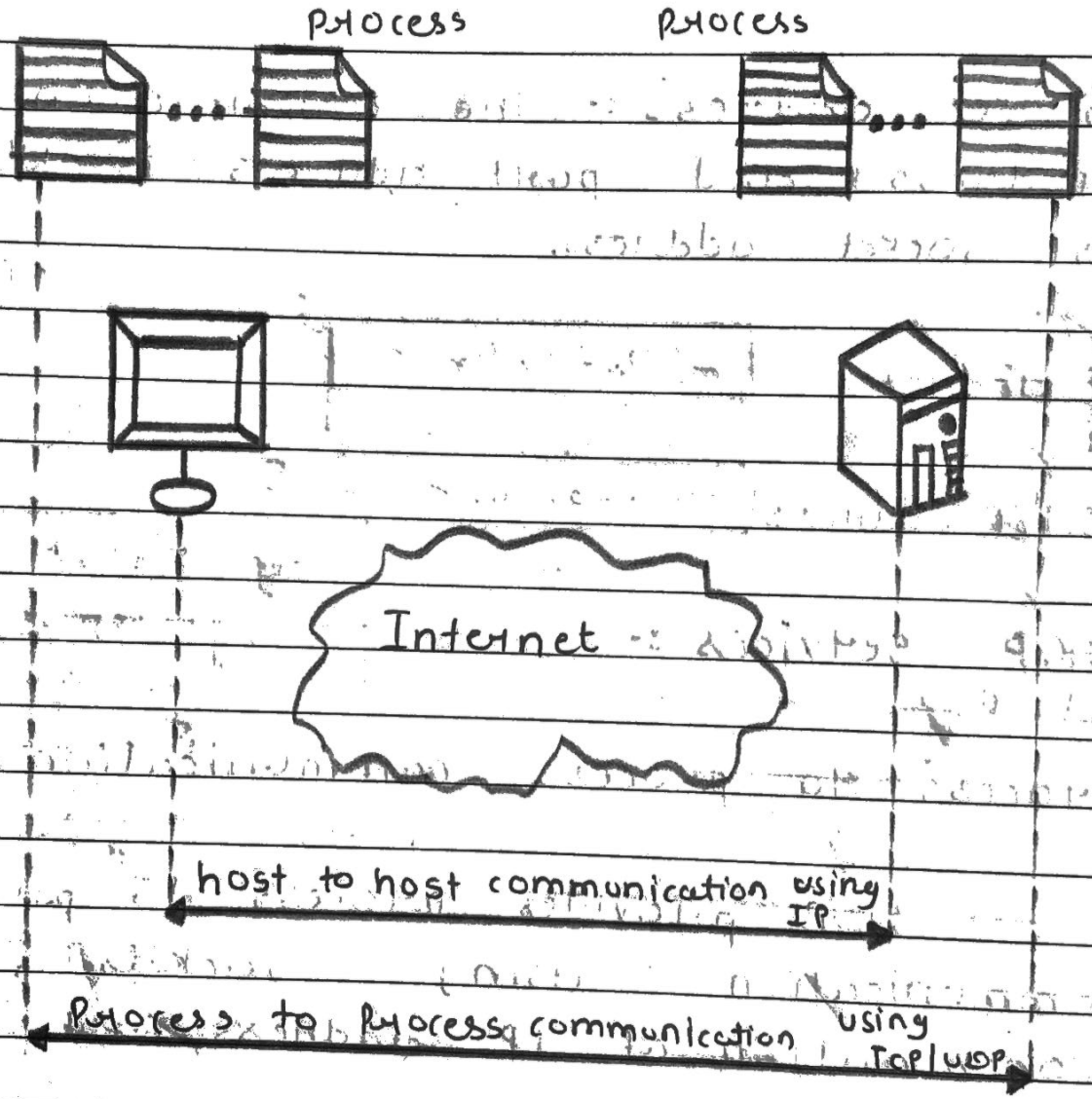
* **UDP services** :-

IMP(c)

- process to process communication :

UDP provides process to process communication using sockets, a combination of IP address and port number.

port	protocol	Description
67	Bootps	Server port to download bootstrap information.
68	Bootpc	Client port to download bootstrap information.
69	TFTP	Trivial file Transfer protocol
162	SNMP	Simple network management protocol (trap)



* Conectionless services :-

→ UDP is a conectionless 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

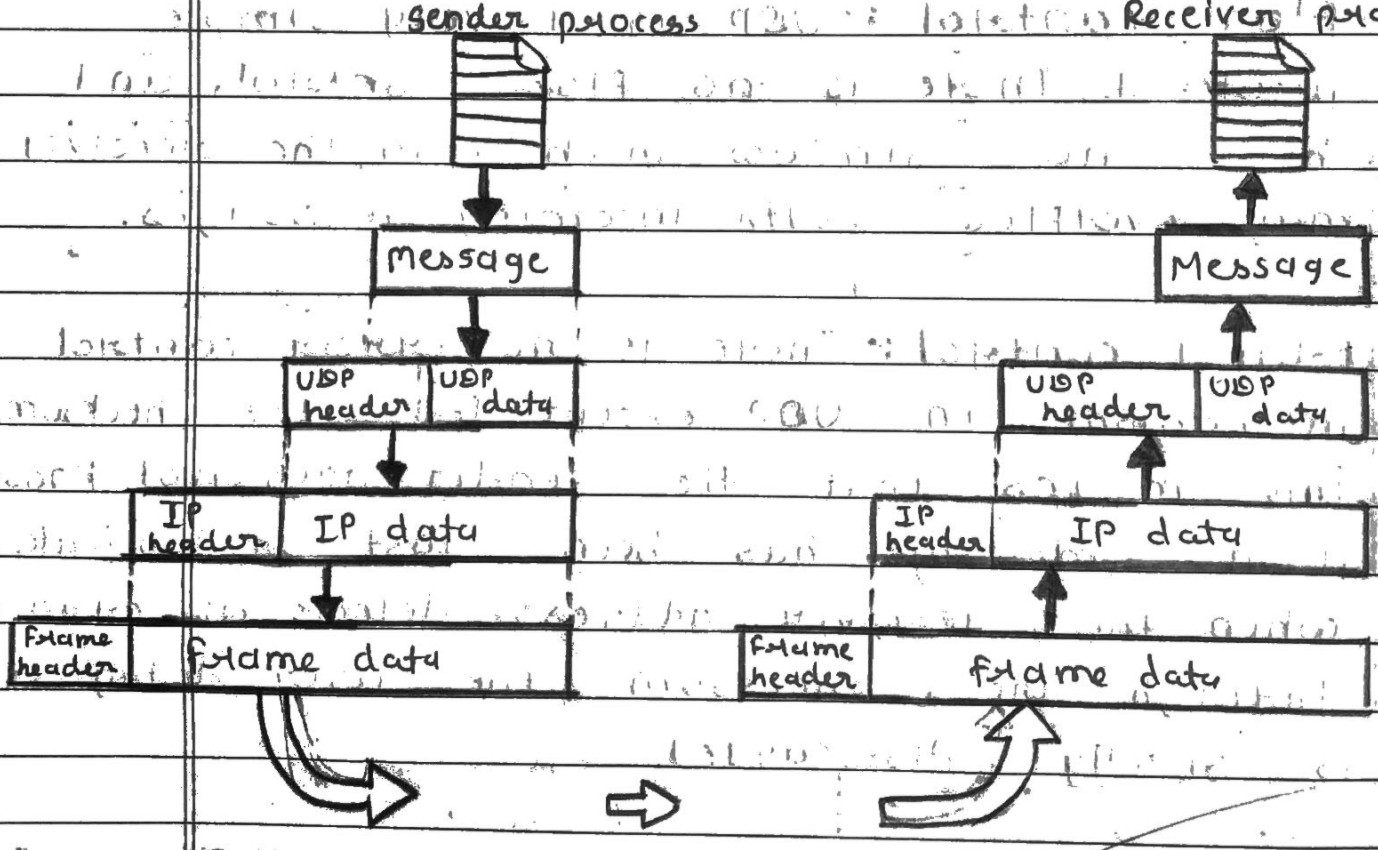
same source process and going to the same destination program.

* **Flow control :-** UDP is a very simple protocol. There is no flow control, and hence no window mechanism. The receiver may overflow with incoming messages.

* **Error control :-** There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver address detects an error through the checksum, the user datagram is silently dis-carded.

* **Congestion control :-** Since UDP is a connectionless protocol, it does not provide congestion control. UDP assumes that the packets sent are small and sporadic, and cannot create congestion in network. This assumption may or may not be true today when UDP is used for real-time transfer of audio and video.

* **Encapsulation:-** To send a message from one process to another, the UDP protocol encapsulate and decapsulates messages.



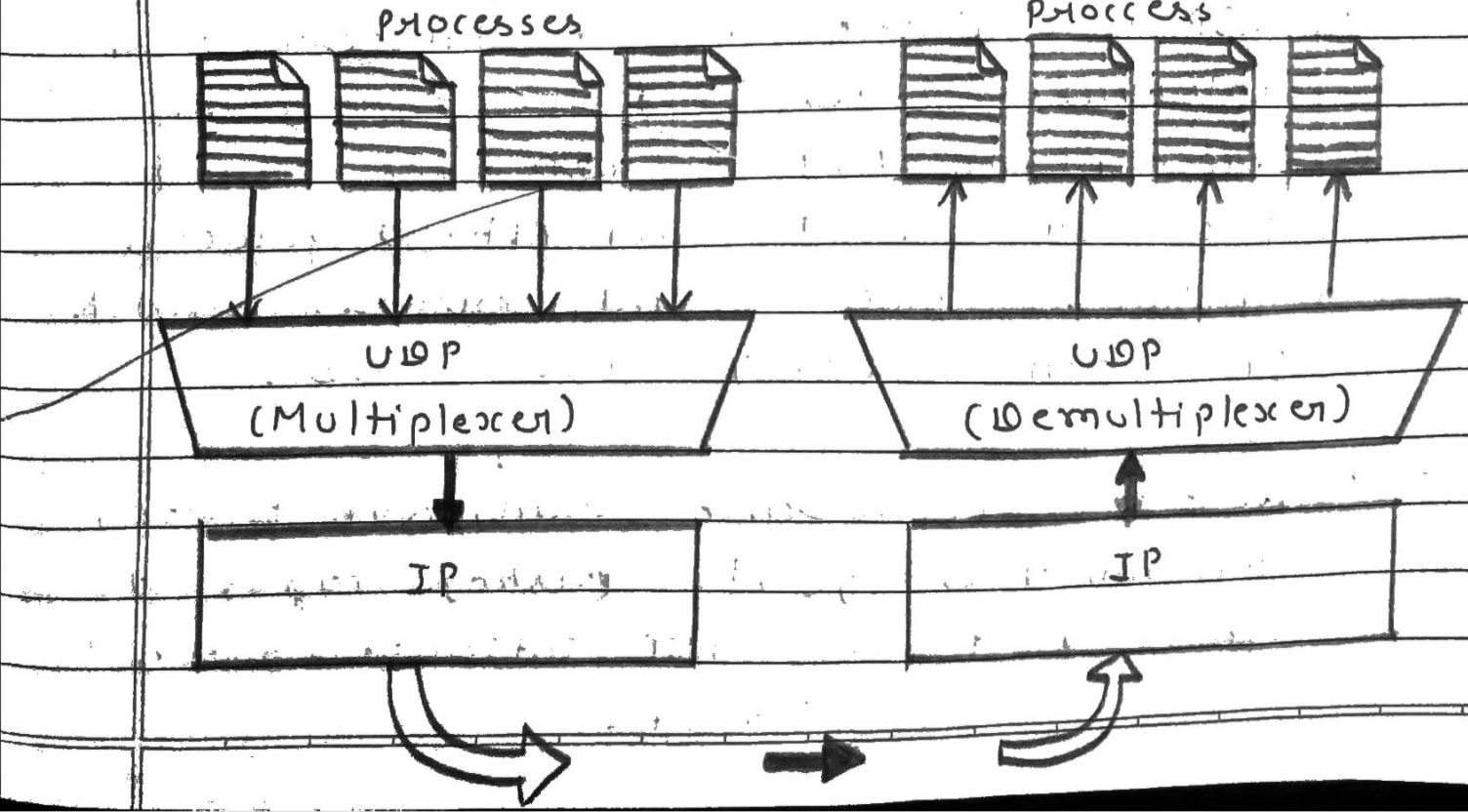
a. Encapsulation b. Decapsulation

* **Decapsulation:-** When the message arrives at the destination host, the physical layer decodes the signals into bits and passes it to the data link layer.

* **Queuing :-** At the client site, when a process starts, it requests a port number from the operating system. Some implementations create both an incoming and an out going queue associated with each process.

* **Multiplexing and demultiplexing :-**

→ In a host running a TCP/IP protocol suite, there is only UDP but possibly several processes that may want to use the service of UDP to handle this situation, UDP multiplexes and demultiplexes.



* Typical Applications :-

The following shows some typical applications that can benefit more from the services of UDP than from those of TCP.

→ UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk of data.

→ UDP is suitable for a process with internal flow and error-control mechanisms. For example, the Trivial file transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.

→ UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.

→ UDP is used for management processes such as SNMP.

→ UDP is used for some route updating protocols such as Routing Information protocol.

→ UDP is normally used for real-time applications that cannot tolerate uneven delay between sections of a received message.

* Difference between UDP and TCP.

ZMP

UDP	TCP
• connection less.	connection oriented.
• unreadible	Relible
• Not secure	Secure
• Small header size	bit header size
• ~~Fast for bound casting~~	Used

UDP	TCP
Used for broad casting	Used for unicasting
fast data transfer	slower data transfer
minimum overhead	more overhead
no error control	provides error control
no flow control	flow control
no Acknowledgement	provides acknowledgement
used for real-time applications	used for critical applications
such as audio and video transfer	such as ftp (file transfer protocol)
process to process communication	process to process communication
Transport layer protocol	Transport layer protocol