3.2 Multiplexing and Demultiplexing: day / date:

A process can have one or more sockets, doors through which data passes from network to the process & vice versa.

The T-Layer in the recieving host deliver data to an intermediary socket, because at the given time there can be more than one socket in recv. host.
(Each socket has a unique identifier.)

HOW SEGMENT IS DIRECTED TO APPROPIATE SOCKET?

Each T-layer segment has a set of fields in the segments for this purpose. T-Layer examines these fields to identify the recieving socket then directs the segment.

"The job of delivering data in the T-Layer segment to the correct socket is called Demultiplexing."

"The job of gathering data chunks at the source host from diff sockets, encapsulating each data chunk with header info to create segments & and passing the segments to the network layer is called Multiplexing"

T-layer multiplexing requires,
i- that the sockets have unique identifiers.
ii- that each segment have special fields that indicate the socket for segment delivery.

Special Fields:
i- Source port No. field.
ii- Dest. port No. field.

day / date:

Each port number is a 16 bit number, ranging from 0 – 65535.
port number ranging ~~for~~ from 0 – 1023.
well Known port Nos. – These are Restricted.

HTTP – 80
FTP – 21

means that they are reserved for use by well Known app protocols. (HTTP-80)

Each developed application must have a port No. assigned to Socket.

- Connectionless Multiplexing & Demultiplexing. (UDP.)

2 Segments with same dest. Add. is directed to the same ~~port~~. Sockets.

In Server Side, the developer would have to assign the corresponding well Known port no.
In client Side, the ~~server side~~ T-Layer automatically assigns the port number.

- Connection Oriented Multiplexing & Demultiplexing. (TCP.)

TCP Socket is identified by 4-tuple.
UDP Socket is identified by 2-tuple.
(IP Address, Dest. port number)

(Source Ip, Source Port No, Dest Ip, Dest Port No)

2 segments with diff Source IP & source port no. will be directed to two different sockets.

- With UDP, there's no handshaking b/w sending & recieving entities

Reason for the UDP to be connectionless.
(DNS uses UDP)

UDP is required by those real time systems/apps which often require a minimum sending rate & donot want to overly delay Segment transmission, as TCP uses congestion control & Keeps resending until acknowledged by dest.

day / date:

UDP better suited than TCP for following reasons:

1. Finer Application level control over what data is sent, and when.

2. No Connection establishment. (No. Handshaking.)

3. No connection state. (More Active clients.) (No Record.)

4. Small packet header overhead.
   - TCP has ~~4~~ 20 bytes of header overhead.
   - UDP has 8 bytes of header overhead.

PG. (230) [Table at PG 231]. Important.

3.3 : UDP (User Datagram Protocol):

3.3.1 UDP Segment Structure:

Application data occupies the data field of the UDP segment. For DNS, a query or response, for audio streaming, sample audio as data.

UDP header has 4 fields.
each field is 2 bytes long

- Length field specifies the no. of bytes in the UDP segment (header plus data.)

- Checksum to check whether error introduced, at recv. end.

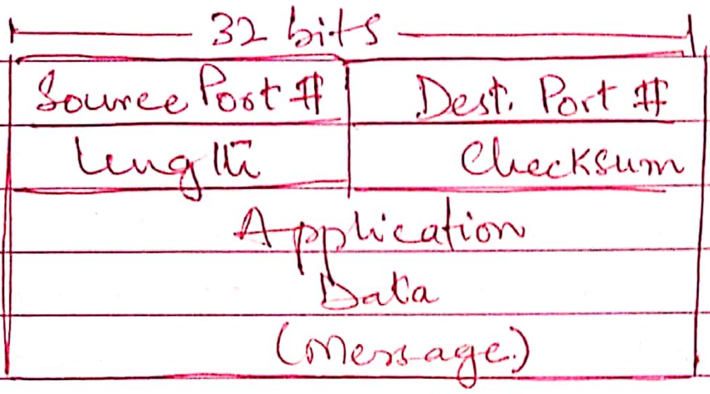
- port numbers.

day / date:

3.3.2. UDP CheckSum.

- provides for error detection.

checks whether the bits within the UDP segment have been altered.



- UDP Segment Structure.

Udp at the sender side performs 1st complement of the sum of all the 16 bit words in the segment. (with any overflow)

↓

The result is then put in the checksum field. of UDP

```
      1 1 1 1 1 1     1 1 1 1
      1011101110110101
    + 1000111100001100
    1 0100101011000001
                    +1
    ------------------
      0100101011000010
1st   1011010100111101
Comp.
```

day / date:

Adding all the 16 bits words including checksum,
If 16 bit 1s is the result, then it means no error.
If any one bit is 0, then error occured.