UDP Header

- **Transmission Control Protocol (TCP) :**
  - Is reliable while delivering an entire message
  - Is connection oriented
  - Is full duplex and point to point
  - Has 3 phases : Connection establishment , Data transfer , Connection termination
  - Each TCP connection is associated with four window :

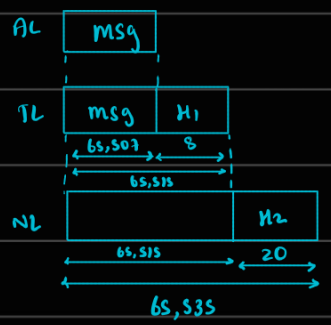
: Both sender & receiver has 2 windows : sending and receiving windows since it is full duplex

  - Uses 3 way handshake to establish a connection
  - Is NOT useful for multicasting and broadcasting
  - Provides end-to-end error control and flow control
  - Data CAN received out of order (in today's impl) but TCP NEVER delivers out of order to the process. TCP guarantees that data is delivered in order to the process

- **User Datagram Protocol (UDP) :**
  - Is a message-oriented connectionless datagram protocol
  - Is unreliable. Does not provide flow, error OR congestion control
  - Does not add anything to services except PROCESS TO PROCESS delivery of data
  - UDP header is simple and FIXED IN SIZE : 8 Bytes

Source Port (16 bits)	Destination Port (16 bits)
Total Length (16 bits)	Checksum (16 bits) → OPTIONAL

- **Total Length** (16 bits)
  - Max. no. = $2^{16} - 1$ = 65,535
  - Total Length = Data size + Header size
  - ⇒ max. data size = 65,527 → Theoretical maximum only



⇒ Practical Maximum : 65,515 − 8 = 65,507 Bytes

- TCP does not have such limit since TCP takes care of segmentation. So, AL can send arbitary data & TCP STREAMS it reliably
- In the case of UDP, we want to MINIMIZE SEGMENTATION since UDP is unreliable. IP fragments it but it is better to reduce that as well since there is no guarantee of reassembly (UDP is message-oriented)

- **Checksum** (16 bits)
  - IS OPTIONAL. If checksum is not computed, checksum field is ALL 0s.
  - Includes 3 sections: UDP header, UDP data, IP pseudo-header

IP pseudo-header

Source IP addr. (32 bits)		
Dest. IP addr (32 bits)		
Reserved (8 bits) 00000000	Protocol (8 bits)	UDP Total Length (16 bits)

WE DO NOT TRANSMIT THIS. IT IS REMOVED AFTER CHECKSUM CALCULATION

- Since the checksum is optional, there are 3 cases that can occur:
  - 1. Sender decides not to include checksum:

    Solution: Sender sets the checksum field as ALL 0s.
  - 2. Sender decides to includes the checksum but the value of sum is all 1s.

    In other words, checksum is all 0s:

    Solution: Sender take 1's complement again and sets the checksum field (to all 1s)
  - 3. Sender decides to includes the checksum but the value of sum is all 0s

    In other words, checksum is all 1s:

    Solution: This case is NOT POSSIBLE as they are fields like source IP addr., etc which can NEVER be 0 and this case implies that all the values in the sum are 0 and that is not the case

- Process is similar to that in TCP:
  - 1. Divide all 3 into 16-bit segments (since checksum is 16 bits)
  - 2. Add them all up using 1's complement addition (last carry wrap around)
  - 3. Compute 1's complement of the result

Eg. ① UDP header: $(CB84\ 000D\ 001C\ 001C)_{16}$. Identify the UDP header fields and is this packet directed from client to server or vice versa?

$(CBB4)_{16} = (52,100)_{10}$ = Source port addr.

$(000D)_{16} = (13)_{10}$ = Dest. port addr. ⇒ Belongs to well known port nos. (0 to 1023) ⇒ This is a SERVER.

$(001C)_{16} = (28)_{10}$ = Total length ⇒ length of data = 28 − 8 = 20 Bytes

⇒ Packet is moving from client to server.

- **Why UDP :**

  - If an application requires : (and / or)
    - One request one response
    - Constant dataflow
    - multimedia data transfer
    - Fastness and then reliability
  - Used in real-time interactive apps (to do something within a timeframe)
  - used in management protocols such as SNMP (simple network management protocol)
  - used in route updating protocol such as RIP (routing information protocol)
  - used for broadcasting and multi-casting
  - Is suitable for internal flow and error control mechanisms. For eg. the trivial file transfer protocol (TFTP) uses UDP for internal flow and error control

NOTE : A client-server application such as SMTP, which is used in email, CAN NOT use the services of UDP because a user might send a long message which could include multi-media. If the app uses UDP and the message does not fit in one user datagram, the message must be split into multiple datagrams. Here, the connectionless service may create problems since there is NO seq. no. field in UDP (so the receiver might receive out of order.