In a TCP client - server

If one node is waiting on recv/read

Other has initiated close request, read/recv returns zero

Now break the loop, close the socket

MTU - Max Transmission Unit

TCP Window Size

```
pthread_create(pt1, NULL, sender, (void*) csd );
```

UDP Sender - Receiver

TCP client - server, multiple messages

Wireshark (sudo apt install wireshark)

HLEN : 0101 ==> 5x4 -- 20 bytes

TCP Header : min 20 bytes ,
max 60 bytes (HLEN : 5 to 15)

Flag bits:-
- URG
- ACK
- PSH
- RST
- SYN
- FIN

SYN - connection estbalishment
ACK - acknowledgement
RST - Reset/Reject
PSH - Last segment
FIN - closing the connection
URG - urgent pointer (priority)

Window size -- mutually acceptable payload size

---

server --> socket, bind, listen ==> passive open
client --> socket, bind ==> active open

ISN -- initial sequence no., e.g. 9000
ack no = seq no + 1 for SYN

* Socket States - CLOSED, LISTEN, SYN-SENT, SYN_RECVD,
ESTABLISHED, FIN_WAIT1, FIN-WAIT2, CLOSE-WAIT, TIME-WAIT,
LAST-ACK, CLOSING

ack no = seq no + data length + 1 for data transmission

connection establishment (SYN)
data transfer (PSH, ACK)
connection termination (FIN)

Concurrent server using fork:-

```
socket                  socket              socket
bind                bind                    bind
listen                                      listen

accept                  connect             //register SIGINT handler

      send                    recv          while(1) {
      recv              send                      csd=accept(....)
                                                  ret=fork();
close(csd);             close(csd);               if(ret==0) {
                                                      close(ssd);
close(ssd)                                            //talk to client
                                                      close(csd);
                                                      exit(0);
                                                  }
                                                  else
                                                      close(csd);
                                            }

                                            SIGINT handler -- close(ssd)
```

loop

```
socket
bind
listen

//SIGINT handler to stop the server, by closing ssd

while(1) {
        csd = accept(.....);
        pthread_create(pt1, NULL, talk_to_client, (void*)csd);
}

Hint:- detachable threads instead of joinable threads

void* talk_to_client(void* pv) {
        int csd = (int)pv;
        //comm using csd
        close(csd);
}
```

select, poll