Simple client utils:-

telnet -- TCP

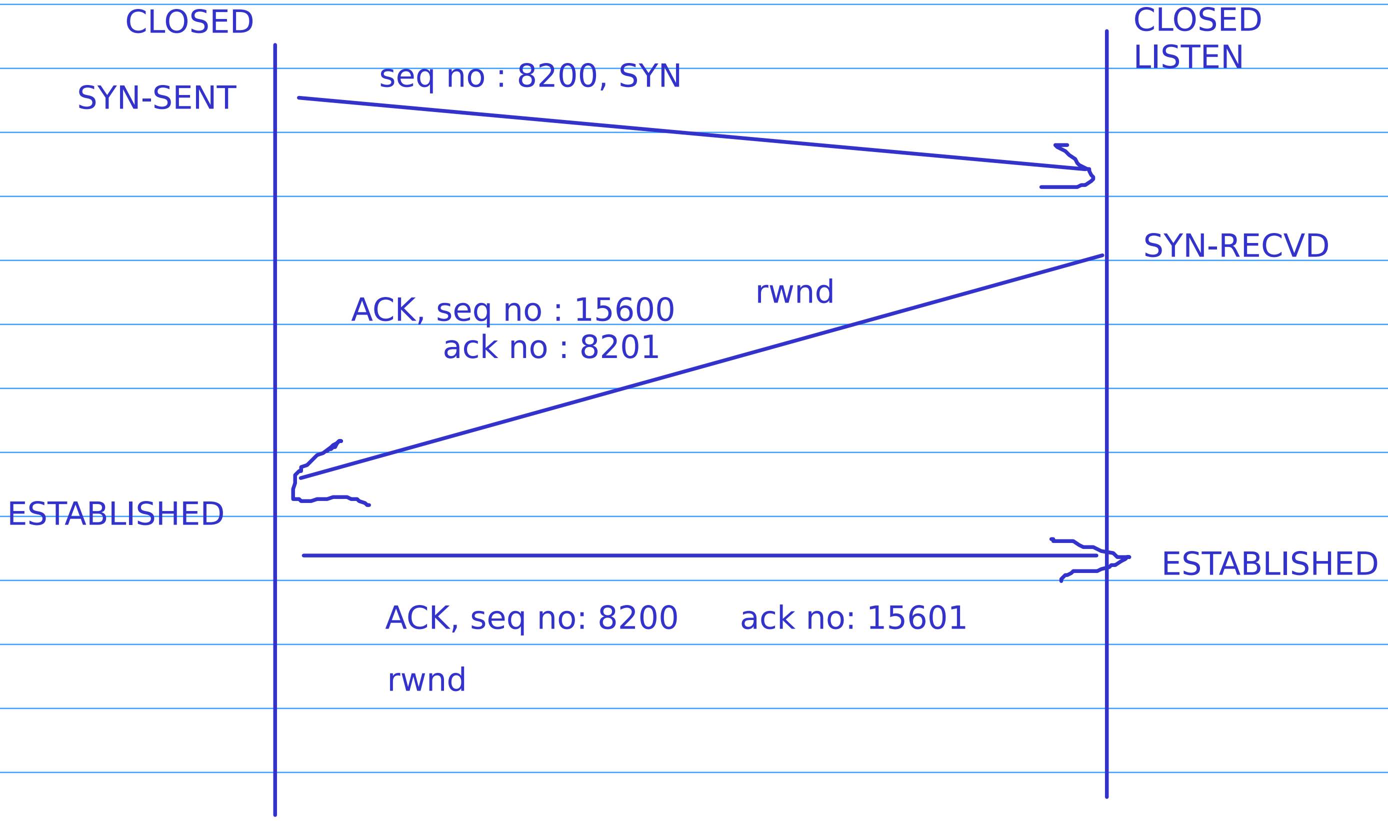
nc -- TCP & UDP

-----------------------------------

Fragmented

Debugging
Bootloader
Device Tree
Interrupts & Timers

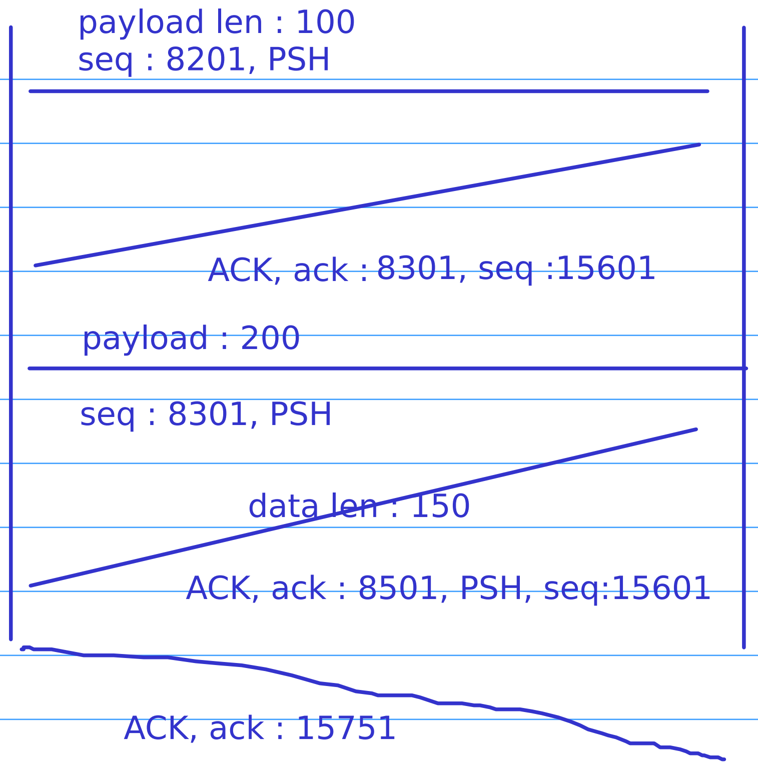
Do PSH flag set if app data is not fragmented, i.e. single segment?



ISN - Initial Sequence Number

ack no : seq no + data len + 1

netstat
-t
-u
-l
-a



ack : seq no + data len

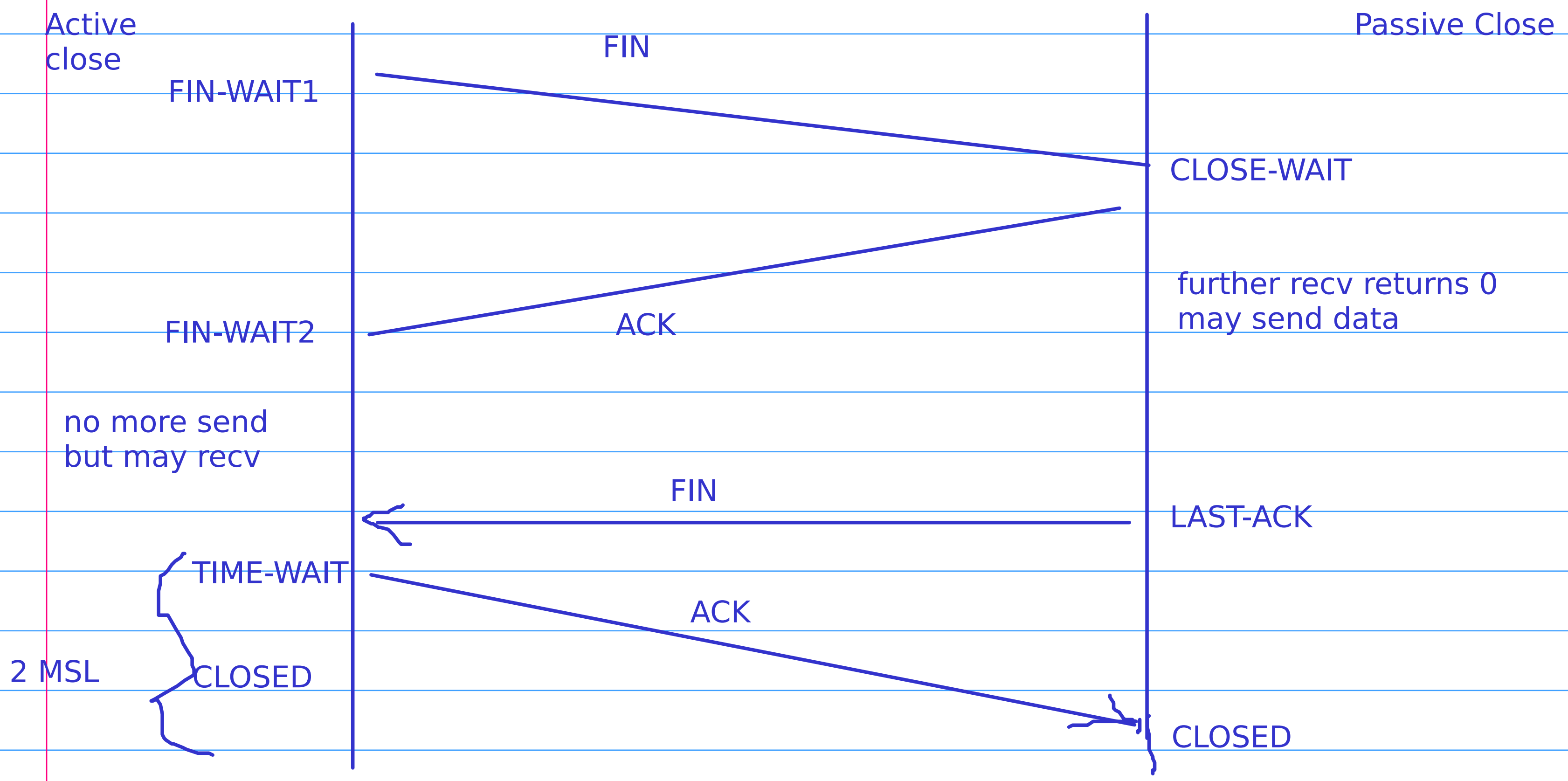
ACK segment may be comined
with data segment in other
direction (piggy back, delayed ACK)

There can be single acknowledgement
for multiple segments

---

## Connection Termination

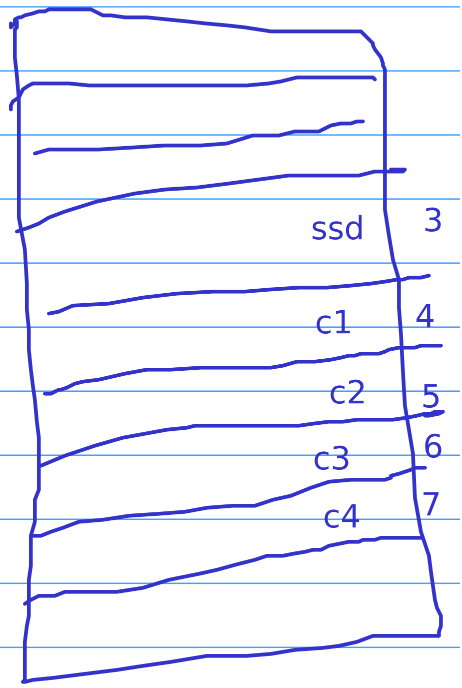
MSL - Max Segment



I/O Multiplexing:-
- select
- poll
- epoll

---

Prep:-
   * socket
   * bind
   * listen



```
fd_set active_fd_set, read_fd_set;
FD_ZERO(&active_fd_set);
FD_SET(ssd,&active_fd_set);

maxfd = ssd;
while(1) {
  read_fd_set=active_fd_set;
  ret=select(maxfd + 1,&read_fd_set,NULL,NULL,NULL);
  for(i=0; i<maxfd + 1 ;++i)  {
     if(FD_ISSET(i,&read_fd_set))  {
     if(i==ssd) {
        csd=accept(ssd, (struct sockaddr*)&caddr, &len);
        FD_SET(csd,&active_fd_set);
        if(csd>maxfd) maxfd=csd; //update maxfd if required
        //print client ip and port based on caddr
     }
     else     //already connected clients (sending data or FIN) {
        csd = i;
        nbytes=recv(csd,buf,maxlen,0);
        if(nbytes==0) {
                close(csd); FD_CLR(i,&active_fd_set);
        }
        //process the data, based on bux, maxlen              (normal data transfer)
   }}
}
close(ssd);
```

## Unix Domain Sockets (Unix Local) - Meant for IPC

Server:-

Step1:-
```
ssd = socket(AF_UNIX, SOCK_STREAM, 0);
```

Step 2:-
```
struct sockaddr_un addr;
addr.sun_family = AF_UNIX;
strcpy(addr.sun_path, "some path");                    //e.g. "sample"
bind(ssd, (struct sockaddr*)&addr, sizeof(addr);
```

Step 3:-
```
listen(ssd, 5)
```

Step 4:-
```
csd = cl = accept(ssd, NULL, NULL)
```

Step5:-
```
//Communcite usin csd .. read/write system calls
```

Step6:-            close(csd);

Client:-

Step1:-
```
csd = socket(AF_UNIX, SOCK_STREAM, 0);
```

Step 2:-
```
struct sockaddr_un addr;
addr.sun_family = AF_UNIX;
strcpy(addr.sun_path, "some path");              //path to recognize server

connect(csd, (struct sockaddr*)&addr, sizeof(addr));      //connect to server
```

Step 3:-
communicate using csd

Step 4:-
```
close(csd);
```

Recap
Pseudo
Drivers

---------------------------------------

Tasks:-
* Socket Programming Assignments - pending, select, unix local
* Book reading , LKD & other (interrupts, bottom halves, timers)
* Learning Report (if any mail reply)
* Any pending hands-on
* Any other tasks over mail

Optional:-
netlink
sockets

| | |
|---|---|
| Peripherals:- | UART, GPIO |
| Next:- | I2C / SPI |

---------------------