Network

→ Network appn programming is all about writing appn program using network API's and Network protocol stack.

→ This program communicate with program either on the same machine or other on diffn machine on remote location.

Computer N/w.

WAN (wide..)

↳ CSN (circuit switched n/w)

↳ A dedicated physical path is established b/w two points through several nodes.

↳ The path is connection of sequence of physical links b/w the nodes. (cable & telephone)

↳ If sender wants to transmit the packet, it will pass through this dedicated lines

PSN

→ A packet switch network uses N/w protocol that divides data in to smaller packets called n/w packets

→ Now n/w packets transmitted to wards the n/w (distant format 0, 1) & Wi-Fi.

LAN (local..)

↳ works on broadcast approach

↳ No intermediate switches.

↳ Because of which the data rate of LAN is much greater than WAN.

n/w devices

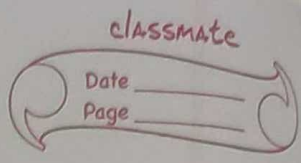
Router & Switch
→ Networking devices to connect one or more devices to a computer

Hub
Hub is used to connect a device in a LAN.

bridge
Bridge is n/w device that connect LANs

Server

7/4/2022



Sender Receiver

FTP/https/DNS/SMTP

Application.

Presentation
① Adding format for packet to send to receiver
② Encryption decryption
③ Compress packet to reduce size

Establish connection b/w sender & Rec Session

Transport

Network

data link

Physical

* Session
→ Maintain session b/w sender and receiver until receiver gets data
→ Apart from establishment & synchronizing terminates; while transmitting packet from session layer, because of some events session layer stops transmitting In next cycle session layer did not start of from beginning, It continues transmitting point of interruption.

* Transport

→ Receive data from session, and divides it to in smaller parts.
→ Two main protocol
1) TCP 2) UDP

* Network
takes the data from transport layer and converts into network packets.
→ Network layer has IP address of src and dest, So all the routing process starts from network layer.
→ Will decide which path in data should be transmitted to the destination.
→ Then all routing takes place in data link layer

* Data link layer:-
→ Creates frames, checking for the data error free
→ If there is an error removes the error and transfer forward.
→ It also maintains data rate speed b/w sender and receiver. (15MBPS - 10MBPS)
→ After some time traffic may increase on receiving side, receiver may loose the packet
→ It will maintain a common data rates b/w send & rec.
→ data link layer also generates physical address

* physical:-
→ format corrupted and converted into 0's and 1's
→ at physical event devices how, convert the data into electrical signals and radio waves

* Ifconfig:-
- Interface Configuration
- can display active n/w interfaces of system
→ Ifconfig is used to view and change, network interface config. of system

* mtu:-
maximum transmit unit
- max size of packet in a transmission

* txqueuelen:-
max no. of packets - 1000
- Interface of linux kernel that deals with network devices.

* lo (loopback address):-
- each device has same lo address
127.0.0.1 (data in a loop in a device itself)
- Called as local host
- When we send the data using loop-back address, data never reaches the address. data is in a loop, n/w card interface itself
→ loop back address is used for testing TCP/IP internal flow path

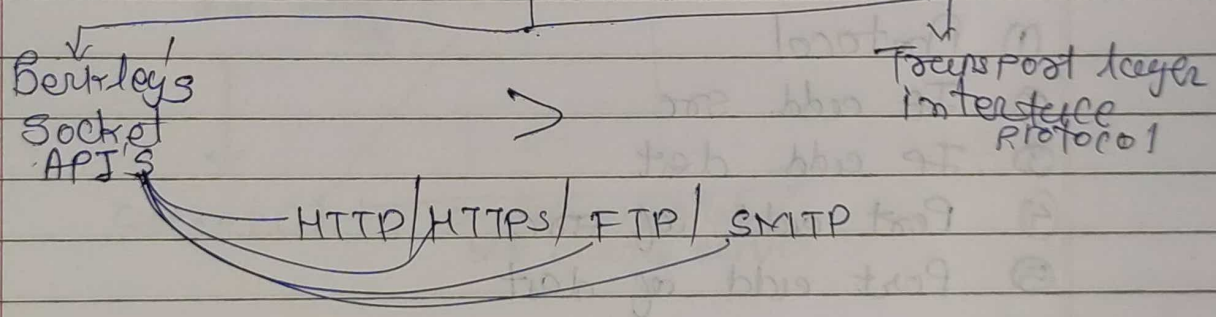
→ loop back add will help to device to send and receive packets of its own

ifconfig wlan up()
{
(driver) open()
{
Initialization allocation
}
}

ifconfig wlan down()

Networking Programming

Choice of API's



* Socket System Calls

→ socket system call supports many protocols. we can say that socket system call designed to support network comm protocol.

→ Because of the same region socket system call parameters are generic in nature.

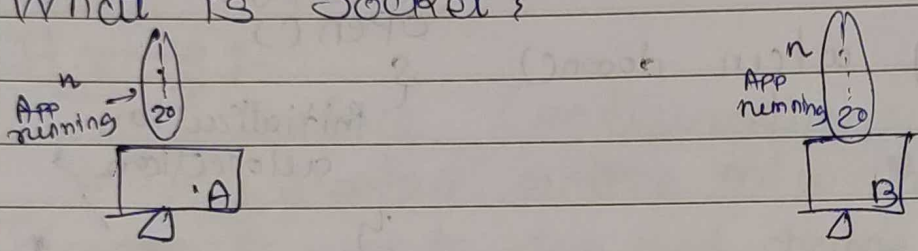
→ As you are using generic socket system call the system call will use same socket structures as a argument and .are socket structures

struct sockaddr

(socket system call)

→ S.S.C also takes the size of as parameter this call identify the size of socket stru

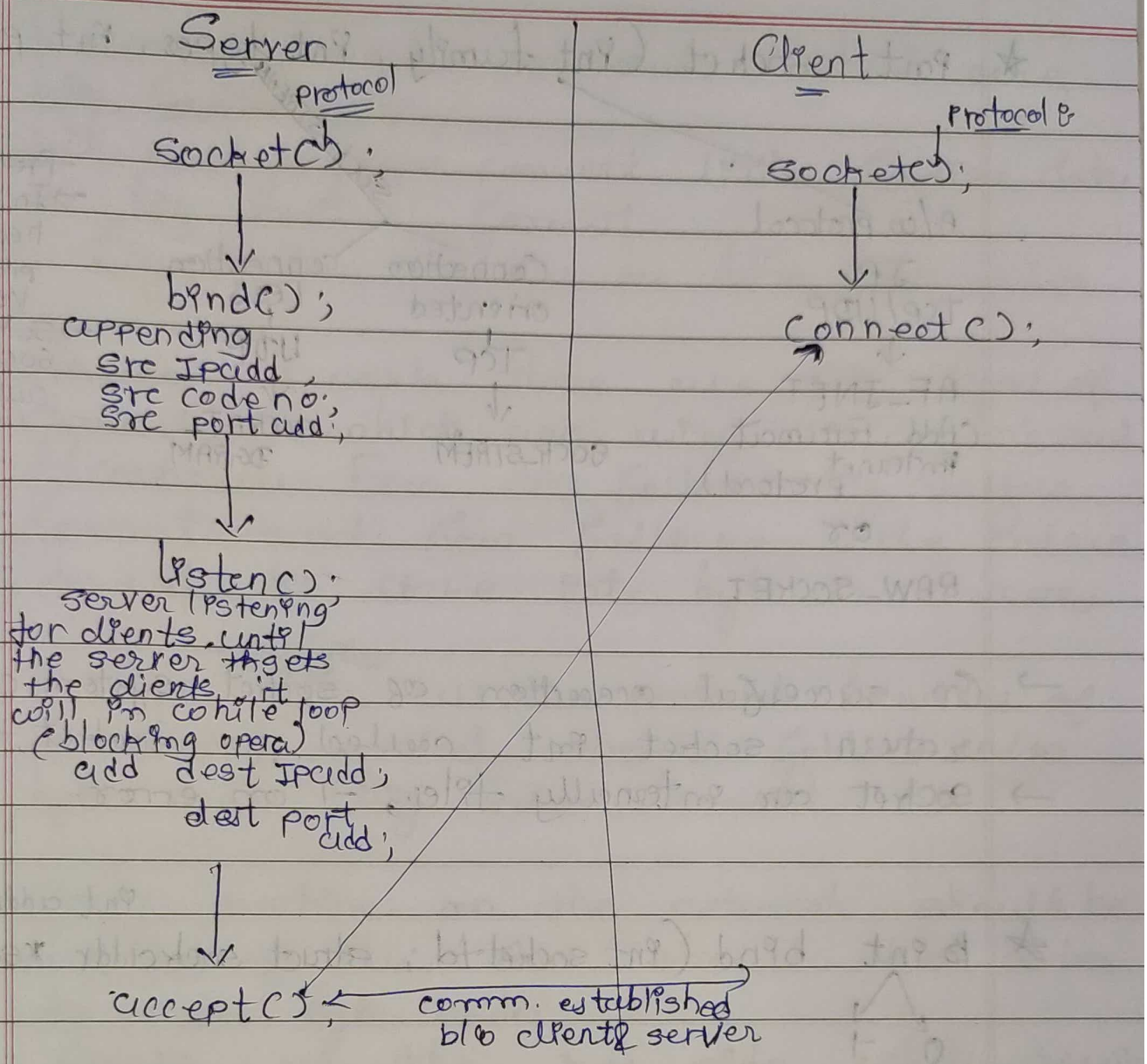
* What is Socket?



→ Socket is end point or end link in any netw comm. clients servers
→ We needs to sockets, senders, receivers

* Socket Parameters:-
① Protocol
② IP add. src
③ IP add dest
④ Port add of src
⑤ Port add of dest

→ N/w cpp programming is all about implementing client and server model programs.
→ HTTP client program connecting to http server program Eg Google chrome
→ client pop connect client server pop
→ Jobs in networking field
→ write routing protocols
→ customizing WANs
→ Doing LANs



→ Once the connection establishes, comm starts.

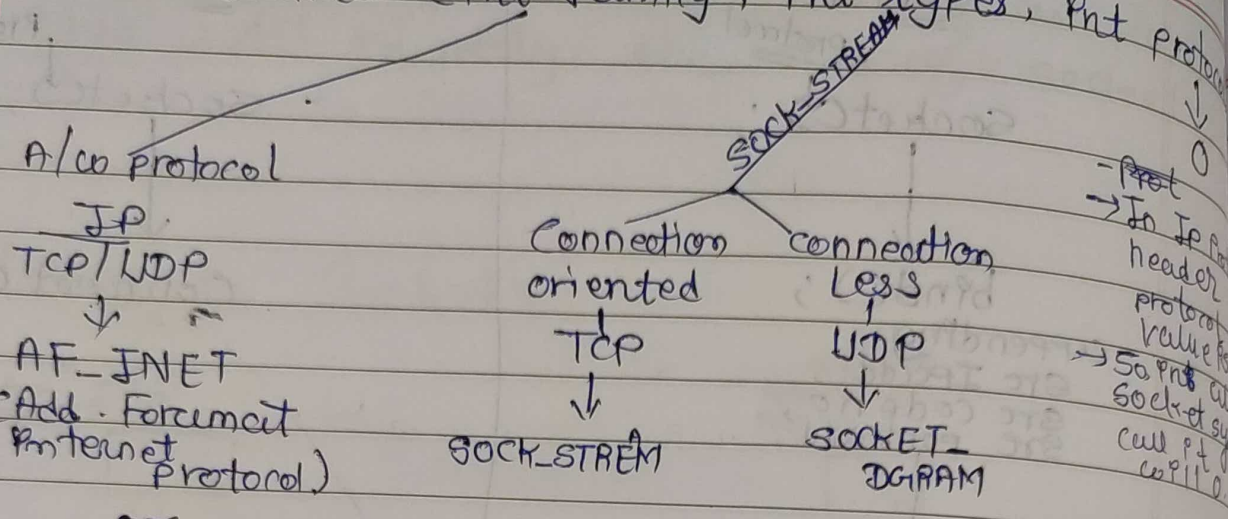
→

```
#include<sys/types.h>
#include <socket.h>
```

* Struct sock addr (socket structure)

```
{
  short sin_family;   n/w protocol
  Ushort sin_port;    16 bit Port Number (n/w byte order)
Struct in_addr sin_addr;   32 bit IP addr (n/w byte order)
};
```

★ int socket (int family, int types, int protocol);



→ On successful creation of socket system call return socket int called as socket fd
→ socket are internally files., -1 on error.

★ int bind (int socket-fd; struct sockaddr *server, int addrlen)

0 -1

★ int listen (int socket fd, int Queuelen);
↳ No. of clients connection

★ int accept (int sock fd, struct sockaddr *cli, int *addcli);
size

★ connect (int sock fd, struct sockaddr *ser, int addrlen);

big Endian

a	12
a+1	34
a+2	56
a+3	78

0x12345678

little Endian

a	78
a+1	56
a+2	34
a+3	12

classmate Date Page

→ Network works on data which are big Endian.

→ We need to ~~con~~ convert little Endian data to big Endian format.
big Endian known as N/w byte order

→ In a network there are diff^n types of computer which are using diff^n CPU's and Mcontrollers. Few CPU follows big Endian format and few follows little Endian format to store int. byte in memory.

→ TCP/IP specifies all the protocol headers of a network packet should follow n/w byte order (Big Endian format).

→ Every machine on the network should be aware of this. And should host order to n/w byte order

→ Again at the rec side while receiving incoming packets n/w byte order host byte order

→ So source code will use the API's to convert host to n/w and n/w to host order. to match respective CPU requirement.

→ This API's will work for both little and big Endian format.

☆
ulong htonl (long hostlong);
takes 32 bit host addr IP Addr
convert it to 32bit n/w byte order

☆
ushort htons (ushort hostshort);
takes 16 bit port addr to convert it to 16 bit n/w byte order

☆ ntohl ( );

☆ ntohs ( );

* Address Conversion API's :-

1) ulong inet_addr (char str);
dotted decimal IP addr)
"127.0.0.1"
↓
Converting to 32-bit n/w order

2) char* inet_ntoa (unsigned n/w long);

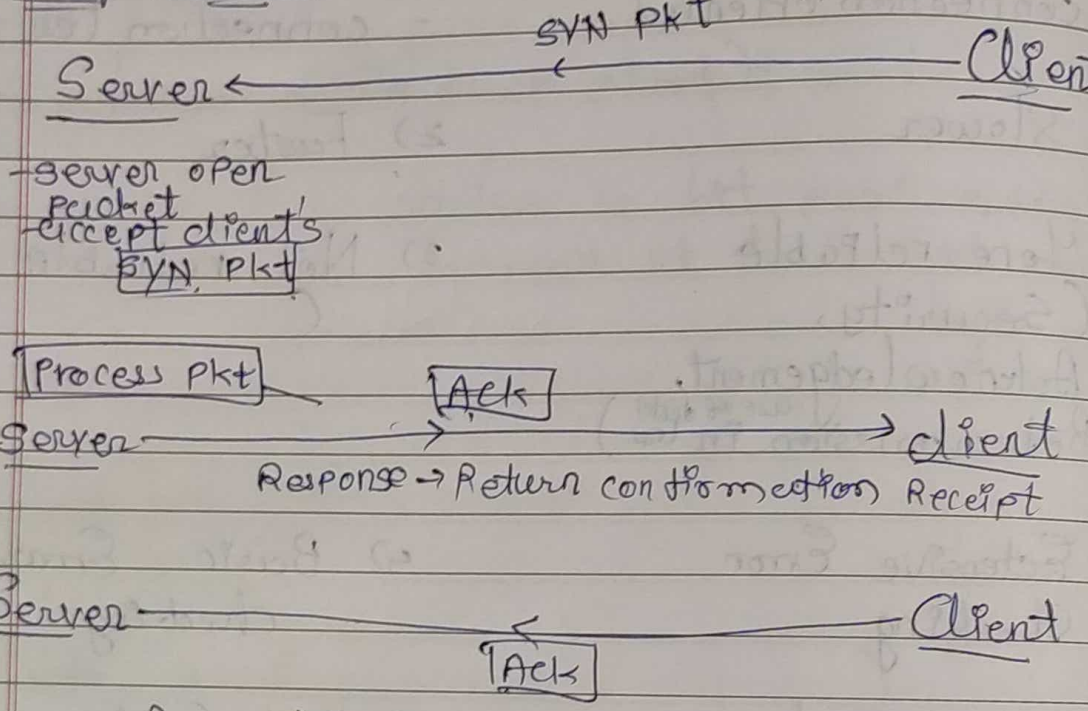
3) inet_pton(family, src *, dest *);
struct sockaddr

TCP	UDP
1) Connection establishment - connection oriented	1) No connection establishment - connection less
2) Slower	2) Faster
3) More reliable (Security, Acknowledgement, Retransmission (case of data loss in n/w))	3) Not reliable (C
4) Extensive Error Checking	4) Basic Error checking
5) streaming byte transfer	5) Unit (data packet) transmission
6) Header 20 - 60 bytes	6) 8 bytes fix
7) Broadcasting No supports	7) Supports broadcasting
8) Supports data Sequencing (into packets (arrived in order)	8) No data sequence (but App in code we can manage)
9) HTTP/HTTPS/POP/FTP	9) DNS/DHSP/TFTP
10) Guarantee on data delivery	11) do not give guarantee

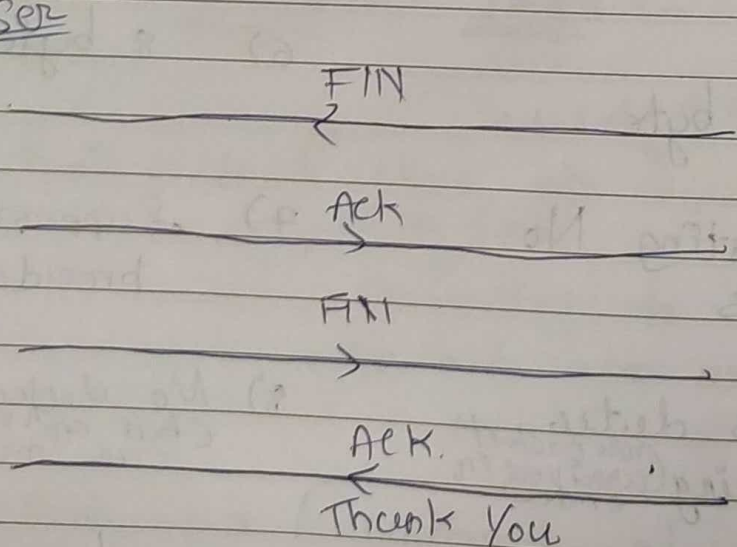
TCP

→ for connection estu. 3- way handshaking
→ for conn. close 4- way hand shaking

3 Way hand Shaking



4 Way hand Shaking



* accept ( ) ;

called by TCP server, shall extract the first client request on the queue and creates a new socket of the same socket type which is specified (Address family of the newly created socket is also of same specified type) and returns a socket file descriptor.

Scenario: Implementing fork(): in servers to handle a client request.

Data transfer API's

* ssize_t recvfrom (int sockfd,
  char *buf,
  int length,
  int flags, (0)
  struct sock addr * addr,
  int addrlen);

flags

MSG_CONNTERM

* ssize_t sendto (int sockfd, char *buf,
  int length, int flags,
  struct sock addr *addr,
  int addr(len);

* RAW sockets:
sends Raw packets to App specific user. Now the user will get untampered final network temple. Now this packet is bypassing all TCP IP processing.
→ Optimizing and customizing WAN Network
→ If you want to improve the performance and security on critical traffic workloads on WANs.
→

11/4/2022

* Iterative server:-
Are one which connects / serves a single client at a time. The 2nd client has to wait till 1st client terminates

```
n= accept( )
        ← a new client socket created of same type
while(1)
{
  n= read( );   ← Enter while loop, server read req data and process it.
  (n=0)         ← Now program will come out of inner while loop when client closes the connection.
}
```

→ Once client closes, server again executing accept() system call and waiting for new client.
→ Have a simple Interface
→ If client has small transaction duration, then clients are good.
→ but concurrent servers serves multiple clients simultaneously

TCP
- connection oriented
- streaming data

FIFO
- connection oriented
- streaming data

UDP
- connectionless
- Packet data
- Not reliable

MQueues
- connectionless
- whole unit
- reliable ↔ because tr through one system.

$ netstat -tulpn
t u l p n
tcp udp listen pid of proc no

→ netstat is an open source tool called as network statistics. Tells about how your computer is communicating with other computers and other communicating devices. provides detailed info. of individual n/w comm. Also protocol info. The info is used in trouble shooting and diagnosing purpose

→ A file called services in /etc folder displays all the services that are used by client on your computer system

→ Function getservent(), will read the line of service file in /etc and return a pointer to next entry in service file.

* data link layer:-

The structure will have to fetch protocol, physical data type, physical Address, Address length

*. struct sockaddr_ll
{
sll_family;
sll_protocol;
sll_pkt_type;
sll_hatype;
sll_addr[8];
sll_addrlen;
}

* Packet Interface at device level:-

→ PF_PACKET family used to create packet socket.
→ Data transmission is from physical link layer which from below protocol.
→ The type - SOCK_RAW, SOCK_DGRAM
→ The type is communication which form using raw packets and pass to user space.
→ It uses socket of the packet link layer pass to specific user

→ This protocol value defined by IEEE of pipe (800)

→

* int setsockopt (int sockfd,
int level,
int option_name,
int option_value,
size_t size of option value);

→ Used by app^n program to change the socket behaviour. We can change any option in the socket.

* Implementing terminal emulator program, while reading multiple I/O sources, If i configured process to read from keyboard when there is no data to read from it will blocked.
→ We can use multiple threads to overcome the issue, to read from multiple sources
→ Advance method, system call file descriptor select() system call().

→ select(); makes the programmer from multiple file descriptors, when file descriptors are ready for I/O operations.

int select (int nfds, fd_set *readfds, fd_set *writefds
fd_set* errorfds, struct time_val timeout);

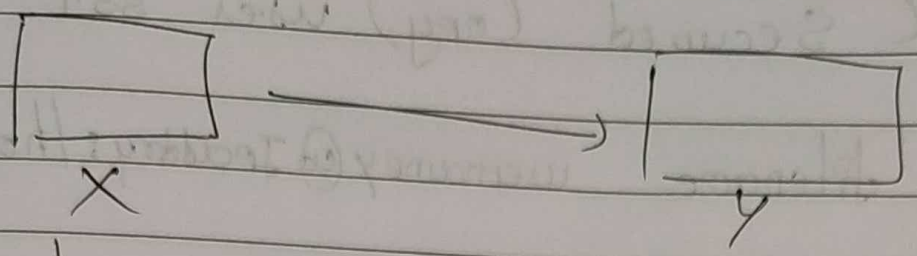
→ The first parameter in fds tell select(),
system call, the no. of fds to be checked.
→ The first pointer to type fd set is for
testing the readiness of fds for read
operation.
→ The second " " " " " " "
" " " " " " Write
operations.
→ If any of fds have any error pending
will be verified by this pointer.
→ timeouts is a variable of type
struct time_val structure.
→ This timeout will make select()
to block until fd is ready for
IO operation
→ Only timeout when interrupted by
signal.

→ No. of fds ⇒ set by either read fds
write fds
error fds

* FD_ISSET(), checking fd is set or not

* select system call can work for
an more no. of size which is less
than fd-setsize. which is 1024.

Open ssh client



$ ssh username Y @ IPaddr Y

Y Ps send public key to X, ask if you really cocent to establish connection

Y / N

X

file dot

./home/user → .ssh → Non host.

using open

→ ssh client ccen connect to remote machines cend controls the machine.

→ Open ssh is cen open source software that ps used to provide seecure cend encrypted comm. pn a computer networks using ssh protocols.

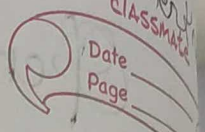
→ ssh protocol Ps esteblishing connection b/co ssh client cend server

→ Used by system administrator to acces a remote machine pn unsecured networks. ssh Ps network protocol cohich ps comes under TCP/IP protocol.

→ That uses post no. 22

→ ssh is used to connect two machines remotely over network and also to control the machine

→ scp is used to transferring the files



★ scp (Secured (copy) uses ssh

scp filename username@ IPaddress: /home/dhru

→ After establishing a connecting connect b/w client ssh and server ssh, ssh protocol is providing encryption and algorithms for ensuring the privacy and data security b/w cli/ server machine

→ No theft of hacking.

★ IOPERM → IO permission → Only one system call from U.S. which can access the data port and command port from device