Network Programming Unit 1

A naag lok exclusive, Baki units jaise jaise karta waise bana ke send karta lavdo, patience rakho

No sharing Beyond Naag lok, since i am feeling okish today unlike most days, I'll allow that useless noisy to be part of naag lok notion bonanza.

The questions marked low and very low can be skipped since each unit has choice.

Again noisy is allowed. But seriously please just die man

1. **BSD Network architecutre - very low**

   BSD stands for Berkerly software distribution. It is an Unix operating system derivative developed and distributed by University of California

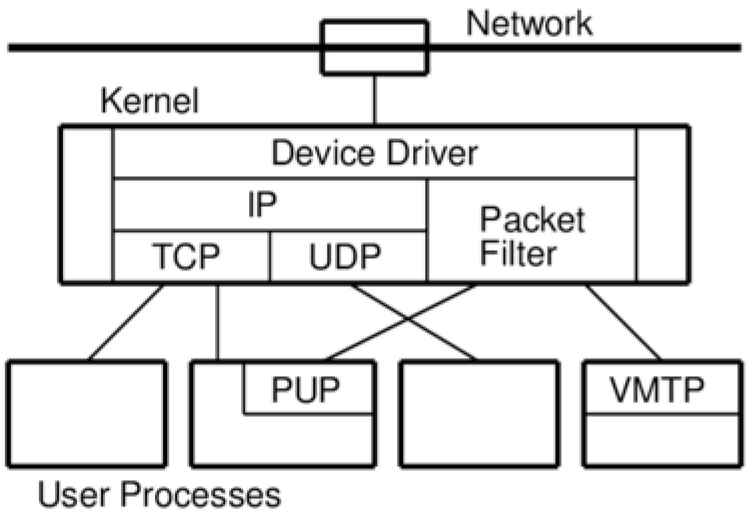
   #The original BSD architecture came out in 1983 and it was the first widely avalaible release of TCP/IP and sockets API

   #The 2nd version consisted of TCP performance improvements

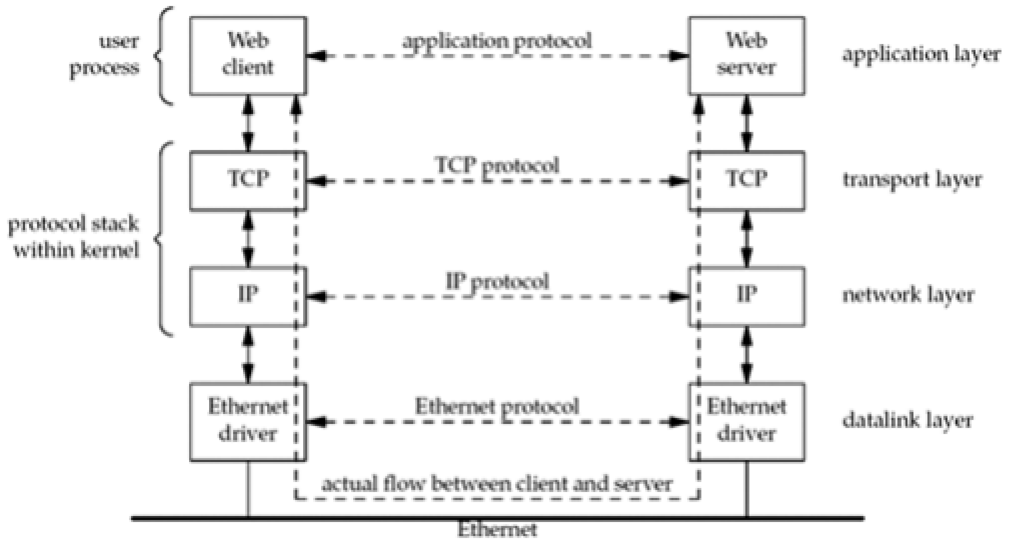
   #The 3rd iteration of BSD consisted of slow start, congestion avoidance and fast retransmit

   #The next iteration worked on fast recovery, TCP header prediction etc.

   #The Final version in 1993 brought out multicasting, long fat pipe modifications.



2. communication over LAN

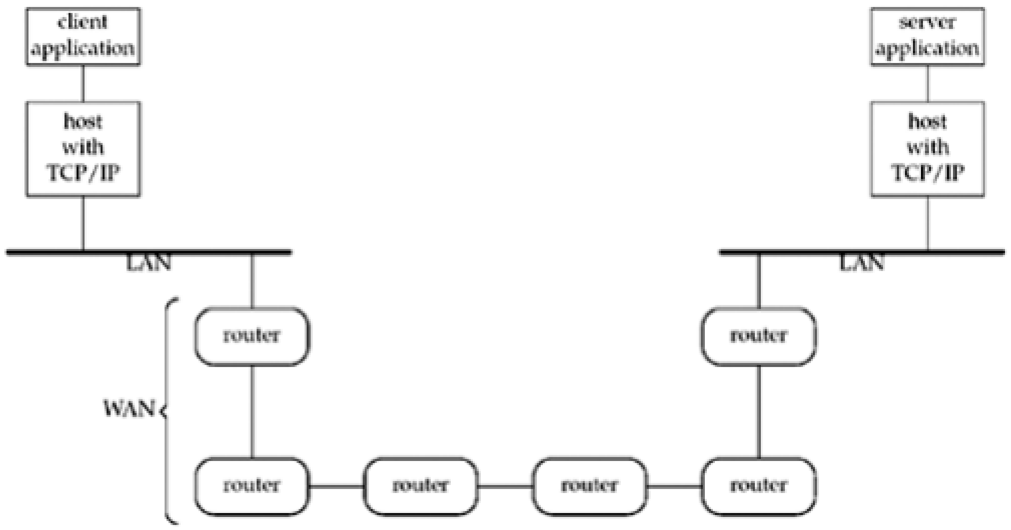


Even though the client and server communicate using an application protocol, the transport layers communicate using TCP.
The actual flow of information between client and server goes down the protocol stack on one side, across the network, and up the protocol stack on the other side.
Client & Server are typically user processes, while TCP and IP protocols are normally part of the protocol stack within the kernel.

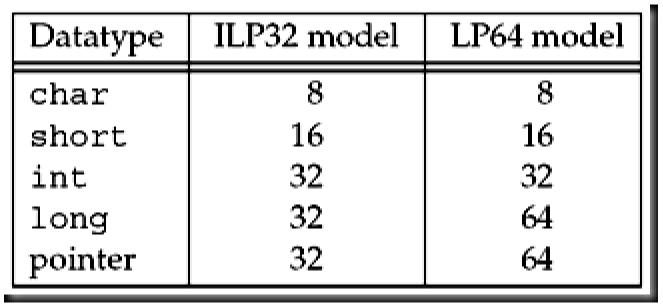
The four layers labeled in the diagram are the Application layer, Transport layer, Network layer, Data-link layer.
Some clients and servers use the User Datagram Protocol (UDP) instead of TCP.

3. Communication over WAN

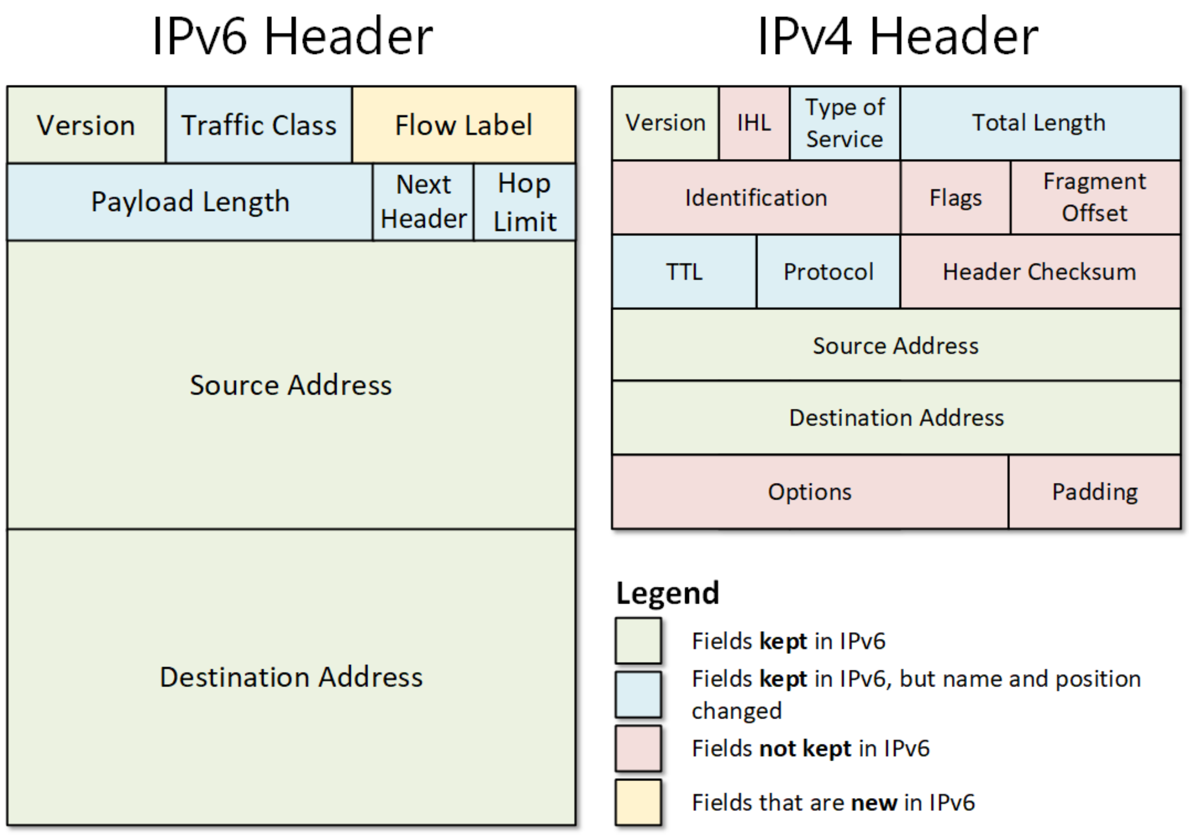
The client & server on different LANs is connected to a Wide Area Network (WAN) via a router.
Routers are the building blocks of WANs. The largest WAN today is the Internet. Many companies build their own WANs and these private WANs may or may not be connected to the Internet.



4. 64 bit architectures - very very low



Datatype	ILP32 model	LP64 model
char	8	8
short	16	16
int	32	32
long	32	64
pointer	32	64



5. Day time client - high

int main(int argc, char **argv)

{

int sockfd, n = 0;

char recvline[1000 + 1];

struct sockaddr_in servaddr;

int port = 13;

```
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
  err(1, "Socket Error");
}

bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(port);
inet_pton(AF_INET,argv[1],&servAddress.sin_addr);
if (connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr)) < 0) {
  err(1, "Connect Error");
}
while ((n = read(sockfd, recvline, 1000)) > 0) {
      recvline[n] = 0;
      fputs(recvline, stdout);
  }

  return 0;
}
```

6. Day time Server - Low

```
int main(int argc, char **argv)
{
int listenfd, connfd;
int port = atoi(argv[1]);
struct sockaddr_in servaddr;
char buff[1000];
time_t ticks;
listenfd = socket(AF_INET, SOCK_STREAM, 0);
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(port);
bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
listen(listenfd, 8);
for (;;) {
connfd = accept(listenfd, (struct sockaddr *) NULL, NULL);
ticks = time(NULL);
snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
write(connfd, buff, strlen(buff));
close(connfd);
}
}
```

7. SCTP state transition - very high

   The transitions from one state to another in the state machine are dictated by the rules of SCTP, based on the current state and the chunk received in that state.

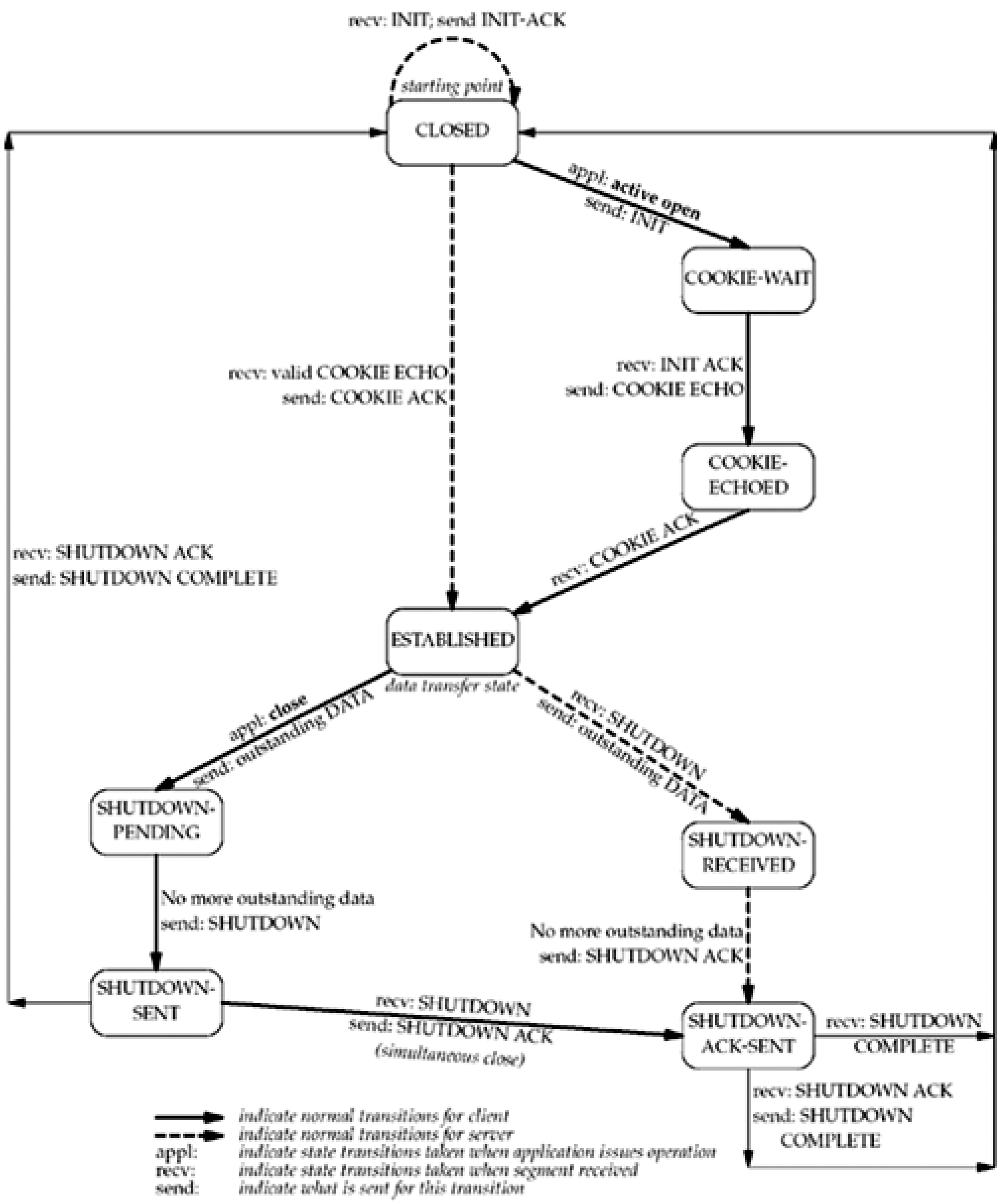
   For example, if an application performs an active open in the CLOSED state, SCTP sends an INIT and the new state is COOKIE-WAIT.

   If SCTP next receives an INIT ACK, it sends a COOKIE ECHO and the new state is COOKIE-ECHOED.

   If SCTP then receives a COOKIE ACK, it moves to the ESTABLISHED state. This final state is where most data transfer occurs.

   The two arrows leading from the ESTABLISHED state deal with the termination of an association.

   If an application calls close before receiving a SHUTDOWN, the transition shifts to the SHUTDOWN-PENDING state.



8. tcp vs udp vs sctp - very high

	TCP	UDP	SCTP
Reliability	trustworthy	Unreliable	Trustworthy
Connection type	Connection-oriented	Connectionless	Connection-oriented
Transmission type	Byte-oriented	News-oriented	News-oriented
Transfer sequence	Strictly ordered	Disordered	Partially ordered
Overload control	Yes	No	Yes
Error tolerance	No	No	Yes

9. wrapper functions - very high

In any real-world program, it is essential to check *every* function call for an error return. We check for errors from socket, inet_pton, connect, read, and fputs, and when one occurs, we call our own functions, err_quit and err_sys, to print an error message and terminate the program.

```
sockfd = Socket(AF_INET, SOCK_STREAM, 0);
```

lib/wrapsock.c

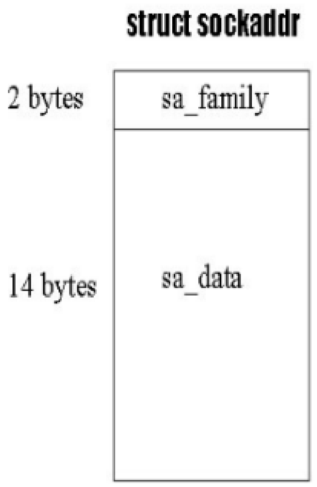
```
int
 Socket(int family, int type, int protocol)
 {
     int     n;

     if ( (n = socket(family, type, protocol)) < 0)
         err_sys("socket error");
     return (n);
 }
Whenever you encounter a function name in the text that begins with an uppercase lett
er, that is one of our wrapper functions. It calls a function whose name is the same
 but begins with the lowercase letter.

When describing the source code that is presented in the text, we always refer to the
lowest level function being called (e.g., socket), not the wrapper function (e.g., So
cket).
```

10. Sock addr vs sock addr_in - pakka

sockaddr

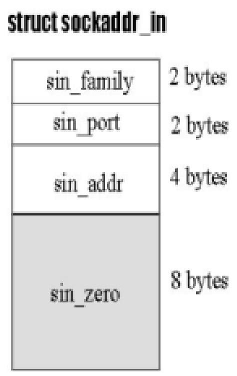


- struct sockaddr is a general structure valid for any protocol. It is the generic socket address type.
- `sa_family` — 16 but integer value identifying the protocol family being used. Eg: TCP/IP → AF_INET.
- `sa_data` — Address information used in protocol family. Eg: TCP/IP → IP address & port no.
- It is used as the base of a set of address structures that acts like a discrimination union sockaddr holding the socket information.

Structure:

```
struct sockaddr {
    unsigned short sa_family;  // Address family( Eg. AF_INET)
    char sa_data[14]; // Family-specific address info
};
```

sockaddr_in



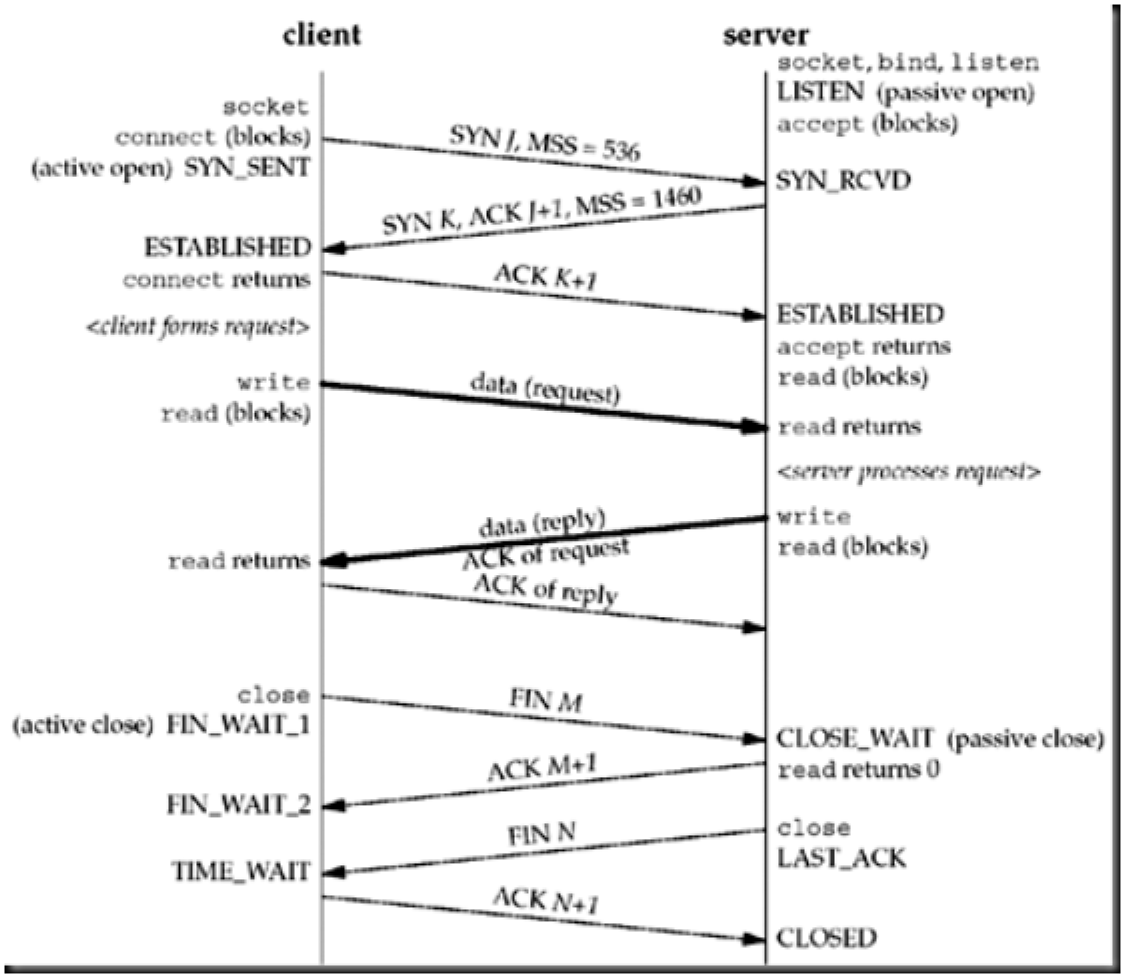
- struct sockaddr_in is protocol specific, to be specific for IPv4 address family.
- `sin_addr` — 32-bit in_addr structure.
- It specifies a transport address & port for the AF_INET address family.

Structure:

```
struct sockaddr_in {
    short           sin_family; // 2 bytes e.g. AF_INET, AF_INET6
    unsigned short  sin_port; // 2 bytes e.g. htons(3490)
    struct in_addr  sin_addr; // 4 bytes see struct in_addr, below
    char            sin_zero[8];// 8 bytes zero this if you want to
};
```

11. TCP - data transfer - high

    Simple explanation will do for this. Same can be asked for 3 way handshake or connection termination



12. TCP vs UDP - use cases - high - 5mks

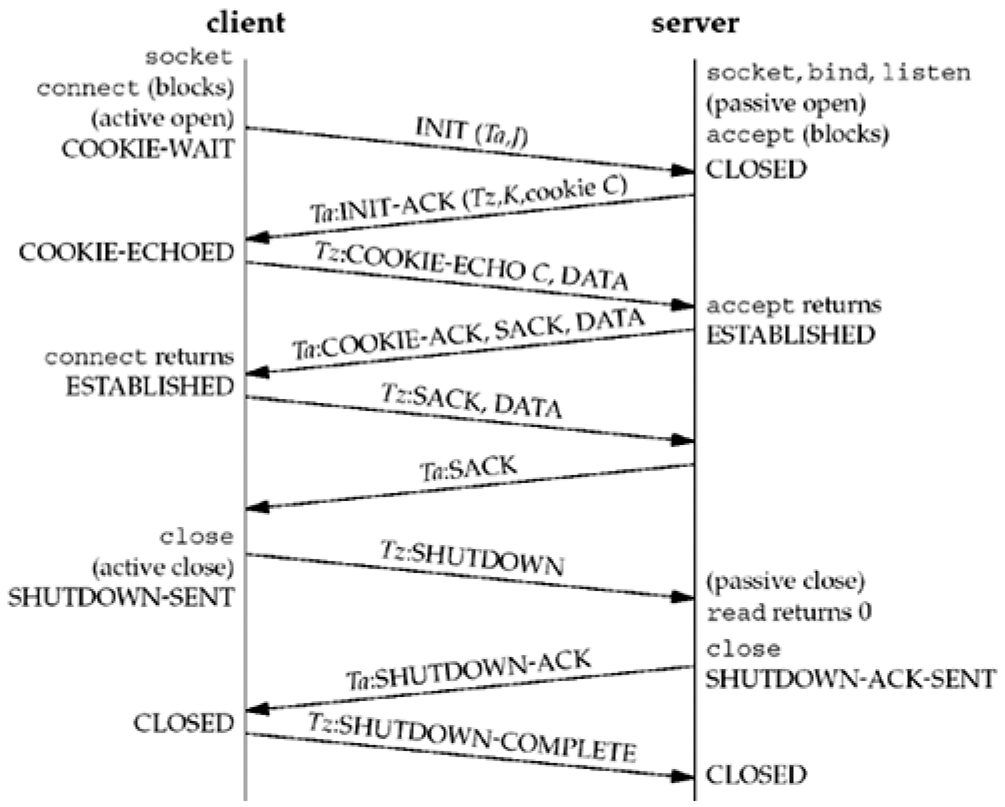
    TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.
    Uses of TCP
    World Wide Web(HTTP)
    E-mail (SMTP TCP)
    File Transfer Protocol (FTP)
    Secure Shell (SSH)

    UDP is a datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast types of network transmission.
    Uses of UDP

Domain Name Server (DNS)
Media streaming
Online multiplayer games
Voice over IP (VoIP)
Tunneling/VPN

13. sctp establishment and teardown - neutral, sctp is also in unit 3, so mostly udhar aayega,

    SImple explanantion with this, same can be asked for connection estavfubgvurebdrfv



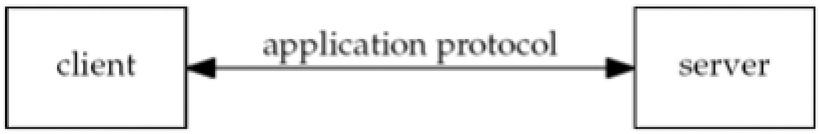
14.What is Network Programming & Decisions to be made - high

Network programming involves writing programs to communicate with processes either on the same or on other machines on the network using standard protocols.

A high-level decision must be made as to which program would initiate the communication first & when responses are expected.

For example, A web server is typically thought of as a long-running program that sends network messages only in response to requests coming in from the network.
The other side of the protocol is a web client, such as a browser which always initiates communication with the server.
This organization into client and server is used by most network-aware applications. Deciding that the client always initiates requests tends to simplify the protocol as well as the programs themselves.

Figure 1.1. Network application: client and server.



15. Difference between IPv4 and IPv6

IPv4	IPv6
IPv4 has a 32-bit address length	IPv6 has a 128-bit address length
Address representation of IPv4 is in decima	lAddress Representation of IPv6 is in
	hexadecimal
The security feature is dependent on the	IPSEC is an inbuilt security feature in
the IPv6	
application	protocol
End to end, connection integrity is	End to end, connection integrity is
achievable	
unachievable	
In IPv4 checksum field is available	In IPv6 checksum field is not available