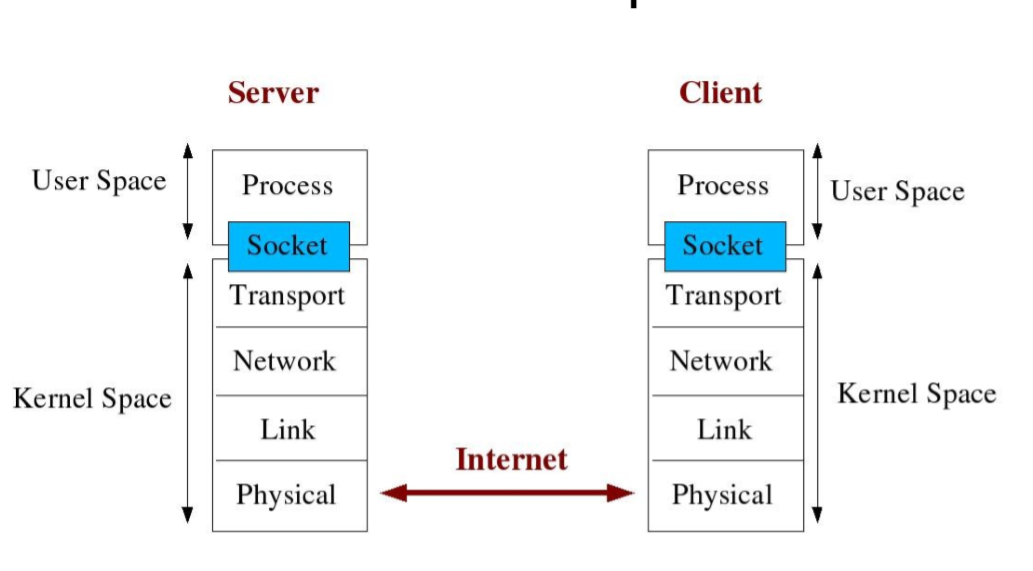
**What is a socket?**

1.Socket: An interface between an application process and transport layer

The application process can send/receive messages to/from another application proces (local or remote)via a socket

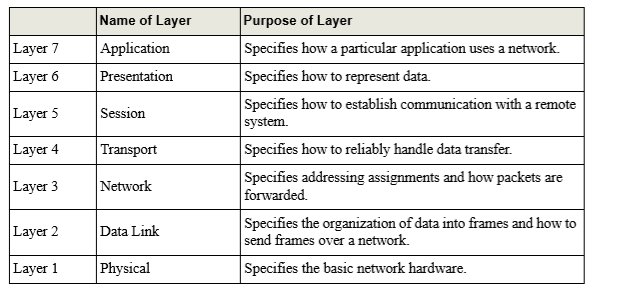
2.A socket is a file descriptor – an integer associated with an open file

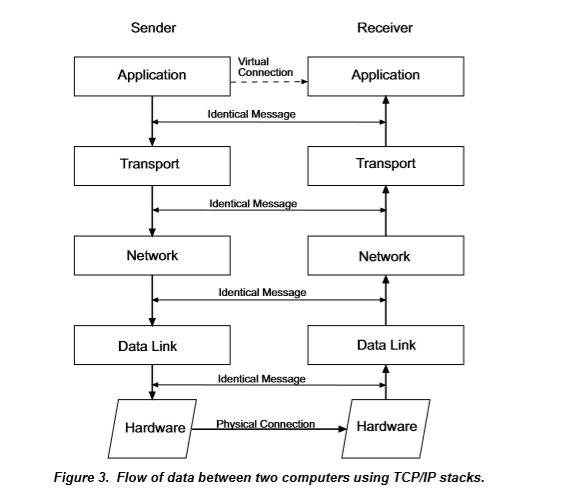
3.Internet sockets characterized by IP Address (4 bytes), port number (2 bytes)



**ISO 7-Layer Model:**

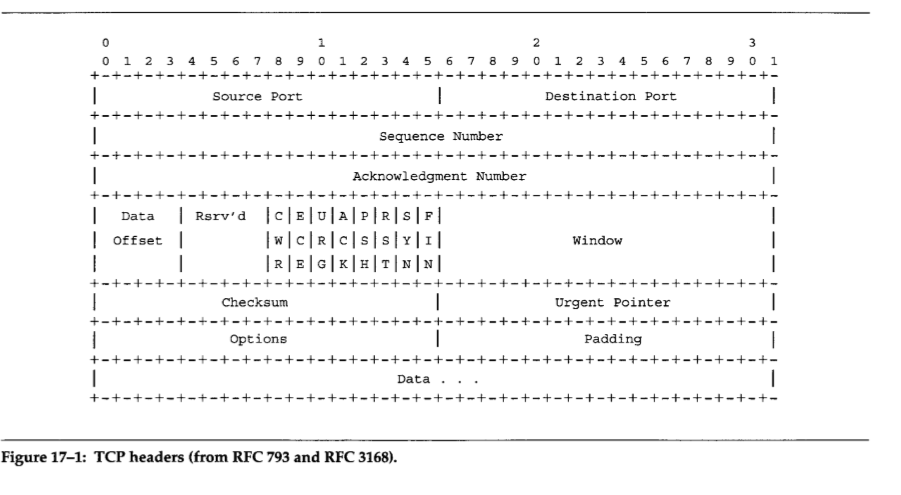
**-----------------------------**





**TCP Header**

**------------------------**

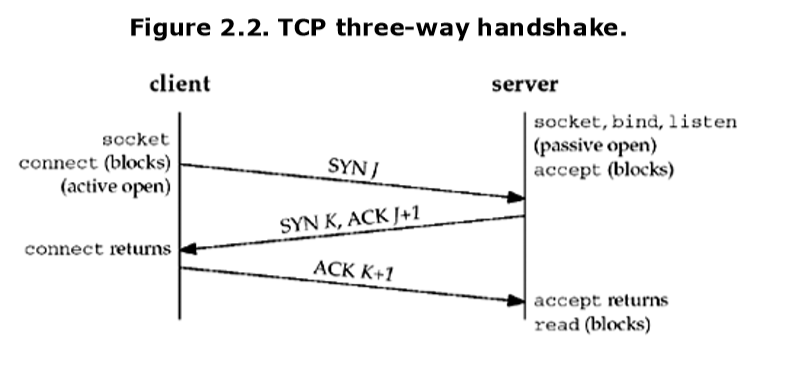


**TCP Connection Oriented**

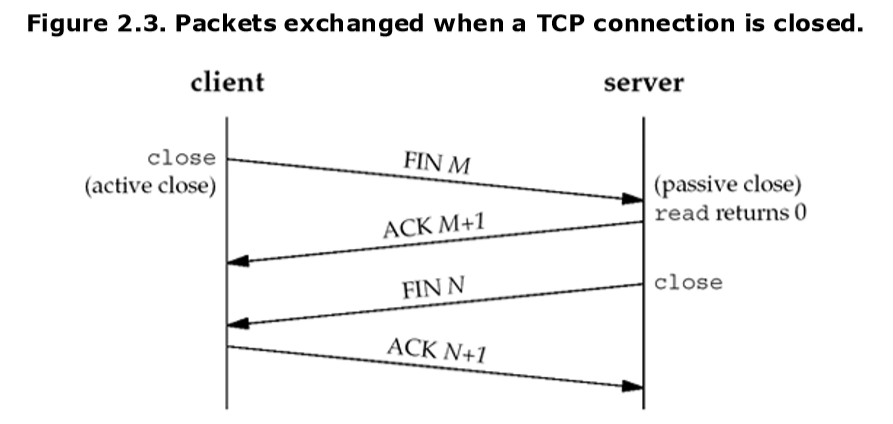
**-----------------------------------**

Three-Way Handshake

The following scenario occurs when a TCP connection is established

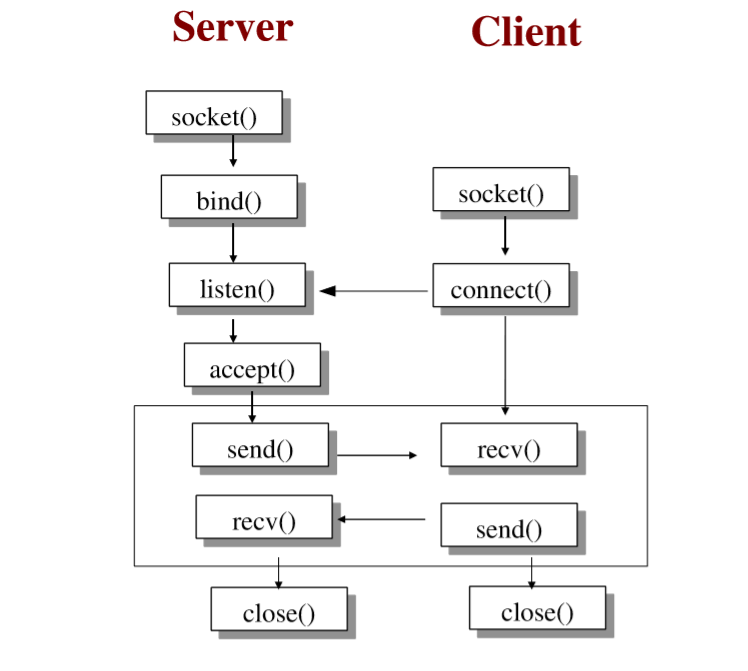


**Connection Establishment**



**Connection Termination**

**Client server communication on TCP**



**socket() -- Get the file descriptor**

int socket(int domain, int type, int protocol);

domain should be set to AF\_INET

type can be SOCK\_STREAM or SOCK\_DGRAM

set protocol to 0 to have socket choose the correct protocol based on type

socket() returns a socket descriptor for use in later system calls or -1 on error

**Socket Structures**

struct sockaddr\_in {

       short int  sin\_family;    // set to AF\_INET

  unsigned short int  sin\_port;        // Port number

struct in\_addr  sin\_addr;       // Internet address

       unsigned char  sin\_zero[8];  //set to all zeros

};

struct sockaddr\_in  my\_addr;

my\_addr.sin\_family = AF\_INET;

my\_addr.sin\_port = htons(MYPORT);

inet\_aton(“10.0.0.5”,&(my\_addr.sin\_addr));

memset(&(my\_addr.sin\_zero),'\0',8);

**bind Function**

The bind function assigns a local protocol address to a socket. With the Internet protocols, the protocol address is the combination of either a 32-bit IPv4 address or a 128-bit IPv6 address, along with a 16-bit TCP or UDP port number.

#include <sys/socket.h>

int bind (int sockfd, const struct sockaddr \*myaddr, socklen\_t addrlen);

A process can bind a specific IP address to its socket. The IP address must belong to an interface on the host. For a TCP client, this assigns the source IP address that will be used for IP datagrams sent on the socket. For a TCP server, this restricts the socket to receive incoming client connections destined only to that IP address.

**listen Function**

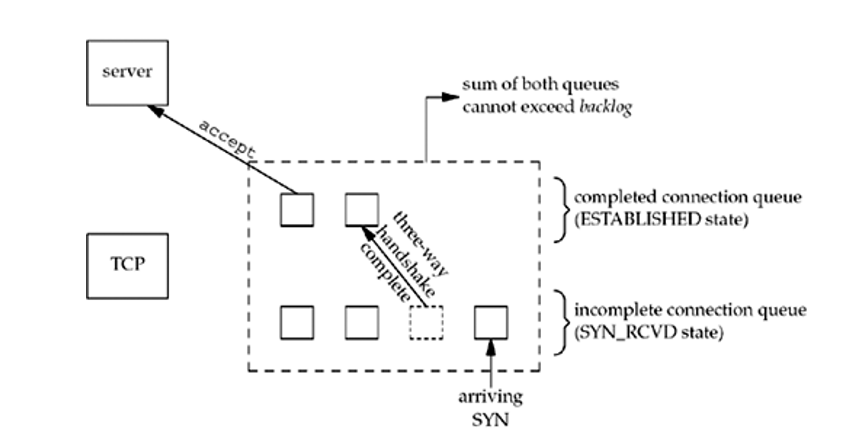
The listen function is called only by a TCP server and it performs two actions:

When a socket is created by the socket function, it is assumed to be an active socket, that is, a client socket that will issue a connect. The listen function converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests directed to this socket.

The second argument to this function specifies the maximum number of connections the kernel should queue for this socket.

#include <sys/socket.h>

#int listen (int sockfd, int backlog);



**accept Function**

accept is called by a TCP server to return the next completed connection from the front of the completed connection queue (Figure 4.7). If the completed connection queue is empty, the process is put to sleep (assuming the default of a blocking socket).

#include <sys/socket.h>

int accept (int sockfd, struct sockaddr \*cliaddr, socklen\_t \*addrlen);

The cliaddr and addrlen arguments are used to return the protocol address of the connected peer process (the client). addrlen is a value-result argument : Before the call, we set the integer value referenced by \*addrlen to the size of the socket address structure pointed to by cliaddr; on return, this integer value contains the actual number of bytes stored by the kernel in the socket address structure

**connect Function**

The connect function is used by a TCP client to establish a connection with a TCP server.

#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr \*servaddr, socklen\_t addrlen);

**Example Client and Server Program(in same machine)**

**Example Client program:**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/select.h>

#include<sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int max(int a,int b)

{

if(a>b)

return a;

else

return b;

}

int main()

{

int fd,sockfd;

sockfd=socket(AF\_INET,SOCK\_STREAM,IPPROTO\_TCP);

fd=0;

struct sockaddr\_in serveraddr,clientaddr;

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(5555);

inet\_aton("127.0.0.1",&serveraddr.sin\_addr.s\_addr);

fd\_set reset;

connect(sockfd,&serveraddr,sizeof(serveraddr));

int maxfd=max(fd,sockfd)+1;

for(;;)

{

FD\_ZERO(&reset);

FD\_SET(fd,&reset);

FD\_SET(sockfd,&reset);

select(maxfd,&reset,0,0,0);

if(FD\_ISSET(fd,&reset))

{

char sendbuf[100]={0};

int n= read(fd,sendbuf,100);

write(sockfd,sendbuf,n);

}

if(FD\_ISSET(sockfd,&reset))

{

char recvbuf[100]={0};

int n=read(sockfd,recvbuf,100);

write(1,recvbuf,n);

}

}

}

**Example Server program**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<sys/select.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int max(int a,int b)

{

if(a>b)

return a;

else

return b;

}

int main()

{

int fd,listenfd;

fd=0;

listenfd=socket(AF\_INET,SOCK\_STREAM,IPPROTO\_TCP);

struct sockaddr\_in serveraddr,clientaddr;

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(5555);

serveraddr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

bind(listenfd,&serveraddr,sizeof(serveraddr));

listen(listenfd,5);

socklen\_t len;

int sockfd=accept(listenfd,&clientaddr,&len);

fd\_set reset;

int maxfd=max(fd,sockfd)+1;

for(;;)

{

FD\_ZERO(&reset);

FD\_SET(fd,&reset);

FD\_SET(sockfd,&reset);

select(maxfd,&reset,0,0,0);

if(FD\_ISSET(fd,&reset))

{

char sendbuf[100]={0};

int n=read(fd,sendbuf,100);

write(sockfd,sendbuf,n);

}

if(FD\_ISSET(sockfd,&reset))

{

char recvbuf[100]={0};

int n=read(sockfd,recvbuf,100);

write(1,recvbuf,n);

}

}

}

**UDP**

UDP Header

