Assignment No.8

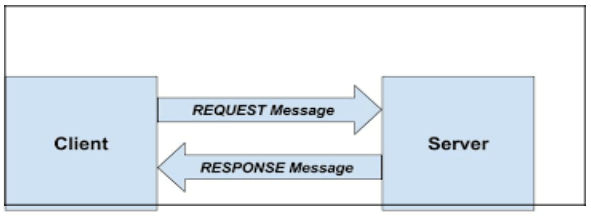
Title:

Write a program using TCP socket for wired network for following
a. Say Hello to Each other
b. File transfer
c. Calculator

Theory:

Client-Server Model

Network applications can be divided into two process: a Client and a Server, with a communication link joining the two processes.



Normally, from Client side it is one-one connection. From the Server Side, it is many-one connection. The standard model for network applications is the Client-Server model. A Server is a process that is waiting to be contacted by a Client process so that server can do something for the client. Typical BSD Sockets applications consist of two separate application level processes; one process (the client) requests a connection and the other process (the server) accepts it.
Client-Server Model Using TCP

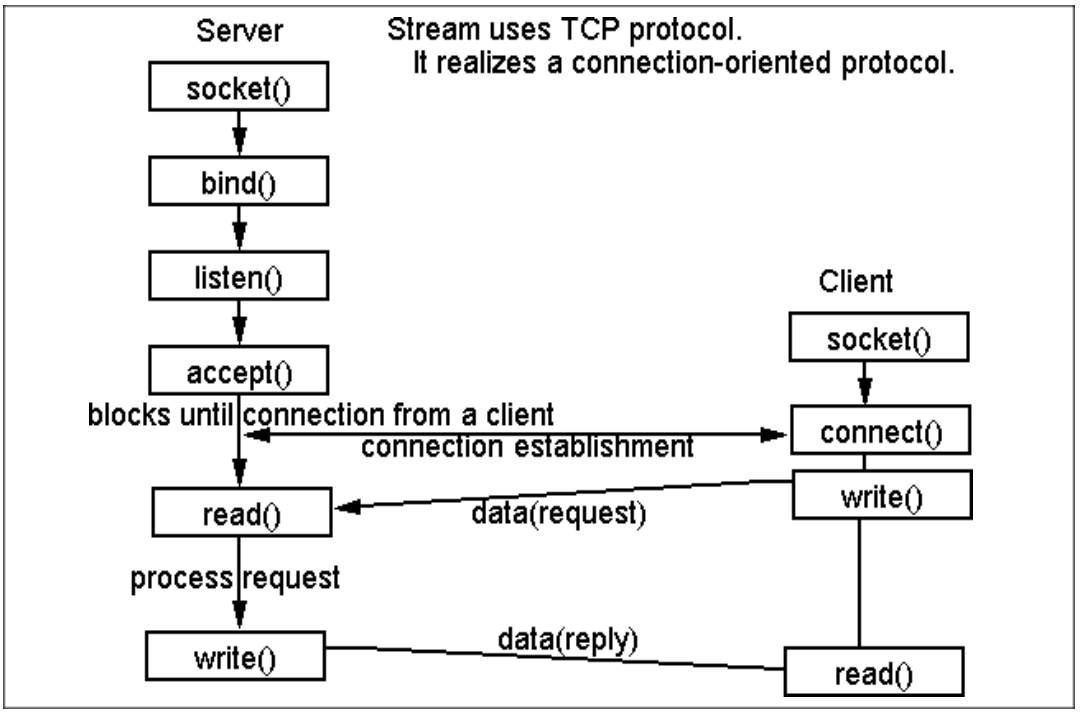
TCP Clients sends request to server and server will receives the request and response with acknowledgement. Every time client communicates with server and receive response from it.
Algorithm to create client and server process is as below.
ALGORITHM:

Server
1. Create a server socket and bind it to port
2. Listen for new connection and when a connection arrives, accept it
3. Read Client's message and display it
4. Get a message from user and send it to client
5. Close the server socket
6. Stop

Client
1. Create a client socket and connect it to the server‟s port number
2. Get a message from user and send it to server
3. Read server's response and display it
4. Close the client socket

5. Stop

Socket functions for TCP client/server in Connection-oriented Scenario



Elementary Socket System Calls

1. socket()	socket() System Call: Creates an end point for communication and returns a descriptor. #include <sys/socket.h> #include <sys/types.h> int socket (int Address Family, int Type, int Protocol); Return Values: Upon successful completion, the socket subroutine returns an integer (the socket descriptor). It returns -1 on error.
2. bind( )	bind( ) System call: Binds a name to a socket. Description: The bind subroutine assigns a Name parameter to an unnamed socket. It assigns a local protocol address to a socket. #include <sys/socket.h> int bind (int sockfd, struct sockaddr *myaddr, int addrlen);

	Return Values: Upon successful completion, the bind subroutine returns a value of 0. Otherwise, it returns a value of -1 to the calling program.
3. connect( )	connect() System call: 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, struct sockaddr *servaddr, int addrlen); Return Values: Upon successful completion, the connect subroutine returns a value of 0. Otherwise, it returns a value of -1 to the calling program.
4. listen()	listen() System call: This system call is used by a connection-oriented server to indicate that it is willing to receive connections. #include <sys/socket.h> int listen (int sockfd, int backlog); Return values: Returns 0 if OK, -1 on error
5. accept()	accept() System call: The actual connection from some client process is waited for by having the server execute the accept system call. #include <sys/socket.h> int accept (int sockfd, struct sockaddr *cliaddr, int *addrlen); Return Values: This system call returns up to three values: an integer return code that is either a new socket descriptor or an error indication, the protocol address of the client process (through the cliaddr pointer), and the size of this address (through the addrlen pointer).
6. read() and write()	read() and write() System call: - read/write from a file descriptor #include <unistd.h> ssize_t read(int *fd*, void **buf*, size_t *count*); ssize_t write(int *fd*, const void **buf*, size_t *count*); Return Values: On success, the number of bytes read or written is returned (zero indicates nothing was written/read). On error, -1 is returned.
7. Send( ), sendto( ), recv( )	Send( ),sendto( ),recv( ) and recvfrom( ) system calls: These system calls are similar to the standard read and write functions, but one additional argument is required.

and recvfrom( )	#include <sys/socket.h>
	int send(int sockfd, char *buff, int nbytes, int flags);
	
	int sendto(int sockfd, char void *buff, int nbytes, int flags, struct sockaddr *to, int addrlen);
	
	int recv(int sockfd, char *buff, int nbytes, int flags);
	
	int recvfrom(int sockfd, char *buff, int nbytes, int flags, struct sockaddr *from, int *addrlen);
	The first three arguments, sockfd, buff and nbytes are the same as the first three arguments to read and write. The flags argument is either 0 or is formed by logically OR'ing one or more of the constants.
	Return Values: All four system calls return the length of the data that was written or read as the value of the function. Otherwise it returns, -1 on error.

8. Close( )	Close( ) system call: The normal Unix close function is also used to close a socket and terminate a TCP connection. #include <unistd.h> int close (int sockfd);

Testing

1. Run Wireshark tool

2. Run client and server program

3. Exchange message

4. Capture TCP packets in Wireshark

Conclusion:
Hence we have studied TCP Socket Programming in C for echo message, file transfer, arithmetic and trigonometric calculator and captured TCP packets in Wireshark tool.