University of Central Punjab

(Faculty of Information Technology)

Course: CSNC 2411

Computer Communications and Networks (Lab)

Lab 3

Socket Programming:
TCP Client Server Communication
(TCP iterative server)

Lab Manual 03

Objectives

- Introduction to TCP
- TCP Iterative Server and Client communication
- TCP related Socket API functions

Reference Material

What is TCP?

TCP (Transmission Control Protocol) is a standard that defines how application programs can exchange data reliably over the Internet, by establishing and maintaining network connections. It provides process to process communication. TCP operates on top of Internet Protocol (IP), which defines how hosts (computers) send/receive packets of data to each other, providing host to host connectivity. Together, TCP and IP are the basic rules defining the Internet operation.

TCP is a connection-oriented protocol, where a connection is established and maintained till the application programs at each end have finished exchanging messages. TCP uses 3-way handshake to establish the connection; and 4-way handshake to close the connection. Below is the sequence of system calls, used for both client and server, to communicate using TCP.

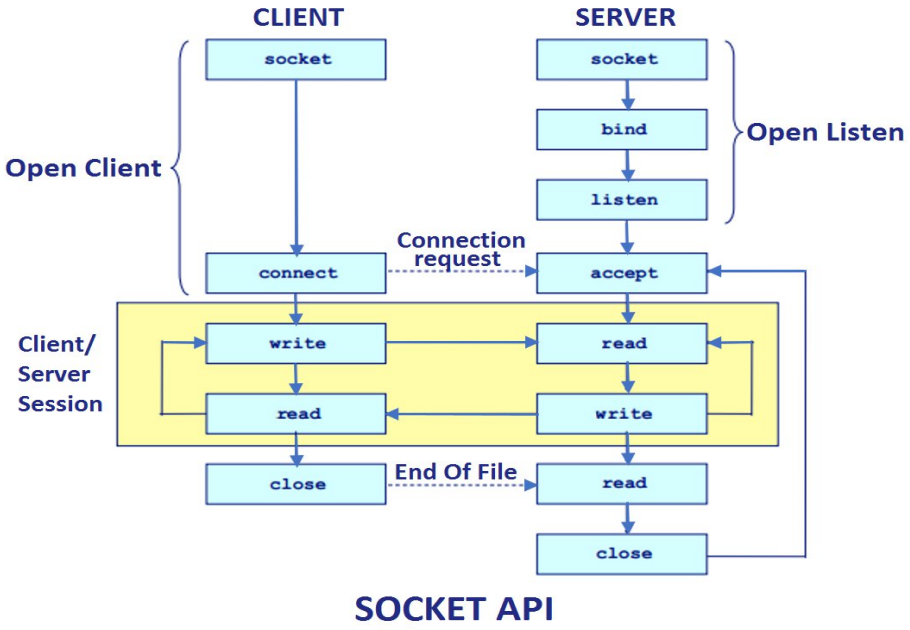

Figure 1: TCP Client Server (Iterative)

A TCP iterative Server can handle only one client at a time. Once the first client's request is serviced, the Server can connect to the next waiting or incoming client.

TCP Related Socket API Calls

Create Socket

Int socket (int family, int type, int protocol);

returns socket descriptor; -1 on error and sets errno

family : address family / protocol family
- AF_INET for IPv4, AF_INET6 for IPv6

type : type of communication
- SOCK_STREAM for TCP
- SOCK_DGRAM for UDP
- SOCK_RAW for Raw socket

Protocol : protocol within family
- typically 0 (except for raw socket)

Bind the Socket

int bind (int sockfd, struct sockaddr* serverAddr, int addrlen);

bind a socket to a local IP address & port number

returns 0 on success; -1 on failure and sets errno

sockfd : socket descriptor (returned from socket)

serverAddr : includes IP address and port number
- IP address set by kernel if value passed is INADDR_ANY, else set by caller
- port number set by kernel if value passed is 0, else set by caller

addrlen : length of address structure
- sizeof (structsockaddr_in)

Listen

int listen (int sockfd, int backlog);

server puts socket into listening state *(wait for connections rather than initiate a connection)*

returns 0 on success; -1 on failure and sets errno

sockfd : socket descriptor

backlog : bound on length of un-accept()ed connection queue (connection backlog)

Accept

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

server accepts a new connection *(first one off the queue of pending connections)*

returns a new socket descriptor (connected socket) created by kernel;
-1 on error and sets errno

sockfd : socket descriptor (listening socket)
cliaddr : IP address and port number of client (when returned)
addrlen : length of address structure

- addrlen is a value-result argument
- caller passes size of client's socket address structure
- kernel returns number of bytes stored in the address structure

Connect

int connect (int sockfd, struct sockaddr* servaddr, int addrlen);

client connects to another socket (server)
returns 0 on success; -1 on failure and sets errno

sockfd : socket descriptor
servaddr : IP address and port number of server
addrlen : length of address structure

Lab Tasks

Task 1. **Your task is to add required socket Api calls in the files provided to run TCP server and client. After that, compile and run both server and client, understand the code and paste the screenshot of the output here. [5 marks]**

Task 2. **In this task,client needs to read the data from the file and encrypt it. Requirements are as follows [15 marks]**

- **Read data from the file named as fileData.txt**
- **Add 3 in all the lowercase letters of the data**
- **Add 2 in all the uppercase letters of the data**
- **Add 1 in the numeric letter of the data**
- **Send the encrypted data to server for decryption**

Requirements for the server are as follows

- **Receive data from client**
- **subtract 3 in all lowercase letters of the data**
- **subtract 2 in all the uppercase letters of the data**
- **subtract 1 in all the numeric letters of the data**
- **Send back the decrypted data to the client**

In the end client will show the decrypted data on terminal sent by the server.

Task 3. **In previous Lab, you have done Client-Server communication for UDP. The flow chart for TCP Client-Server communication is provided in the reference material above. You are now required to write the code for a TCP iterative Server and Client, following the steps from flow chart, and run the TCP Client-Server programs. [30 marks]**

- Take snap of the Server in listening stage in terminal window.
- When client is connected to Server, show its Port number and process ID in terminal window.
- Client sends a file name to Server.
- Server sends the file to Client.
- After receiving file, Client closes its connection with Server.
- But Server should keep running and now be ready to service a new Client request.