Client-Server Communication (TCP Mode)

TCP (Transmission Control Protocol) is a reliable, connection-oriented protocol commonly used for data transmission in networks. By setting up a TCP connection, users can exchange data between two systems in real-time, making it useful for testing, development, or transferring files in a private and secure environment.

This document provides step-by-step instructions for establishing a TCP connection between two Linux systems via the terminal using direct Ethernet communication.

1.a=>TCP Connection{Linux_terminal.1/Linux_terminal.2} :

i. Required Software:

a. Linux in system (ubuntu.v18) .

b. Networking Tools like (**netcat , telnet**) .

ii. Steps establishing connection :

a. Use any protocol to communicate client – server in tcp mode (**SOCKET_PEOGRAMMING , Tcp/Ip programming**) .

b. Using ARP/INET extension .

c. Set static ip in enp0s8 .

d. Create Socket connection between server&client .

//server_side :>

Socket Creation : This creates a new socket for the server to communicate over the network

a. **AF_INET :** specifies the **IPV4** protocol for communication .
b. **SOCKET_STREAM :** indicates socket will use tcp (connection oriented communication) .
c. **SERVER_FD :** uniquely identifies the socket , if resources are missing then returned error -1 .

Binding The Socket : The bind() function associates the socket with specific ip address and port .

d. **SERVER_ADDR.SIN_FAMILY :** For specific IPV4 address.
e. **SERVER_ADDR.SIN_PORT = HTONS(PORT) :** Convert port number into **NETWORK BYTE ORDER** base on system .
f. **BIND() :** error handling .

Listening for Connections : The listen() function puts the server socket into a listening state , making it ready to accept incoming client connection .

g. **SERVER_FD :** server file – descriptor .
h. **BACKLOG :** the maximum number of pending client connections the server will queue.

Accepting a Client Connection :. Accept established connection with a client .

i. **CLIENT_FD :** return a new socket descriptor specific to the client connection.
j. **CLIENT_ADDR :** store client address information .
k. **ADDR_LEN :** contain size of client_addr() .

COMMUNICATION : enable to server exchange data with connected client .

l. **READ(CLIENT_FD , BUFFER , SIZEOF(BUFFER) :** reads data sent by client into buffer .

m. **SEND(CLIENT_FD , HELLO . STRLEN(HELLO) , 0) :** sends a message (“hello From server “).

CLEANUP : closes the client socket **(CLIENT_FD)** and the server socket **(SERVER_FD)** to release system resources .

//client side :>

Socket Creation : This creates a new socket for the server to communicate over the network

a. **AF_INET :** specifies the **IPV4** protocol for communication .
b. **SOCKET_STREAM :** indicates socket will use tcp (connection oriented communication) .
c. **SERVER_FD :** uniquely identifies the socket , if resources are missing then returned error -1 .

SET UP SERVER ADDRESS :

d. **SIN_FAMILY :** For specific IPV4 address.
e. **SIN_PORT :** Set the server port converted to **NETWORK BYTE ORDER** using htons .
f. **INET_PTON :** Convert server ip address from text to binary .

CONNECTED TO SERVER : attempts to established a connection to the server .

g. **SOCK :** the client socket .
h. **SERVER_ADDR :** address of server .

CHAT LOOP

i. INPUT AND SEND :

i.1. FGETS() : User input a msg using fgetS().
i.2. STRCSPN: Removes new line character using strcspn().

j. RECEIVE AND PRINT :

j.1. READ(): reads server response .
j.2. BYTES_RECEIVED <= 0 : if server disconnects loop terminates .
j.3. BUFFER : print server response.

k. LOOP CONTINUITY : continues until client decide exit or disconnects.

Now open two terminal and run server and run client in next terminal . if provide ip belongs to your system than client and server will connect and got a massage “hello from server “ .

1.a=>TCPConnection{Linux_terminal.System1/Linux_terminal.SYstem 2} :

i. Required Software:

a. Linux in system (ubuntu.v18) .

b. Networking Tools like (**netcat , telnet**) .

ii. Steps establishing connection :

a. Use any protocol to communicate client – server in tcp mode (**SOCKET_PEOGRAMMING , Tcp/Ip programming**) .

b. Using ARP/INET extension .

c. Set static ip in enp0s8 .

d. Create Socket connection between server&client .

//server_side :>

Socket Creation : This creates a new socket for the server to communicate over the network

f. **AF_INET :** specifies the **IPV4** protocol for communication .
g. **SOCKET_STREAM :** indicates socket will use tcp (connection oriented communication) .
h. **SERVER_FD :** uniquely identifies the socket , if resources are missing then returned error -1 .

Binding The Socket : The bind() function associates the socket with specific ip address and port .

i. **SERVER_ADDR.SIN_FAMILY :** For specific IPV4 address.
j. **SERVER_ADDR.SIN_PORT = HTONS(PORT) :** Convert port number into **NETWORK BYTE ORDER** base on system .
f. **BIND() :** error handling .

Listening for Connections : The listen() function puts the server socket into a listening state , making it ready to accept incoming client connection .

g. **SERVER_FD :** server file – descriptor .
h. **BACKLOG :** the maximum number of pending client connections the server will queue.

Accepting a Client Connection :. Accept established connection with a client .

i. **CLIENT_FD :** return a new socket descriptor specific to the client connection.
j. **CLIENT_ADDR :** store client address information .
k. **ADDR_LEN :** contain size of client_addr() .

COMMUNICATION : enable to server exchange data with connected client .

l. **READ(CLIENT_FD , BUFFER , SIZEOF(BUFFER) :** reads data sent by client into buffer .
m. **SEND(CLIENT_FD , HELLO . STRLEN(HELLO) , 0) :** sends a message (“hello From server “).

CLEANUP : closes the client socket **(CLIENT_FD)** and the server socket **(SERVER_FD)** to release system resources .

//client side :>

Socket Creation : This creates a new socket for the server to communicate over the network

d. **AF_INET :** specifies the **IPV4** protocol for communication .
e. **SOCKET_STREAM :** indicates socket will use tcp (connection oriented communication) .
f. **SERVER_FD :** uniquely identifies the socket , if resources are missing then returned error -1 .

SET UP SERVER ADDRESS :

d. **SIN_FAMILY :** For specific IPV4 address.
e. **SIN_PORT :** Set the server port converted to **NETWORK BYTE ORDER** using htons .
f. **INET_PTON :** Convert server ip address from text to binary .

CONNECTED TO SERVER : attempts to established a connection to the server .

g. **SOCK :** the client socket .
h. **SERVER_ADDR :** address of server .

CHAT LOOP

i. INPUT AND SEND :

i.1. FGETS() : User input a msg using fgetS().
i.2. STRCSPN: Removes new line character using strcspn().

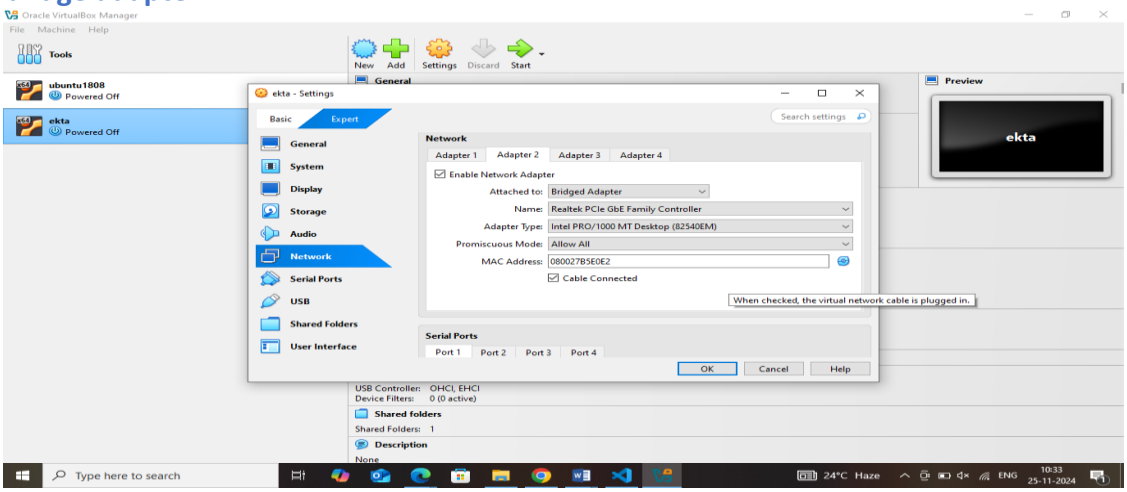
j. RECEIVE AND PRINT :

j.1. READ(): reads server response .
j.2. BYTES_RECEIVED <= 0 : if server disconnects loop terminates .
j.3. BUFFER : print server response.

k. LOOP CONTINUITY : continues until client decide exit or disconnects.

settings added in oracle while we are communicating in liniux from one system to another .

bridge adapter :>



Now open linux terminal in both system and run server and run client in next terminal . if provide ip belongs to your system than client and server will connect and got a massage “hello from server “ .