MIT World Peace University

Computer Networks
Second Year B. Tech, Semester 3

# TCP Socket Programming

Practical Report
Assignment 8

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 29, 2022

# Contents

# 1 Aim and Objectives

To understand Concept of TCP Socket programming.

# 2 Problem Statement

Write a C Program to implement TCP Socket Programming, and simulate a Chat Application using it.

# 3 Platform

**Operating System**: Arch Linux x86-64
**IDEs or Text Editors Used**: Visual Studio Code
**Programs Used**: Cisco Packet Tracer v6.0.1

# 4 Code

```c
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Function designed for chat between client and server.
void func(int connfd)
{
  char buff[MAX];
  int n;
  // infinite loop for chat
  for (;;)
  {
    bzero(buff, MAX);

    // read the message from client and copy it in buffer
    read(connfd, buff, sizeof(buff));
    // print buffer which contains the client contents
    printf("From client: %s\t To client : ", buff);
    bzero(buff, MAX);
    n = 0;
    // copy server message in the buffer
    while ((buff[n++] = getchar()) != '\n')
      ;

    // and send that buffer to client
    write(connfd, buff, sizeof(buff));

    // if msg contains "Exit" then server exit and chat ended.
```

```
37      if (strncmp("exit", buff, 4) == 0)
38      {
39        printf("Server Exit...\n");
40        break;
41      }
42    }
43  }
44
45  // Driver function
46  int main()
47  {
48    int sockfd, connfd, len;
49    struct sockaddr_in servaddr, cli;
50
51    // socket create and verification
52    sockfd = socket(AF_INET, SOCK_STREAM, 0);
53    if (sockfd == -1)
54    {
55      printf("socket creation failed...\n");
56      exit(0);
57    }
58    else
59      printf("Socket successfully created..\n");
60    bzero(&servaddr, sizeof(servaddr));
61
62    // assign IP, PORT
63    servaddr.sin_family = AF_INET;
64    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
65    servaddr.sin_port = htons(PORT);
66
67    // Binding newly created socket to given IP and verification
68    if ((bind(sockfd, (SA *)&servaddr, sizeof(servaddr))) != 0)
69    {
70      printf("socket bind failed...\n");
71      exit(0);
72    }
73    else
74      printf("Socket successfully binded..\n");
75
76    // Now server is ready to listen and verification
77    if ((listen(sockfd, 5)) != 0)
78    {
79      printf("Listen failed...\n");
80      exit(0);
81    }
82    else
83      printf("Server listening..\n");
84    len = sizeof(cli);
85
86    // Accept the data packet from client and verification
87    connfd = accept(sockfd, (SA *)&cli, &len);
88    if (connfd < 0)
89    {
90      printf("server accept failed...\n");
91      exit(0);
92    }
93    else
94      printf("server accept the client...\n");
95
```

```
  // Function for chatting between client and server
  func(connfd);

  // After chatting close the socket
  close(sockfd);
}
```

Listing 1: Server

```
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
  char buff[MAX];
  int n;
  for (;;)
  {
    bzero(buff, sizeof(buff));
    printf("Enter the string : ");
    n = 0;
    while ((buff[n++] = getchar()) != '\n')
      ;
    write(sockfd, buff, sizeof(buff));
    bzero(buff, sizeof(buff));
    read(sockfd, buff, sizeof(buff));
    printf("From Server : %s", buff);
    if ((strncmp(buff, "exit", 4)) == 0)
    {
      printf("Client Exit...\n");
      break;
    }
  }
}

int main()
{
  int sockfd, connfd;
  struct sockaddr_in servaddr, cli;

  // socket create and verification
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd == -1)
  {
    printf("socket creation failed...\n");
    exit(0);
  }
  else
    printf("Socket successfully created..\n");
  bzero(&servaddr, sizeof(servaddr));

```

```
  // assign IP, PORT
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  servaddr.sin_port = htons(PORT);

  // connect the client socket to server socket
  if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) != 0)
  {
    printf("connection with the server failed...\n");
    exit(0);
  }
  else
    printf("connected to the server..\n");

  // function for chat
  func(sockfd);

  // close the socket
  close(sockfd);
}
```

Listing 2: Client

## 5 Output

```
SERVER

Socket successfully created..
Socket successfully binded..
Server listening..
server accept the client...
From client: something
         To client : hello bro
From client: hii
         To client : how are you?
From client: im doing good
         To client : me 2
From client: byee
         To client : byee
From client: exit
         To client : exit
Server Exit...

CLIENT

Socket successfully created..
connected to the server..
Enter the string : something
From Server : hello bro
Enter the string : hii
From Server : how are you?
```

```
Enter the string : im doing good
From Server : me 2
Enter the string : byee
From Server : byee
Enter the string : exit
From Server : exit
Client Exit...
```

Krishnaraj P7.
to Batch # 1 ; PA20.

28/11/22

CN - Assignment -

Theory :

(1) Client Server Communication : Client and servers exchange messages in a request response messaging pattern. The client sends a request and the server sends a response. This exchange of messages is an example of inter-process communication.

(2) Introduction to TCP (Transmission Control Protocol)

It is a standard that defines how to establish & maintain a network conversion by which applications can exchange data.

TCP works with IP (Internet Protocol) which defines how computers send packets of data to each other

(3) The TCP Segment Header :

This is a 4 bit field that indicates the length of the TCP header by a number of 4 byte words in the header. i.e. If the header is 20 bytes (min length of TCP header); then this field will hold 5 (4 x 5 = 20) & maximum length : 60 bytes, then it holds the value 15 as 15 x 4 = 60. So the value of this field is always between 5 and 15.

(*) TCP connection Establishment & Release.

TCP uses a 3-way handshake to establish a reliable connection. The connection is a full duplex & both sides synchronize (S/N) and acknowledge each other. The exchange of these 4 flags is performed in 3 steps.

SYN , SYN - ACK , ACK

(**) Socket: A process sends messages into & receives messages from the network through a software interface called a socket.

(***) TCP socket functions:

The TCP socket is able to listen on the TCP port for incoming connections. The TCP socket is able to initiate a connection to remote server. as well.

(†) TCP socket flow description:

Steps:

Socket () → Bind () → Listen()

↓

Exit(). ← send()/ Rec () ← Accept ()

Client :

Socket ()

↓

Connect ()

↓

write ()

↓

Read ()

↓

close ().

(*) FAQ's

1. State the IANA Range for ports. List atleast well known ports.

→ Well known ports : 0 - 1023

→ Register ports : 1024 - 49151

→ Dynamic Ports : 49152 - 65535

Some well known ports :

(1) 7 - ECHO

(2) 20, 21 - FTP

(3) 22 - SSH

(4) 37 - Time

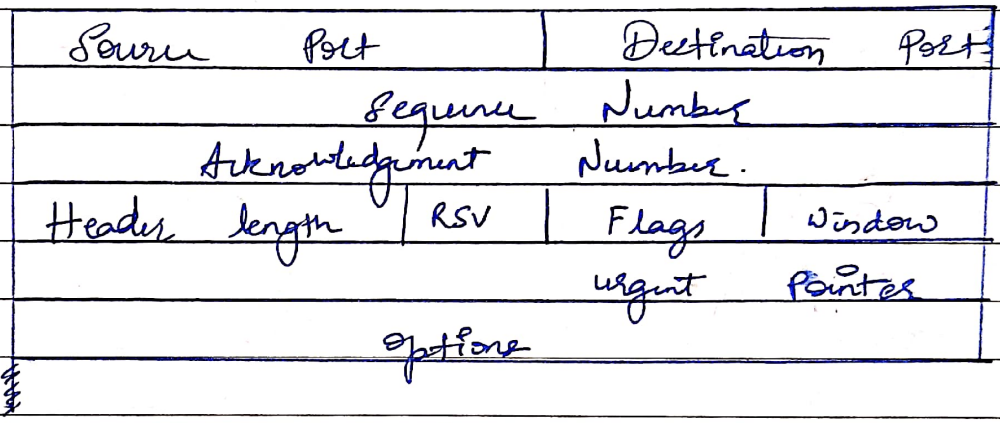
(5) 53 - DNS

(6) 80 - HTTP

Q.2. If bind() fails, what should I do with the socket descriptor?

→ The UNIX system will close all open file descriptors on exit. If the code is not exited the programmer can close it with the close() call.

Q.3. Draw and Explain Header.




(1) Source Port : 16 bit field that specifies sender
(2) Destination Port : Specifies Reciever.
(3) Acknowledgment : 32 bit field that is used by receiver to request next TCP segment.
(4) Header length : 4 bit data indicating length of Header.
(5) Reserved : 3 bit reserved data.
(6) Flag : 9 1 bits for Flags
(7) Window : 16 bit field species how many bytes the sending is willing to receive.