RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous institute affiliated to VTU)

VIDYA SOUDHA, M.S.R NAGAR, M.S.R.I.T. POST, BANGALORE-560054

RAMAIAH

INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE ENGINEERING

Computer Networks Lab Manual (2020-2021)

Semester: 5

Course coordinator:

Sandeep B L

Assistant Professor

Dept. of Information Science and Engineering

M S Ramaiah Institute of Technology, Bangalore

Syllabus

Course code: ISL56	Course Credits : 0:0:1
Title: Computer Networks Laboratory	SEE : **50 Marks**
Total No of Lab Hours : 14 Labs (Each of 2 hours)	
Prerequisite: Data Communications	

Course Objectives

- Design program for file transfer using the concepts of Socket programming.
- Build programs to implement congestion control techniques.
- Construct programs to build optimal routing table.
- Build programs to implement error detection.
- Analyze the network behavior with respect to different parameters and conditions.

Course Content

1. Using TCP/IP sockets, write a client-server program to make client send the file name and the server to send back the contents of the requested file name "sample.txt" with the following contents: "Hello we are at Computer Networks Lab" Display suitable error message in case the file is not present in the server.

2. Write a program to implement a chat application, where the Client establishes a connection with the Server. The Client and Server can send as well as receive messages at the same time. Both the Client and Server exchange messages

3. Given a graph, each node A knows the shortest path to node Z and node A can determine its shortest path to Z by calculating the minimum cost. Now when packet flows through a path it incurs some cost to the network, find shortest paths from source to all nodes in the given graph using Bellman Ford Algorithm. The graph may contain negative weight edges.

4. Given a graph find shortest paths from source to all nodes in the graph using Dijkstra's shortest path algorithm.

5. Write a program for implementing the error detection technique for data transfer in unreliable network code using CRC (16-bits) Technique.

6. Write a program to implement internet checksum for error correction and detection.

7. Write a program to achieve Traffic management at Flow level by implementing Leaky Bucket Algorithm.

8. Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Implement priority queuing as a technique to improve Quality of Service.

9. Write a program to implement Go Back N sliding window protocol

10. Write a program to implement Selective repeat sliding window protocol

11. Write a program to implement the Diffie-Hellman Key Exchange algorithm.

12. Write a program for simple RSA algorithm to encrypt and decrypt the data.

Text books

1. Behrouz A. Forouzan, Data Communications and Networking, Fourth Edition, Tata McGraw-Hill, 2006
2. William Stallings, Cryptography and Network security, Principles and Practices, Third Edition, PHI, 2005

Reference books

1. Alberto Leon-Garcia and Indra Widjaja, Communication Networks –Fundamental Concepts and Key architectures, Second Edition, Tata McGraw-Hill, 2004.
2. William Stallings, Data and Computer Communication, Eight Edition, Pearson Education, 2007.
3. Larry L. Peterson and Bruce S. David, Computer Networks – A Systems Approach, Fourth Edition, Elsevier, 2007.

Course Outcomes

The students will be able to-

1. Design and implement the functionalities of various layers of the OSI model.
2. Implement and analyze the working of different techniques for data handling to ensure secure and error free transmission.

Course Outcomes	Program Outcomes												Program Specific Outcomes		
	PO a	PO b	PO c	PO d	PO e	PO f	PO g	PO h	PO i	PO j	PO k	PO l	PSO 1	PSO 2	PSO 3
1			3	3										3	
2	1	3	2	3	3									3	

Course Assessment

	What		To whom	When/ Where (Frequency in the course)	Max marks	Evidence collected	Contributing to Course Outcomes
Direct Assessment Methods	**CIE**	Internal assessment tests	Students	Twice	15+15	Data sheets	1, 2
		Continuous Evaluation		Regular Labs	10	Records	1,2
		Viva Voce		Once	10	Online result sheet	1, 2
	SEE	Standard examination		End of course	50	Answer scripts	1, 2
Indirect Assessment Methods	Students feedback		Students	Middle of the course	-	Feedback forms	1, 2
	End of course survey			End of course	-	Feedback forms	1, 2

Program 1: Using TCP/IP sockets, write a client-server program to make client send the file name and the server to send back the contents of the requested file name "sample.txt" with the following contents: "Hello we are at Computer Networks Lab". Display suitable error message in case the file is not present in the server.

server.c

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>

int main()

{

  int cs,ns,fd,n;
  int bufsize=1024;
  char *buffer=malloc(bufsize);
  struct sockaddr_in address;
  char fname[255];
  address.sin_family=AF_INET;
  address.sin_port=htons(15000);
  address.sin_addr.s_addr=
INADDR_ANY;
  cs=socket(AF_INET,SOCK_STREAM,0);
  bind(cs,(struct sockaddr *)&address,sizeof(address));
  listen(cs,3);
  ns=accept(cs,(struct sockaddr *)NULL,NULL);
  recv(ns,fname,255,0);
  fd=open(fname,O_RDONLY);
  n=read(fd,buffer,bufsize);
  send(ns,buffer,n,0);
  close(ns);
  return close(cs);
```

}

client.c

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>

int main(int argc,char **argv)
{
  int cs,n;
  int bufsize=1024;
  char *buffer=malloc(bufsize);
  char fname[255];
  struct sockaddr_in address;
  address.sin_family=AF_INET;
  address.sin_port=htons(15000);
  inet_pton(AF_INET,argv[1],&address.sin_addr);
  cs=socket(AF_INET,SOCK_STREAM,0);
  connect(cs,(struct sockaddr *)&address,sizeof(address));
  printf("\nEnter filename: ");scanf("%s",fname);
  send(cs,fname,255,0);
  while((recv(cs,buffer,bufsize,0))>0)
  printf("%s",buffer);
  printf("\nEOF\n");
  return close(cs);
}
```

steps to execute:

-->netstat -tulnp
-->gcc server.c
-->./a.out 631

open another terminal
-->gcc client.c
-->./a.out 127.0.0.1

Program 2: Write a program to implement a chat application, where the Client establishes a connection with the Server. The Client and Server can

send as well as receive messages at the same time. Both the Client and Server exchange messages

TCP server:

```
#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

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

		// read the message from client and copy it in buffer
		read(sockfd, 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(sockfd, buff, sizeof(buff));

		// if msg contains "Exit" then server exit and chat ended.
		if (strncmp("exit", buff, 4) == 0) {
			printf("Server Exit...\n");
			break;
		}
	}
```

```
}

// 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(PORT);

	// 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 acccept failed...\n");
		exit(0);
```

```
	}
	else
		printf("server acccept the client...\n");

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

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

TCP Client:

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#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 varification
	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);
}
```

Compilation –
Server side:
gcc server.c -o server
./server

Client side:
gcc client.c -o client
./client

Output –

Server side:

```
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: hi
     To client : hello
From client: exit
     To client : exit
Server Exit...
```

Client side:

```
Socket successfully created..
connected to the server..
Enter the string : hi
From Server : hello
Enter the string : exit
From Server : exit
Client Exit...
```

Program 3: Given a graph, each node A knows the shortest path to node Z and node A can determine its shortest path to Z by calculating the minimum cost. Now when packet flows through a path it incurs some cost to the network, find shortest paths from src to all nodes in the given graph using Bellman Ford Algorithm. The graph may contain negative weight edges.

```
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{
   int costmat[20][20];
   int nodes,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&nodes);//Enter the nodes
   printf("\nEnter the cost matrix :\n");
   for(i=0;i<nodes;i++)
   {
      for(j=0;j<nodes;j++)
      {
         scanf("%d",&costmat[i][j]);
         costmat[i][i]=0;
         rt[i].dist[j]=costmat[i][j];
         rt[i].from[j]=j;
      }
   }
      do
      {
         count=0;
         for(i=0;i<nodes;i++)
         for(j=0;j<nodes;j++)
         for(k=0;k<nodes;k++)
            if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
            {
               rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
               rt[i].from[j]=k;
               count++;
            }
```

```
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
            for(j=0;j<nodes;j++)
            {
                printf("\t\nnode %d via %d Distance %d
",j+1,rt[i].from[j]+1,rt[i].dist[j]);
            }
        }
    printf("\n\n");

}
```

Program 4: Given a graph find shortest paths from source to all nodes in the graph using Dijkstra's shortest path algorithm.

```
#include <stdio.h>
#define infinity 999
void dij(int n,int v,int cost[10][10],int dist[])
{
 int i,u,count,w,flag[10],min;
 for(i=1;i<=n;i++)
  flag[i]=0,dist[i]=cost[v][i];
 count=2;
 while(count<=n)
 {
  min=99;
  for(w=1;w<=n;w++)
   if(dist[w]<min && !flag[w])
    min=dist[w],u=w;
  flag[u]=1;
  count++;
  for(w=1;w<=n;w++)
   if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
    dist[w]=dist[u]+cost[u][w];
 }
}

void main()
{
 int n,v,i,j,cost[10][10],dist[10];

 printf("\n Enter the number of nodes:");
 scanf("%d",&n);
 printf("\n Enter the cost matrix:\n");
 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
  {
   scanf("%d",&cost[i][j]);
   if(cost[i][j]==0)
    cost[i][j]=infinity;
  }
 printf("\n Enter the source node:");
 scanf("%d",&v);
 dij(n,v,cost,dist);
 printf("\n Shortest path:\n");
```

```
 for(i=1;i<=n;i++)
  if(i!=v)
 printf("%d->%d,cost=%d\n",v,i,dist[i]);
}
```

Program 5: Write a program for implementing the error detection technique for data transfer in unreliable network code using CRC (16-bits) Technique.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<time.h>

void divide(char agdtw[],char divs[],char remd[])

{
   int i,r,l,a,t;
   r=strlen(divs);
   t=strlen(agdtw)-r+1;
   char divd[18],rem[18];
   strncpy(divd,agdtw,r);
   divd[r]='\0';
   l=0;
   memset(rem, 0,18);
   while(l<t)
   {
      a=0;
      memset(rem, 0,18);
      if(divd[0]==divs[0])
      {
         for(i=1;i<r;i++)
         {
            if(divd[i]==divs[i])
               rem[a++]='0';
            else
               rem[a++]='1';
         }
         rem[a]='\0';
         strcpy(divd,rem);
      }
      else
      {
         strncpy(divd,&divd[1],strlen(divd)-1);
         divd[r-1]='\0';
      }
      int o=strlen(divd);
      divd[o]=agdtw[l+r];
      divd[r]='\0';
```

```
        l++;
    }

    strncpy(remd,divd,r-1);
    remd[r-1]='\0';
}

void binary(char letter,char bin[])
{
    int t,c,i=7;
    c=(int)letter;
    while(i>=0)
    {
        t=c%2;
        c=c/2;
        bin[i--]=t+'0';
    }
    bin[8]='\0';
}

char ascii(char bin[])
{
    int t=0,c,i=7;
    while(i>=0)
    {
        t=t+pow(2,7-i)*(bin[i]-'0');
        i--;
    }
    return t;
}

void main()
{
char
dw[126],augdw[1018],div[18],rem[18],cw[1018],rcw[1018],bin[9],rdw[100
1],msg[126];
    printf("Enter a Message to be sent (Max 125 Char)\n");
    fgets(dw, sizeof(dw), stdin);
    binary(dw[0],bin);
    strcpy(augdw,bin);
    int j,k,e;
    for(j=1;j<strlen(dw);j++)
    {
        binary(dw[j],bin);
        strcat(augdw,bin);
```

```
    }
    strcat(augdw,"0000000000000000");

    printf("\nEnter Divisor (generator) of 17 bits\n");
    scanf("%s",div);
    divide(augdw,div,rem);
    strcpy(cw,augdw);
    strcpy(&cw[strlen(augdw)-16],rem);
    strcpy(rcw,cw);
    printf("\nEnter no. of errors to be introduced during transmission :");
    scanf("%d",&e);
    srand(time(0));
    for(j=0;j<e;j++)
    {
        k=rand()%strlen(rcw)-1;
        if(rcw[k]=='0')
            rcw[k]='1';
        else
            rcw[k]='0';
        printf("Error Generated at %d th bit %d thcharacter\n",k,(k/8)+1);
    }
    divide(rcw,div,rem);
    if(strcmp(rem,"0000000000000000")!=0)
        printf("\n\nErroneous Transmission detected!\n");
    strncpy(rdw,rcw,strlen(rcw)-16);
    rdw[strlen(rcw)-16]='\0';
    for(j=0,k=0;j<strlen(rdw);j=j+8)
    {
        strncpy(bin,&rdw[j],8);
        bin[8]='\0';
        msg[k++]=ascii(bin);
    }
    msg[k]='\0';
    printf("\nRecieved Message = %s\n\n",msg);
}

gcc prog5.c -lm
```

Program 6: Write a program to implement internet checksum for error correction and detection.

```
#include<stdio.h>
#include<string.h>
int checksum(int fl)
{
char in[100];
int buf[25];
int i,sum=0,n,temp,temp1;
scanf("%s",in);
if(strlen(in)%2!=0)
        n=(strlen(in)+1)/2;
else
        n=n=(strlen(in))/2;
for(i=0;i<n;i++)
    {
temp=in[i*2];
temp=(temp*256)+in[(i*2)+1];
sum=sum+temp;
    }
if(fl==1)
    {
printf("Enter the checksum value \n");
scanf ("%x", &temp);
sum+=temp;
    }
if(sum%65536!=0)
    {
        n=sum%65536;
sum=(sum/65536) + n;
    }
sum=65535-sum;
printf("%x\n",sum);
return sum;
}
void main()
{
int ch,sum;
do{
printf("1.Encode \n2.Decode \n3.Exit \n");
scanf("%d",&ch);
```

```
switch(ch)
      {

case 1: printf("Enter the string \n");
sum=checksum(0);
printf("Checksum to append is:%x \n",sum);
break;
case 2: printf("Enter the string \n");
sum=checksum(1);
if(sum!=0)
printf("The data has been tampered with or invalid checksum\n");
else
printf("The checksum is valid \n");
break;
case 3: break;
default: printf("Invalid option, try again \n");
      }
   }
while(ch!=3);
}
```

Program 7: Write a program to archive Traffic management at Flow level by implementing Leaky Bucket Algorithm.

```
#include<stdio.h>
#include<stdlib.h>
#define MIN(x,y) (x>y)?y:x

int main()
{
int orate,drop=0,cap,x,count=0,
inp[10]={0},i=0,nsec,ch;
printf(" \n enter bucket size : ");
scanf("%d",&cap);
printf("\n enter output rate :");
scanf("%d",&orate);
do{
printf("\n enter number of packets coming at second %d : ",i+1);
scanf("%d",&inp[i]);
i++;
printf("\n enter 1 to contiue or 0 to quit.....");
scanf("%d",&ch);
}while(ch);
nsec=i;
printf("\n second \t recieved \t sent \t dropped \t remained \n");
for(i=0;count || i<nsec;i++)
{
printf("%d",i+1);
printf(" \t %d\t ",inp[i]);
printf(" \t %d\t ",MIN((inp[i]+count),orate));
if((x=inp[i]+count-orate)>0)
{
if(x>cap)
{
count=cap;
drop=x-cap;
}
else
{
count=x;
drop=0;
}
}
else
```

```
{
drop=0;
count=0;
}
printf(" \t %d \t %d \n",drop,count);
}
return 0;
}
```

Program 8: Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Implement priority queuing as a technique to improve Quality of Service.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
void insert_by_priority(int);
void delete_by_priority(int);
void create();
void check(int);
void display_pqueue();

int pri_que[MAX];
int front, rear;

void main()
{
   int n, ch;

   printf("\n1 - Insert an element into queue");
   printf("\n2 - Delete an element from queue");
   printf("\n3 - Display queue elements");
   printf("\n4 - Exit");

   create();

   while (1)
   {
      printf("\nEnter your choice : ");
      scanf("%d", &ch);

      switch (ch)
      {
      case 1:
         printf("\nEnter value to be inserted : ");
         scanf("%d",&n);
         insert_by_priority(n);
         break;
      case 2:
         printf("\nEnter value to delete : ");
         scanf("%d",&n);
         delete_by_priority(n);
         break;
```

```
        case 3:
            display_pqueue();
            break;
        case 4:
            exit(0);
        default:
            printf("\nChoice is incorrect, Enter a correct choice");
        }
    }
}

/* Function to create an empty priority queue */
void create()
{
    front = rear = -1;
}

/* Function to insert value into priority queue */
void insert_by_priority(int data)
{
    if (rear >= MAX - 1)
    {
        printf("\nQueue overflow no more elements can be inserted");
        return;
    }
    if ((front == -1) && (rear == -1))
    {
        front++;
        rear++;
        pri_que[rear] = data;
        return;
    }
    else
        check(data);
    rear++;
}

/* Function to check priority and place element */
void check(int data)
{
    int i,j;

    for (i = 0; i <= rear; i++)
    {
        if (data >= pri_que[i])
        {
            for (j = rear + 1; j > i; j--)
            {
```

```
            pri_que[j] = pri_que[j - 1];
         }
         pri_que[i] = data;
         return;
      }
   }
   pri_que[i] = data;
}

/* Function to delete an element from queue */
void delete_by_priority(int data)
{
   int i;

   if ((front==-1) && (rear==-1))
   {
      printf("\nQueue is empty no elements to delete");
      return;
   }

   for (i = 0; i <= rear; i++)
   {
      if (data == pri_que[i])
      {
         for (; i < rear; i++)
         {
            pri_que[i] = pri_que[i + 1];
         }

      pri_que[i] = -99;
      rear--;

      if (rear == -1)
         front = -1;
      return;
      }
   }
   printf("\n%d not found in queue to delete", data);
}

/* Function to display queue elements */
void display_pqueue()
{
   if ((front == -1) && (rear == -1))
   {
      printf("\nQueue is empty");
      return;
   }
```

```
    for (; front <= rear; front++)
    {
        printf(" %d ", pri_que[front]);
    }

    front = 0;
}
```

Program 9: Write a program to implement Go Back N sliding window protocol

```
#include<stdio.h>

int main()
{
   int w,i,f,frames[50];

   printf("Enter window size: ");
   scanf("%d",&w);

   printf("\nEnter number of frames to transmit: ");
   scanf("%d",&f);

   printf("\nEnter %d frames: ",f);

   for(i=1;i<=f;i++)
      scanf("%d",&frames[i]);

     printf("\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n");
      printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n",w);

   for(i=1;i<=f;i++)
   {
      if(i%w==0)
      {
         printf("%d\n",frames[i]);
               printf("Acknowledgement of above frames sent is received by sender\n\n");
      }
      else
         printf("%d ",frames[i]);
   }

   if(f%w!=0)
         printf("\nAcknowledgement of above frames sent is received by sender\n");

   return 0;
}
```

Output:

Enter window size: 3

Enter number of frames to transmit: 5

Enter 5 frames: 12 5 89 4 6

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver

12 5 89
Acknowledgement of above frames sent is received by sender

4 6
Acknowledgement of above frames sent is received by sender

Program 10: Write a program to implement Selective repeat sliding window protocol

Steps:

1. Start.
2. Establish connection (recommended UDP)
3. Accept the window size from the client(should be <=40)
4. Accept the packets from the network layer.
5. Calculate the total frames/windows required.
6. Send the details to the client(totalpackets,totalframes.)
7. Initialise the transmit buffer.
8. Built the frame/window depending on the windowsize.
9. Transmit the frame.
10. Wait for the acknowledgement frame.
11. Check for the acknowledgement of each packet and repeat the process for the packet for which the negative acknowledgement isreceived.
    i. Else continue as usual.
12. Increment the frame count and repeat steps 7 to 12 until all packets are transmitted.
13. Close the connection.
14. Stop.

Program

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

#include<sys/types.h>
```

```
#include<netinet/in.h>
#include<netdb.h>
 #define cls() printf("33[H33[J")

//structure definition for designing the packet.
struct frame
{
 int packet[40];
};
//structure definition for accepting the acknowledgement.
struct ack
{
 int acknowledge[40];
};

 int main()
{
 int serversocket;
 sockaddr_in serveraddr,clientaddr;
 socklen_t len;
 int windowsize,totalpackets,totalframes,framessend=0,i=0,j=0,k,l,m,n,repacket[40];
 ack acknowledgement;
 frame f1;
 char req[50];

 serversocket=socket(AF_INET,SOCK_DGRAM,0);
 bzero((char*)&serveraddr,sizeof(serveraddr));
 serveraddr.sin_family=AF_INET;
 serveraddr.sin_port=htons(5018);
 serveraddr.sin_addr.s_addr=INADDR_ANY;
 bind(serversocket,(sockaddr*)&serveraddr,sizeof(serveraddr));
  bzero((char*)&clientaddr,sizeof(clientaddr));
 len=sizeof(clientaddr);

   //connection establishment.
 printf("\nWaiting for client connection.\n");
 recvfrom(serversocket,req,sizeof(req),0,(sockaddr*)&clientaddr,&len);
 printf("\nThe client connection obtained.\t%s\n",req);
```

```
 //sending request for windowsize.
 printf("\nSending request for window size.\n");
 sendto(serversocket,"REQUEST FOR WINDOWSIZE.",sizeof("REQUEST FOR
WINDOWSIZE."),0,(sockaddr*)&clientaddr,sizeof(clientaddr));

//obtaining windowsize.
 printf("\nWaiting for the windowsize.\n");
 recvfrom(serversocket,(char*)&windowsize,sizeof(windowsize),0,(sockaddr*)&clientaddr,&len
);
 cls();
 printf("\nThe windowsize obtained as:\t%d\n",windowsize);
  printf("\nObtaining packets from network layer.\n");
 printf("\nTotal packets obtained:\t%d\n",(totalpackets=windowsize*5));
 printf("\nTotal frames or windows to be transmitted:\t%d\n",(totalframes=5));

 //sending details to client.
 printf("\nSending total number of packets.\n");
 sendto(serversocket,(char*)&totalpackets,sizeof(totalpackets),0,(sockaddr*)&clientaddr,sizeof(c
lientaddr));
 recvfrom(serversocket,req,sizeof(req),0,(sockaddr*)&clientaddr,&len);

 printf("\nSending total number of frames.\n");
 sendto(serversocket,(char*)&totalframes,sizeof(totalframes),0,(sockaddr*)&clientaddr,sizeof(cli
entaddr));
 recvfrom(serversocket,req,sizeof(req),0,(sockaddr*)&clientaddr,&len);

 printf("\nPRESS ENTER TO START THE PROCESS.\n");
 fgets(req,2,stdin);
 cls();

 j=0;
 l=0;                                    //starting the process of sending
 while( l<totalpackets)
 {
                                         //initialising the transmit buffer.
  bzero((char*)&f1,sizeof(f1));
  printf("\nInitialising the transmit buffer.\n");
  printf("\nThe frame to be send is %d with packets:\t",framessend);
                                         //Builting the frame.
```

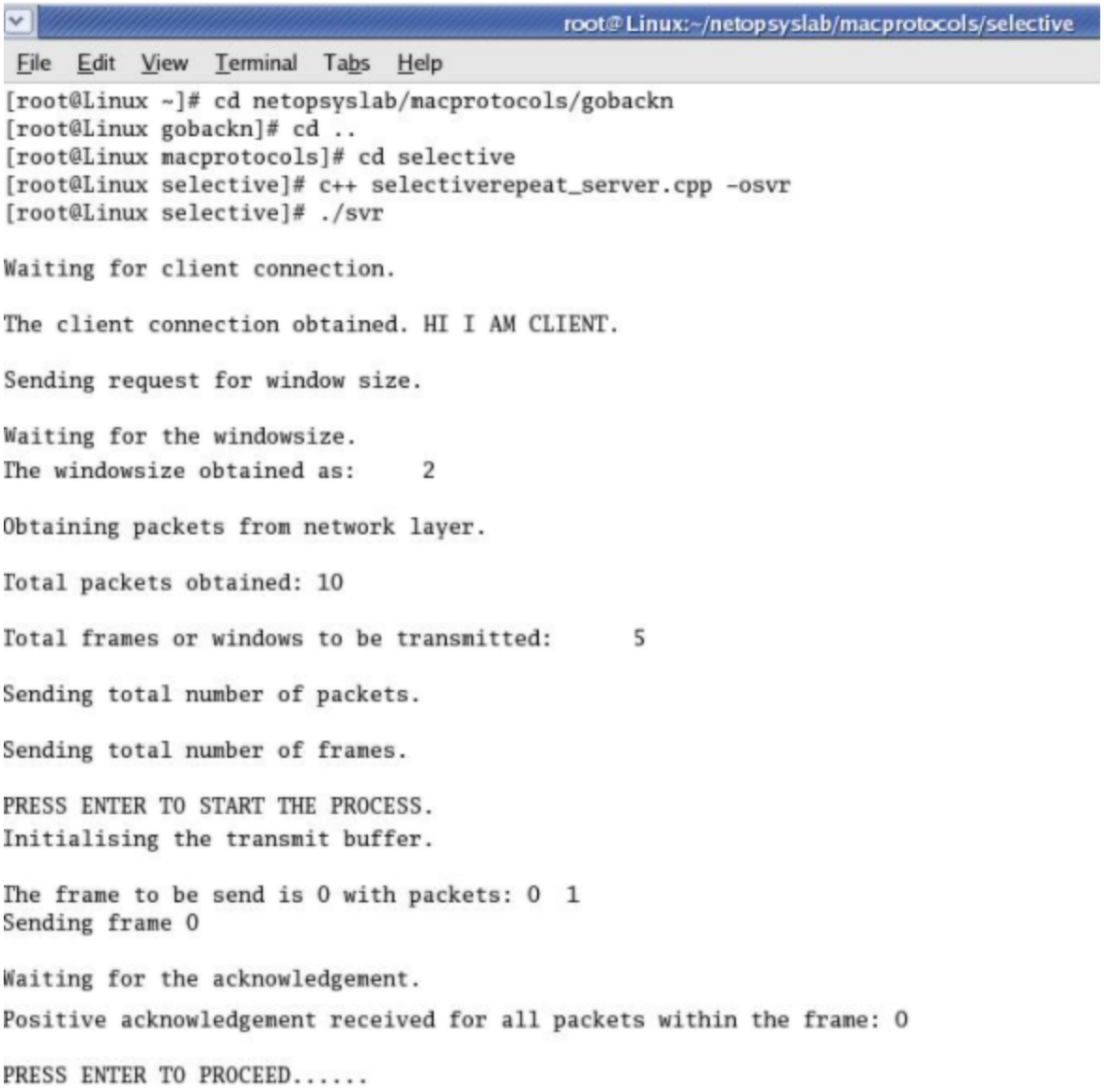
```
for(m=0;m<j;m++)
{
        //including the packets for which negative acknowledgement was received.
 printf("%d  ",repacket[m]);
 f1.packet[m]=repacket[m];
}

while(j<windowsize && i<totalpackets)
{
 printf("%d  ",i);
 f1.packet[j]=i;
 i++;
 j++;
}
printf("\nSending frame %d\n",framessend);

                                        //sending the frame.
sendto(serversocket,(char*)&f1,sizeof(f1),0,(sockaddr*)&clientaddr,sizeof(clientaddr));
                                        //Waiting for the acknowledgement.
printf("\nWaiting for the acknowledgement.\n");
recvfrom(serversocket,(char*)&acknowledgement,sizeof(acknowledgement),0,(sockaddr*)&cli
entaddr,&len);
cls();

                                        //Checking acknowledgement of each packet.
j=0;
k=0;
m=0;
n=l;
while(m<windowsize && n<totalpackets)
{
 if(acknowledgement.acknowledge[m]==-1)
 {
  printf("\nNegative acknowledgement received for packet: %d\n",f1.packet[m]);
  k=1;
  repacket[j]=f1.packet[m];
  j++;
 }
 else
```

```
   {
    l++;
   }
   m++;
   n++;
  }
   if(k==0)
   {
 printf("\nPositive acknowledgement received for all packets within the frame:
%d\n",framessend);
   }
    framessend++;
  printf("\nPRESS ENTER TO PROCEED……\n");
  fgets(req,2,stdin);
  cls();
 }
  printf("\nAll frames send successfully.\n\nClosing connection with the client.\n");
 close(serversocket);
}
```


root@Linux:~/netopsyslab/macprotocols/selective
File Edit View Terminal Tabs Help
[root@Linux ~]# cd netopsyslab/macprotocols/gobackn
[root@Linux gobackn]# cd ..
[root@Linux macprotocols]# cd selective
[root@Linux selective]# c++ selectiverepeat_server.cpp -osvr
[root@Linux selective]# ./svr
Waiting for client connection.
The client connection obtained. HI I AM CLIENT.
Sending request for window size.
Waiting for the windowsize.
The windowsize obtained as: 2
Obtaining packets from network layer.
Total packets obtained: 10
Total frames or windows to be transmitted: 5
Sending total number of packets.
Sending total number of frames.
PRESS ENTER TO START THE PROCESS.
Initialising the transmit buffer.
The frame to be send is 0 with packets: 0 1
Sending frame 0
Waiting for the acknowledgement.
Positive acknowledgement received for all packets within the frame: 0
PRESS ENTER TO PROCEED.....

Program 11: Write a program to implement the Diffie-Hellman Key Exchange algorithm.

The algorithm in itself is very simple. Let's assume that Alice wants to establish a shared secret with Bob. Here is an example of the protocol with secret values in red.

1. Alice and Bob agree to use a prime number p = 23 and base g = 5. (These two values are chosen in this way to ensure that the resulting shared secret can take on any value from 1 to p–1).
2. Alice chooses a secret integer a = 6, then sends Bob A = ga mod p (A = 56 mod 23 = 8)
3. Bob chooses a secret integer b = 15, then sends Alice B = gb mod p (B = 515 mod 23 = 19)
4. Alice computes s = Ba mod p (s = 196 mod 23 = 2)
5. Bob computes s = Ab mod p (s = 815 mod 23 = 2)
6. Alice and Bob now share a secret (the number 2).

The number Alice get at step 4 is same as Bob got at step 5 as

Bob computes

Ab mod p = (ga mod p)b mod p = gab mod p

Alice computes

Ba mod p = (gb mod p)a mod p = gba mod p

```
#include <stdio.h>

// Function to compute a^m mod n
int compute(int a, int m, int n)
{
	int r;
	int y = 1;
```

```
	while (m > 0)
	{
		r = m % 2;

		// fast exponention
		if (r == 1)
			y = (y*a) % n;
		a = a*a % n;

		m = m / 2;
	}

	return y;
}

// C program to demonstrate Diffie-Hellman algorithm

int main()
{
	int p = 23;		// modulus
	int g = 5;		// base

	int a, b;	// a - Alice's Secret Key, b - Bob's Secret Key.
	int A, B;	// A - Alice's Public Key, B - Bob's Public Key

	// choose secret integer for Alice's Pivate Key (only known to Alice)
	a = 6;			// or use rand()

	// Calculate Alice's Public Key (Alice will send A to Bob)
	A = compute(g, a, p);

	// choose secret integer for Bob's Pivate Key (only known to Bob)
	b = 15;			// or use rand()

	// Calculate Bob's Public Key (Bob will send B to Alice)
	B = compute(g, b, p);

	// Alice and Bob Exchanges their Public Key A & B with each other
```

```
	// Find Secret key
	int keyA = compute(B, a, p);
	int keyB = compute(A, b, p);

	printf("Alice's Secret Key is %d\nBob's Secret Key is %d", keyA, keyB);

	return 0;
}
```

Output:

Alice’s Secret Key is 2
Bob’s Secret Key is 2

Program 12: Write a program for simple RSA algorithm to encrypt and decrypt the data.

The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

Algorithm:

- Generate two large random primes, P and Q, of approximately equal size.
- Compute N=PxQ
- Compute Z=(P-1) x (Q-1).
- Choose an integer E, 1<E<Z, such that GCD (E, Z) = 1
- Compute the secret exponent D, 1<D<Z, such that ExD ≡ 1(mod Z)

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

//to find gcd
int gcd(int a, int h)
{
   int temp;
   while(1)
   {
      temp = a%h;
      if(temp==0)
      return h;
      a = h;
      h = temp;
   }
}

int main()
{
   //2 random prime numbers
   double p = 3;
   double q = 7;
```

```
double n=p*q;
double count;
double totient = (p-1)*(q-1);

//public key
//e stands for encrypt
double e=2;

//for checking co-prime which satisfies e>1
while(e<totient){
count = gcd(e,totient);
if(count==1)
   break;
else
   e++;
}

//private key
//d stands for decrypt
double d;

//k can be any arbitrary value
double k = 2;

//choosing d such that it satisfies d*e = 1 + k * totient
d = (1 + (k*totient))/e;
double msg = 12;
double c = pow(msg,e);
double m = pow(c,d);
c=fmod(c,n);
m=fmod(m,n);

printf("Message data = %lf",msg);
printf("\np = %lf",p);
printf("\nq = %lf",q);
printf("\nn = pq = %lf",n);
printf("\ntotient = %lf",totient);
printf("\ne = %lf",e);
printf("\nd = %lf",d);
```

```
    printf("\nEncrypted data = %lf",c);
    printf("\nOriginal Message Sent = %lf",m);

    return 0;
}
```