**EXP:1 DATA LINK LAYER FRAMING METHODS (CHARACTER STUFFING & BIT STUFFING)**

**Program Code:**

**// BIT Stuffing program**

#include<stdio.h>

#include<string.h>

void main()

{

int a[20],b[30],i,j,k,count,n;

printf("Enter frame length:");

scanf("%d",&n);

printf("Enter input frame (0's & 1's only):");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

i=0; count=1; j=0;

while(i<n)

{

if(a[i]==1)

{

b[j]=a[i];

for(k=i+1;a[k]==1 && k<n && count<5;k++)

{

j++;

b[j]=a[k];

count++;

if(count==5)

{

j++;

b[j]=0;

}

i=k;

}}

else

{

b[j]=a[i];

}

i++;

j++;

}

printf("After stuffing the frame is:");

for(i=0;i<j;i++)

printf("%d",b[i]);

}

**Program Output:**

Enter frame length:5

Enter input frame (0's & 1's only):

1

1

1

1

1

After stuffing the frame is:111110

**Program Code:**

**//Program for Character Stuffing**

#include<stdio.h>

#include<string.h>

#include<process.h>

void main()

{

int i=0,j=0,n,pos;

char a[20],b[50],ch;

printf("Enter string\n");

scanf("%s",&a);

n=strlen(a);

printf("Enter position\n");

scanf("%d",&pos);

if(pos>n)

{

printf("invalid position, Enter again :");

scanf("%d",&pos);}

printf("Enter the character\n");

ch=getche();

b[0]='d';

b[1]='l';

b[2]='e';

b[3]='s';

b[4]='t';

b[5]='x';

j=6;

while(i<n)

{

if(i==pos-1)

{

b[j]='d';

b[j+1]='l';

b[j+2]='e';

b[j+3]=ch;

b[j+4]='d';

b[j+5]='l';

b[j+6]='e';

j=j+7;

}

if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')

{

b[j]='d';

b[j+1]='l';

b[j+2]='e';

j=j+3;

}

b[j]=a[i];

i++;

j++;

}

b[j]='d';

b[j+1]='l';

b[j+2]='e';

b[j+3]='e';

b[j+4]='t';

b[j+5]='x';

b[j+6]='\0';

printf("\nframe after stuffing:\n");

printf("%s",b);

}

**Program Output:**

Enter string

MLRITM

Enter position

2

Enter the character

frame after stuffing:

dlestxMdldleLRITMdleetx

**EXP:2-To write a C program to develop a DNS client server to resolve the given**

**Hostname.**

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <netdb.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

void checkHostName(int hostname)

{

if (hostname == -1)

{

perror("gethostname");

exit(1);

}

}

void checkHostEntry(struct hostent \* hostentry)

{

if (hostentry == NULL)

{

perror("gethostbyname");

exit(1);

}

}

void checkIPbuffer(char \*IPbuffer)

{

if (NULL == IPbuffer)

{

perror("inet\_ntoa");

exit(1);

}

}

int main()

{

char hostbuffer[256];

char \*IPbuffer;

struct hostent \*host\_entry;

int hostname;

hostname = gethostname(hostbuffer, sizeof(hostbuffer));

checkHostName(hostname);

host\_entry = gethostbyname(hostbuffer);

checkHostEntry(host\_entry);

IPbuffer = inet\_ntoa(\*((struct in\_addr\*) host\_entry->h\_addr\_list[0]));

printf("Hostname: %s\n", hostbuffer);

printf("Host IP: %s", IPbuffer);

return 0;

}

Output:

Hostname: cContainer

Host IP: 10.98.162.101

**EXP:3- Implement on a dataset of characters the 3 CRC polynomial CRC\_12, CRC\_16, CRC\_CCIP**

**Program:**

// Program for Cyclic Redundancy Check

#include<stdio.h>

int gen[4],genl,frl,rem[4];

void main()

{

int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;

frl=8; genl=4;

printf("Enter frame:");

for(i=0;i<frl;i++)

{

scanf("%d",&fr[i]);

dupfr[i]=fr[i];

}

printf("Enter generator:");

for(i=0;i<genl;i++)

scanf("%d",&gen[i]);

tlen=frl+genl-1;

for(i=frl;i<tlen;i++)

{

dupfr[i]=0;

}

remainder(dupfr);

for(i=0;i<frl;i++)

{

recfr[i]=fr[i];

}

for(i=frl,j=1;j<genl;i++,j++)

{

recfr[i]=rem[j];

}

remainder(recfr);

flag=0;

for(i=0;i<4;i++)

{

if(rem[i]!=0)

flag++;

}

if(flag==0)

{

printf("frame received correctly");

}

else

{

printf("the received frame is wrong");

}

}

remainder(int fr[])

{

int k,k1,i,j;

for(k=0;k<frl;k++)

{

if(fr[k]==1)

{

k1=k;

for(i=0,j=k;i<genl;i++,j++)

{

rem[i]=fr[j]^gen[i];

}

for(i=0;i<genl;i++)

{

fr[k1]=rem[i];

k1++;

}

}

}

}

Program Output:

Enter frame: MLRITM

Enter generator: frame received correctly

**EXP:4-Implement dijkstra's algorithm to compute the shortest path in the graph**

**Program:**

#include <limits.h>

#include <stdbool.h>

#include <stdio.h>

#define V 9

int minDistance(int dist[], bool sptSet[])

{

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++)

if (sptSet[v] == false && dist[v] <= min)

min = dist[v], min\_index = v;

return min\_index;

}

void printSolution(int dist[])

{

printf("Vertex \t\t Distance from Source\n");

for (int i = 0; i < V; i++)

printf("%d \t\t\t\t %d\n", i, dist[i]);

}

void dijkstra(int graph[V][V], int src)

{

int dist[V];

bool sptSet[V];

for (int i = 0; i < V; i++)

dist[i] = INT\_MAX, sptSet[i] = false;

dist[src] = 0;

for (int count = 0; count < V - 1; count++) {

int u = minDistance(dist, sptSet);

sptSet[u] = true;

for (int v = 0; v < V; v++)

if (!sptSet[v] && graph[u][v]

&& dist[u] != INT\_MAX

&& dist[u] + graph[u][v] < dist[v])

dist[v] = dist[u] + graph[u][v];

}

printSolution(dist);

}

int main()

{

int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },

{ 4, 0, 8, 0, 0, 0, 0, 11, 0 },

{ 0, 8, 0, 7, 0, 4, 0, 0, 2 },

{ 0, 0, 7, 0, 9, 14, 0, 0, 0 },

{ 0, 0, 0, 9, 0, 10, 0, 0, 0 },

{ 0, 0, 4, 14, 10, 0, 2, 0, 0 },

{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },

{ 8, 11, 0, 0, 0, 0, 1, 0, 7 },

{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

dijkstra(graph, 0);

return 0;

}

**Output:**

Vertex Distance from Source

0 0

1 4

2 12

3 19

4 21

5 11

6 9

7 8

8 14

**EXP:5 Take an example subnet graph with weights indicating delay between modes. Now, obtain rooting table art for each node using a distance vector routing algorithm using a program in c.**

**Program:**

#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];//initialise the distance equal to cost matrix

rt[i].from[j]=j;

}

}

do

{

count=0;

for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i to k using the cost matrix

//and add the distance from k to node j

for(j=0;j<nodes;j++)

for(k=0;k<nodes;k++)

if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

{//We calculate the minimum distance

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");

getch();

}

**Output:**

A sample run of the program works as:-

Enter the number of nodes :

3

Enter the cost matrix :

0 2 7

2 0 1

7 1 0

For router 1

node 1 via 1 Distance 0

node 2 via 2 Distance 2

node 3 via 3 Distance 3

For router 2

node 1 via 1 Distance 2

node 2 via 2 Distance 0

node 3 via 3 Distance 1

For router 3

node 1 via 1 Distance 3

node 2 via 2 Distance 1

node 3 via 3 Distance 0

**Exp-6**

**Take a example subnet of host obtain broadcast tree to it**

**#include<stdio.h>**

**#include<conio.h> int p,q,u,v,n;**

**int min=99,mincost=0; int t[50][2],i,j;**

**int parent[50],edge[50][50]; main()**

**{**

**clrscr();**

**printf("\n Enter the number of nodes"); scanf("%d",&n);**

**for(i=0;i<n;i++)**

**{**

**printf("%c\t",65+i); parent[i]=-1;**

**}**

**printf("\n"); for(i=0;i<n;i++)**

**{**

**printf("%c",65+i); for(j=0;j<n;j++) scanf("%d",&edge[i][j]);**

**}**

**for(i=0;i<n;i++)**

**{**

**for(j=0;j<n;j++) if(edge[i][j]!=99)**

**if(min>edge[i][j])**

**{**

**min=edge[i][j]; u=i;**

**v=j;**

**}**

**p=find(u); q=find(v); if(p!=q)**

**{ t[i][0]=u;**

**t[i][1]=v; mincost=mincost+edge[u][v]; sunion(p,q);**

**}**

**Else**

**{**

**t[i][0]=-1;t[i][1]=-1;**

**}**

**min=99;**

**}**

**printf("Minimum cost is %d\n Minimum spanning tree is\n" ,mincost); for(i=0;i<n;i++)**

**if(t[i][0]!=-1 && t[i][1]!=-1)**

**{**

**printf("%c %c %d", 65+t[i][0],65+t[i][1],edge[t[i][0]][t[i][1]]);printf("\n");**

**}**

**getch();**

**}**

**sunion(int l,int m)**

**{**

**parent[l]=m;**

**}**

**find(int l)**

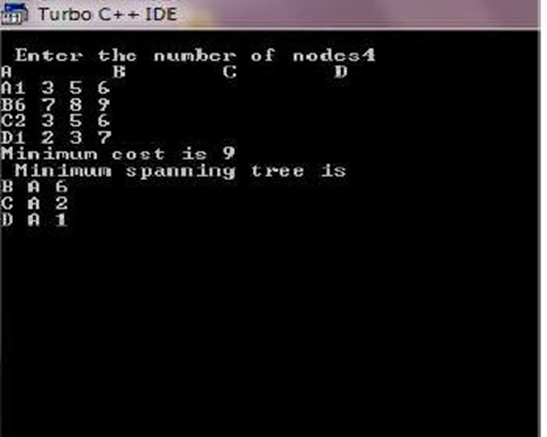
**{**

**if(parent[l]>0)**

**i=parent[i]; return i;**

**}**

**Output:**

****

**Exp:7:**

**Sliding window protocol**

#include<stdio.h>

#include<conio.h>

void main()

{

char sender[50],receiver[50];

int i,winsize;

clrscr();

printf("\n ENTER THE WINDOWS SIZE : ");

scanf("%d",&winsize);

printf("\n SENDER WINDOW IS EXPANDED TO STORE MESSAGE OR WINDOW \n");

printf("\n ENTER THE DATA TO BE SENT: ");

fflush(stdin);

gets(sender);

for(i=0;i<winsize;i++)

receiver[i]=sender[i];

receiver[i]=NULL;

printf("\n MESSAGE SEND BY THE SENDER:\n");

puts(sender);

printf("\n WINDOW SIZE OF RECEIVER IS EXPANDED\n");

printf("\n ACKNOWLEDGEMENT FROM RECEIVER \n");

for(i=0;i<winsize;i++);

printf("\n ACK:%d",i);

printf("\n MESSAGE RECEIVED BY RECEIVER IS : ");

puts(receiver);

printf("\n WINDOW SIZE OF RECEIVER IS SHRINKED \n");

getch();

}

**OUTPUT:**

ENTER THE WINDOWS SIZE : 10

SENDER WINDOW IS EXPANDED TO STORE MESSAGE OR WINDOW

ENTER THE DATA TO BE SENT: ForgetCode.com

MESSAGE SEND BY THE SENDER:

ForgetCode.com

WINDOW SIZE OF RECEIVER IS EXPANDED

ACKNOWLEDGEMENT FROM RECEIVER

ACK:5

MESSAGE RECEIVED BY RECEIVER IS : ForgetCode

WINDOW SIZE OF RECEIVER IS SHRINKED

**EXP:8:simulate the open shortest path routing protocol based on the cost assign to the path**

**Program:**

#include <stdio.h>

#include <string.h>

int main()

{

int count,src\_router,i,j,k,w,v,min;

int cost\_matrix[100][100],dist[100],last[100];

int flag[100];

printf("\n Enter the no of routers");

scanf("%d",&count);

printf("\n Enter the cost matrix values:");

for(i=0;i<count;i++)

{

for(j=0;j<count;j++)

{

printf("\n%d->%d:",i,j);

scanf("%d",&cost\_matrix[i][j]);

if(cost\_matrix[i][j]<0)cost\_matrix[i][j]=1000;

}

}

printf("\n Enter the source router:");

scanf("%d",&src\_router);

for(v=0;v<count;v++)

{

flag[v]=0;

last[v]=src\_router;

dist[v]=cost\_matrix[src\_router][v];

}

flag[src\_router]=1;

for(i=0;i<count;i++)

{

min=1000;

for(w=0;w<count;w++)

{

if(!flag[w])

if(dist[w]<min)

{

v=w;

min=dist[w];

}

}

flag[v]=1;

for(w=0;w<count;w++)

{

if(!flag[w])

if(min+cost\_matrix[v][w]<dist[w])

{

dist[w]=min+cost\_matrix[v][w];

last[w]=v;

}

}

}

for(i=0;i<count;i++)

{

printf("\n%d==>%d:Path taken:%d",src\_router,i,i);

w=i;

while(w!=src\_router)

{

printf("\n<--%d",last[w]);w=last[w];

}

printf("\n Shortest path cost:%d",dist[i]);

}

}

**Output:**

Enter the no of routers3

Enter the cost matrix values:

0->0:

[exam47@cselinux ~]$ ./a.out

Enter the no of routers2

Enter the cost matrix values:

0->0:3

0->1:4

1->0:5

1->1:6

Enter the source router:1

1==>0:Path taken:0

<--1

Shortest path cost:5

1==>1:Path taken:1

Shortest path cost:6

**EXP:9-To write a program to get the mac address from the system using ARP.**

**SOURCE CODE:**

#include<sys/types.h>

#include<sys/socket.h>

#include<net/if\_arp.h>

#include<sys/ioctl.h>

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<math.h>

#include<complex.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<netinet/if\_ether.h>

#include<net/ethernet.h>

#include<stdlib.h>

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

{

struct sockaddr\_in sin={0};

struct arpreq myarp={{0}};

unsigned char \*ptr;

int sd;

sin.sin\_family=AF\_INET;

if(inet\_aton(argv[1],&sin.sin\_addr)==0)

{

printf("IP address Entered '%s' is not valid \n",argv[1]);

exit(0);

}

memcpy(&myarp.arp\_pa,&sin,sizeof(myarp.arp\_pa));

strcpy(myarp.arp\_dev,"echo");

sd=socket(AF\_INET,SOCK\_DGRAM,0);

if(ioctl(sd,SIOCGARP,&myarp)==1)

{

printf("No Entry in ATP cache for '%s'\n",argv[1]);

exit(0);

}

ptr=&myarp.arp\_pa.sa\_data[0];

printf("\nMAC Address for '%s' : ",argv[1]);

printf("%x:%x:%x:%x:%x:%x\n",\*ptr,\*(ptr+1),\*(ptr+2),\*(ptr+3),\*(ptr+4),\*(ptr+5));

printf("\n\t\t\t\t%x:%x:%x:%x:%x:%x\n", myarp.arp\_ha.sa\_data[0],

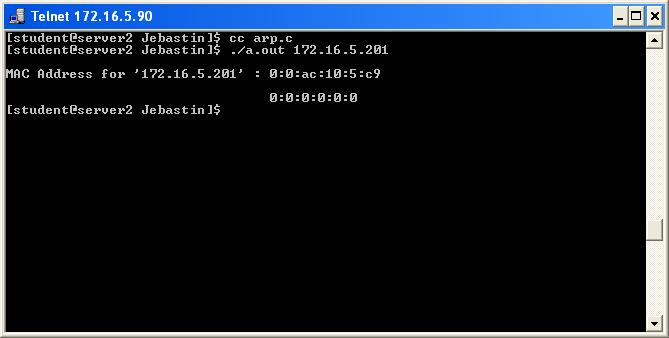
myarp.arp\_ha.sa\_data[1], myarp.arp\_ha.sa\_data[2],

myarp.arp\_ha.sa\_data[3], myarp.arp\_ha.sa\_data[4], myarp.arp\_ha.sa\_data[5]);

return 0;

}

**OUTPUT:**

****

**EXP:10-simulate the implementing routing protocols using border gateway protocol(bgp).**

**Program:**

#include <stdio.h>

#include<conio.h>

int main()

{

int n;

int i,j,k;

int a[10][10],b[10][10];

printf("\n Enter the number of nodes:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("\n Enter the distance between the host %d - %d:",i+1,j+1);

scanf("%d",&a[i][j]);

}}

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d\t",a[i][j]);

}

printf("\n");

}

for(k=0;k<n;k++)

{

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

if(a[i][j]>a[i][k]+a[k][j])

{

a[i][j]=a[i][k]+a[k][j];

}}}}

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

b[i][j]=a[i][j];

if(i==j)

{

b[i][j]=0;

}

}}

printf("\n The output matrix:\n");

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d\t",b[i][j]);

}

printf("\n");

}

getch();

}

**Output:**

[univ10 @ www univ10]$ cc bgp.c

[univ10 @ www univ10]$ ./a.out

Enter the number of nodes:4

Enter the distance between the host 1 - 1:5

Enter the distance between the host 1 - 2:9

Enter the distance between the host 1 - 3:6

Enter the distance between the host 1 - 4:4

Enter the distance between the host 2 - 1:2

Enter the distance between the host 2 - 2:1

Enter the distance between the host 2 - 3:8

Enter the distance between the host 2 - 4:3

Enter the distance between the host 3 - 1:6

Enter the distance between the host 3 - 2:1

Enter the distance between the host 3 - 3:4

Enter the distance between the host 3 - 4:2

Enter the distance between the host 4 - 1:5

Enter the distance between the host 4 - 2:1

Enter the distance between the host 4 - 3:8

Enter the distance between the host 4 - 4:2

5 9 6 4

2 1 8 3

6 1 4 2

5 1 8 2

The output matrix:

0 5 6 4

2 0 8 3

3 1 0 2

3 1 8 0

**EXP:11- To write a client-server application for chat using UDP.**

**Program :**

**// Server Program : udps.c**

#include<stdio.h>

#include<string.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

main()

{

struct sockaddr\_in client,server;

int s,n;

char b1[100],b2[100];

s=socket(AF\_INET,SOCK\_DGRAM,0);

server.sin\_family=AF\_INET;

server.sin\_port=2000;

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

bind(s,(struct sockaddr \*)&server,sizeof(server));

printf("\nServer ready,waiting for client....\n");

n=sizeof(client);

while(1)

{

recvfrom(s,b1,sizeof(b1),0,(struct sockaddr \*) &client,&n);

if(!(strcmp(b1,"end")))

break;

printf("\nClient:%s",b1);

printf("\nServer:");

gets(b2);

sendto(s,b2,sizeof(b2),0,(struct sockaddr \*) &client,n);

}

}

**//Client Program : udpc.c**

#include<stdio.h>

#include<string.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

main()

{

struct sockaddr\_in client,server;

int s,n;

char b1[100],b2[100];

s=socket(AF\_INET,SOCK\_DGRAM,0);

server.sin\_family=AF\_INET;

server.sin\_port=2000;

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

printf("\nClient ready....\n");

n=sizeof(server);

while(1)

{

printf("\nClient:");

gets(b2);

sendto(s,b2,sizeof(b2),0,(struct sockaddr \*) &server,n);

if(strcmp(b2,"end")==0)

break;

recvfrom(s,b1,sizeof(b1),0,NULL,NULL);

printf("\nServer:%s",b1);

}

}

**Output:**

**Terminal 1:** (Server)

student@cselab-desktop:~$ gcc udps.c -o server

/tmp/cc30up4j.o: In function `main':

udps.c:(.text+0x129): warning: the `gets' function is dangerous and should not be used.

student@cselab-desktop:~$ ./server

Server ready,waiting for client....

Client:hi...

Server:hi.. hw r u?

Client:fine.u?

Server:ok.bye...

Client:bye...

Server:end

student@cselab-desktop:~$

**Terminal 2 :(**Client)

student@cselab-desktop:~$ gcc udpc.c -o client

/tmp/ccfz7pFY.o: In function `main':

udpc.c:(.text+0x8b): warning: the `gets' function is dangerous and should not be used.

student@cselab-desktop:~$ ./client

Client ready....

Client:hi...

Server:hi.. hw r u?

Client:fine.u?

Server:ok.bye...

Client:bye...

Server:end

Client:student@cselab-desktop:~$

**EXP:12-To implement raw sockets like packet capturing and filtering using java .**

**Program:**

**//---cat rawtcp.c---**

**// Run as root or SUID 0, just datagram no data/payload #include <unistd.h>**

**#include <stdio.h> #include <sys/socket.h> #include <netinet/ip.h> #include <netinet/tcp.h>**

**// Packet length**

**#define PCKT\_LEN 8192**

**// May create separate header file (.h) for all**

**// headers' structures**

**// IP header's structure struct ipheader {**

**unsigned char iph\_ihl:5, /\* Little-endian \*/ iph\_ver:4;**

**unsigned char iph\_tos; unsigned short int iph\_len; unsigned short int iph\_ident; unsigned char iph\_flags; unsigned short int iph\_offset; unsigned char iph\_ttl; unsigned char iph\_protocol;**

**unsigned short int iph\_chksum; unsigned int iph\_sourceip; unsigned int iph\_destip;**

**};**

**/\* Structure of a TCP header \*/ struct tcpheader {**

**unsigned short int tcph\_srcport; unsigned short int tcph\_destport; unsigned int tcph\_seqnum; unsigned int tcph\_acknum;**

**unsigned char tcph\_reserved:4, tcph\_offset:4;**

**// unsigned char tcph\_flags; unsigned int**

**tcp\_res1:4, /\*little-endian\*/**

**tcph\_hlen:4, /\*length of tcp header in 32-bit words\*/**

**tcph\_fin:1, /\*Finish flag "fin"\*/**

**tcph\_syn:1, /\*Synchronize sequence numbers to start a connection\*/**

**tcph\_rst:1, /\*Reset flag \*/ tcph\_psh:1, /\*Push, sends data to the application\*/**

**tcph\_ack:1, /\*acknowledge\*/ tcph\_urg:1, /\*urgent pointer\*/ tcph\_res2:2;**

**unsigned short int tcph\_win; unsigned short int tcph\_chksum; unsigned short int tcph\_urgptr;**

**};**

**// Simple checksum function, may use others such as Cyclic Redundancy Check, CRC**

**unsigned short csum(unsigned short \*buf, int len)**

**{**

**unsigned long sum; for(sum=0; len>0; len--) sum += \*buf++;**

**sum = (sum >> 16) + (sum &0xffff); sum += (sum >> 16);**

**return (unsigned short)(~sum);**

**}**

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

**{**

**int sd;**

**// No data, just datagram char buffer[PCKT\_LEN];**

**// The size of the headers**

**struct ipheader \*ip = (struct ipheader \*) buffer; struct tcpheader \*tcp = (struct tcpheader \*) (buffer + sizeof(struct ipheader));**

**struct sockaddr\_in sin, din; int one = 1;**

**const int \*val = &one; memset(buffer, 0, PCKT\_LEN);**

**if(argc != 5)**

**{**

**printf("- Invalid parameters!!!\n");**

**printf("- Usage: %s <source hostname/IP> <source port>**

**<target hostname/IP> <target port>\n", argv[0]); exit(-1);**

**}**

**sd = socket(PF\_INET, SOCK\_RAW, IPPROTO\_TCP);**

**if(sd < 0)**

**{**

**perror("socket() error"); exit(-1);**

**}**

**else**

**printf("socket()-SOCK\_RAW and tcp protocol is OK.\n");**

**// The source is redundant, may be used later if needed**

**// Address family sin.sin\_family = AF\_INET; sin\_family = AF\_INET;**

**// Source port, can be any, modify as needed sin.sin\_port = htons(atoi(argv[2])); din.sin\_port = htons(atoi(argv[4]));**

**// Source IP, can be any, modify as needed sin.sin\_addr.s\_addr = inet\_addr(argv[1]); din.sin\_addr.s\_addr = inet\_addr(argv[3]);**

**// IP structure ip->iph\_ihl = 5;**

**ip->iph\_ver = 4;**

**ip->iph\_tos = 16;**

**ip->iph\_len = sizeof(struct ipheader) + sizeof(struct tcpheader);**

**ip->iph\_ident = htons(54321); ip->iph\_offset = 0;**

**ip->iph\_ttl = 64;**

**ip->iph\_protocol = 6; // TCP**

**ip->iph\_chksum = 0; // Done by kernel**

**// Source IP, modify as needed, spoofed, we accept through command line argument**

**ip->iph\_sourceip = inet\_addr(argv[1]);**

**// Destination IP, modify as needed, but here we accept through command line argument**

**ip->iph\_destip = inet\_addr(argv[3]);**

**// The TCP structure. The source port, spoofed, we accept through the command line**

**tcp->tcph\_srcport = htons(atoi(argv[2]));**

**// The destination port, we accept through command line tcp->tcph\_destport = htons(atoi(argv[4]));**

**tcp->tcph\_seqnum = htonl(1); tcp->tcph\_acknum = 0;**

**tcp->tcph\_offset = 5;**

**tcp->tcph\_syn = 1;**

**tcp->tcph\_ack = 0;**

**tcp->tcph\_win = htons(32767);**

**tcp->tcph\_chksum = 0; // Done by kernel tcp->tcph\_urgptr = 0;**

**// IP checksum calculation**

**ip->iph\_chksum = csum((unsigned short \*) buffer, (sizeof(struct ipheader) + sizeof(struct tcpheader)));**

**// Inform the kernel do not fill up the headers' structure, we fabricated our own**

**if(setsockopt(sd, IPPROTO\_IP, IP\_HDRINCL, val, sizeof(one))**

**< 0)**

**{**

**perror("setsockopt() error"); exit(-1);**

**} else**

**printf("setsockopt() is OK\n");**

**printf("Using:::::Source IP: %s port: %u, Target IP: %s port: %u.\n", argv[1], atoi(argv[2]), argv[3], atoi(argv[4]));**

**// sendto() loop, send every 2 second for 50 counts unsigned int count;**

**for(count = 0; count < 20; count++)**

**{**

**if(sendto(sd, buffer, ip->iph\_len, 0, (struct sockaddr**

**\*)&sin, sizeof(sin)) < 0)**

**// Verify**

**{**

**perror("sendto() error"); exit(-1);**

**}**

**else**

**printf("Count #%u - sendto() is OK\n", count); sleep(2);**

**}**

**close(sd); return 0;**

**}**

# **OUTPUT:**

**Setsockopt() is OK**

**Using Source IP : 172.17.1.72 Port : 3001**

**Target IP: 172.17.1.76**

**Port : 5001**