DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Year: 3rd **Semester: 5th**

Computer Network-PCS-552

LAB MANUAL

Prepared By: HOD(CSE)

<Name> <Name>

DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

INDEX

S.No	Practical's Name	Date	Remark
1	**To write a C program to develop a DNS client server to resolve the given hostname.**		
2	**To write a client-server application for chat using UDP**		
3	**To implement programs using raw sockets (like packet capturing and filtering)**		
4	**To write a C program to perform sliding window**		
5	**To get the MAC or Physical address of the system using Address Resolution Protocol.**		
6	**To simulate the Implementing Routing Protocols using border gateway protocol(BGP)**		
7	**To simulate the OPEN SHORTEST PATH FIRST routing protocol based on the cost assigned to the path.**		

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name: Computer Networks.	**Experiment No.** 1	
	Course Code : PCS-552 **Faculty :** Ms. Preeti Raturi	**Branch:** CSE	**Semester:** V

AIM: To write a C program to develop a DNS client server to resolve the given hostname.

ALGORITHM:

1. Create a new file. Enter the domain name and address in that file.
2. To establish the connection between client and server.
3. Compile and execute the program.
4. Enter the domain name as input.
5. The IP address corresponding to the domain name is display on the screen
6. Enter the IP address on the screen.
7. The domain name corresponding to the IP address is display on the screen.
8. Stop the program.

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<netdb.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
int main(int argc,char *argv[1])
{
struct hostent *hen;
if(argc!=2)
{
fprintf(stderr,"Enter the hostname \n");
exit(1);
}
hen=gethostbyname(argv[1]);
if(hen==NULL)
{
fprintf(stderr,"Host not found \n");
}
printf("Hostname is %s \n",hen->h_name);
printf("IP address is %s \n",inet_ntoa(*((struct in_addr *)hen->h_addr)));
}
```

OUTPUT

Thus the above program udp performance using domain name server was executed and successfully.

Outcome:

To understand the DNS client server for resolving the hostname.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name: Computer Networks.	**Experiment No.** 2	
	Course Code : ** PCS-552 **Faculty : Ms. Preeti Raturi	**Branch:** CSE	**Semester:** V

AIM: To write a client-server application for chat using UDP

ALGORITHM: CLIENT

1. Include necessary package in java
2. The client establishes a connection to the server.
3. The client accept the connection and to send the data from client to server and vice versa.
4. The client communicate the server to send the end of the message.
5. Stop the program.

ALGORITHM: SERVER

1. Include necessary package in java
2. The server establishes a connection to the client.
3. The server accept the connection and to send the data from server to client and vice versa.
4. The server communicate the client to send the end of the message
5. Stop the program.

Outcome:

To implement a client server application using connection less service i.e. UDP.

DEV BHOOMI INSTITUTE OF TECHNOLOGY			
LAB MANUAL			
	Course Name: Computer Networks.	**Experiment No.** 3	
	Course Code : ** PCS-552 **Faculty : Ms. Preeti Raturi	**Branch:** CSE	**Semester:** V

AIM: To implement programs using raw sockets (like packet capturing and filtering).

ALGORITHM :

1. Start the program and to include the necessary header files
2. To define the packet length
3. To declare the IP header structure using TCPheader
4. Using simple checksum process to check the process
5. Using TCP \IP communication protocol to execute the program
6. And using TCP\IP communication to enter the Source IP and port number and Target IP address and port number.
7. The Raw socket () is created and accept the Socket ( ) and Send to ( ), ACK
8. Stop the program

```
//---cat rawtcp.c---
// Run as root or SUID 0, just datagram no data/payload
```

Program:

```
#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;
din.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;
}
```

RESULT :

Thus the Above programs using raw sockets TCP \IP (like packet capturing and filtering) was executed and successfully.

Outcome:

To understand packet filtering and capturing using raw sockets.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL			
	Course Name: Computer Networks.	**Experiment No.** 4	
	Course Code : PCS-552 **Faculty :** Ms. Preeti Raturi	**Branch:** CSE	**Semester:** V

AIM:To write a C program to perform sliding window.

ALGORITHM:

1. Start the program.

2. Get the frame size from the user

3. To create the frame based on the user request.

4. To send frames to server from the client side.

5. If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal to client.

6. Stop the program

PROGRAM :

```
// SLIDING WINDOW PROTOCOL
Client :
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
struct mymsgbuf
{
long mtype;
char mtext[25];
};
```

```
FILE *fp;
int main()
{
struct mymsgbuf buf;
int msgid;
int i=0,s;
int count=0,frmsz;
int a[100];
char d;
if((msgid=msgget(89,IPC_CREAT|0666))==-1)
{
printf("\n ERROR IN MSGGET");
exit(0);
}
printf("\n Enter the frame size:");
scanf("%d",&frmsz);
if((fp=fopen("check","r"))==NULL)
printf("\n FILE NOT OPENED");
else
printf("\n FILE OPENED");
while(!feof(fp))
{
d=getc(fp);
a[i]=d;
i++;
}
s=i;
for(i=0;i<frmsz;i++) //print from the check file
printf("\t %c",a[i]);
for(i=0;i<frmsz;i++)
{ if((msgrcv(msgid,&buf,sizeof(buf),0,1))==-1)
{
printf("\n ERROR IN MSGRCV");
exit(0);
}
printf("\n RECEIVED FRAMES ARE:%c",buf.mtext[i]);
}
for(i=0;i<frmsz;i++)
{ if(a[i]==buf.mtext[i])
count++;
} if(count==0)
{
printf("\n FRAMES WERE NOT RECEIVED IN CORRECT SEQ");
exit(0);
} if(count==frmsz)
{
printf("\n FRAMES WERE RECEIVED IN CORRECT SEQ");
} else
{
printf("\n FRAMES WERE NOT RECEIVED IN CORRECT SEQ");
}}
```

Sliding Window Protocol -

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
struct mymsgbuf
{ long mtype;
char mtext[25];
```

```
};
FILE *fp;
int main()
{s
truct mymsgbuf buf;
int si,ei,sz;
int msgid;
int i=0,s;
int a[100];
char d;
if((fp=fopen("send","r"))==NULL)
printf("\n FILE NOT OPENED");
else
printf("\n FILE OPENED");
printf("\n Enter starting and ending index of frame array:");
scanf("%d%d",&si,&ei);
sz=ei-si;
if((msgid=msgget(89,IPC_CREAT|0666))==-1)
{
printf("\n ERROR IN MSGGET");
exit(0);
}
while(!feof(fp))
{
d=getc(fp);
a[i]=d;
i++;
}s
=i;
buf.mtype=1;
for(i=si;i<=ei;i++)
```

```
{
buf.mtext[i]=a[i];
}
for(i=si;i<=ei;i++) //the frames to be sent
printf("\t %c",buf.mtext[i]);
for(i=0;i<=sz;i++)
{ if((msgsnd(msgid,&buf,sizeof(buf),0))==-1)
{
printf("\n ERROR IN MSGSND");
exit(0);
}}
printf("\n FRAMES SENT");
return 0;
}
```

OUTPUT

Thus the above program sliding window protocol was executed and successfully.

Outcome:

To understand the implementation of sliding window protocol.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL			
	Course Name: Computer Networks.	**Experiment No.** 5	
	Course Code : PCS-552 **Faculty :** Ms. Preeti Raturi	**Branch:** CSE	**Semester:** V

AIM: To get the MAC or Physical address of the system using Address Resolution Protocol.

ALGORITHM:

1. Include necessary header files.
2. Initialize the arpreq structure initially to zero.
3. Get the IPAddress of the system as command line argument.
4. Check whether the given IPAddress is valid.
5. Copy the IPAddress from sockaddr_in structure to arpreq structure using miscopy () system call.
6. Create a socket of type SOCK_DGRAM.
7. Calculate the MAC address for the given IPAddress using ioctl() system call.
8. Display the IPAddress and MAC address in the standard output.

Program:

```
#include<unistd.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<net/if_arp.h>
#include<stdlib.h>
#include<stdio.h>
#include<netdb.h>
#include<sys/ioctl.h>
#include<arpa/inet.h>
int main(int argc,char *argv[])
```

```
{ int sd;
unsigned char *ptr;
struct arpreq myarp={{0}};
struct sockaddr_in sin={0};
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,"eth0");
sd=socket(AF_INET,SOCK_DGRAM,0);
if(ioctl(sd,SIOCGARP,&myarp)==1)
{
printf("No entry in ARP cache for%s",argv[1]);
exit(0);
}
ptr=&myarp.arp_ha.sa_data[0];
printf("MAC address for%s",argv[1]);
printf("%x%x%x%x%x%x\n",*ptr,*(ptr+1),*(ptr+2),*(ptr+3),*(ptr+4),*(ptr+5));
return(0);
}
```

OUTPUT

Thus the MAC address was generated for IP address using ARP protocol.

Outcome:

To understand Address Resolution Protocol for resolving MAC or Physical address.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL			
	Course Name: Computer Networks.	**Experiment No.** 6	
	Course Code : PCS-552 **Faculty :** Ms. Preeti Raturi	**Branch:** CSE	**Semester:** V

AIM: To simulate the Implementing Routing Protocols using border gateway protocol(BGP)

ALGORITHM:

1. Read the no. of nodes n
2. Read the cost matrix for the path from each node to another node.
3. Initialize SOURCE to 1 and include 1
4. Compute D of a node which is the distance from source to that corresponding node.
5. Repeat step 6 to step 8 for n-l nodes.
6. Choose the node that has not been included whose distance is minimum and include that node.
7. For every other node not included compare the distance directly from the source with the distance to reach the node using the newly included node
8. Take the minimum value as the new distance.
9. Print all the nodes with shortest path cost from source node

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

Thus the above program to simulate the Implementing Routing Protocols using border gateway protocol was executed and successfully.

Outcome:

To understand the implementation border gateway protocol(BGP).

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL			
	Course Name: Computer Networks.	**Experiment No.** 7	
	Course Code : PCS-552 **Faculty :** Ms. Preeti Raturi	**Branch:** CSE	**Semester:** V

AIM: To simulate the OPEN SHORTEST PATH FIRST routing protocol based on the cost assigned to the path.

ALGORITHM:

```
1.Read the no. of nodes n
2.Read the cost matrix for the path from each node to another node.
3.Initialize SOURCE to 1 and include 1
4. Compute D of a node which is the distance from source to that corresponding
node.
5.Repeat step 6 to step 8 for n-l nodes.
6.Choose the node that has not been included whose distance is minimum
and include that node.
7.For every other node not included compare the distance directly from the
source with the distance to reach the node using the newly included node
8.Take the minimum value as the new distance.
9.Print all the nodes with shortest path cost from source node
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int a[5][5],n,i,j;
void main()
{
void getdata();
void shortest();
void display();
clrscr();
printf("\n\n PROGRAM TO FIND SHORTEST PATH BETWEEN TWO
NODES\n");
getdata();
shortest();
display();
getch();
}
void getdata()
{
clrscr();
printf("\n\nENTER THE NUMBER OF HOST IN THE GRAPH\n");
scanf("%d",&n);
printf("\n\nIF THERE IS NO DIRECT PATH \n");
printf(" \n\nASSIGN THE HIGHEST DISTANCE VALUE 1000 \n");
for(i=0;i<n;i++)
{
a[i][j]=0;
for(j=0;j<n;j++)
{
if(i!=j)
{
printf("\n\nENTER THE DISTANCE BETWENN (%d,
%d): ",i+1,j+1);
scanf("%d",&a[i][j]);
if(a[i][j]==0)
a[i][j]=1000;
}
}
}
}
void shortest()
{
int i,j,k;
for(k=0;k<n;k++)
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(a[i][k]+a[k][j]<a[i][j])
a[i][j]=a[i][k]+a[k][j];
}
}
void display()
{
int i,j;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(i!=j)
{
printf("\n SHORTEST PATH IS : (%d,%d)--%d\n",i+1,j+1,a[i][j]);
}
getch();
 }
```

OUTPUT

Thus the above program to simulate the Implementing Routing Protocols using open shortest path first (OSPF) was executed and successfully.

Outcome: To understand the implementation of OPEN SHORTEST PATH FIRST for finding the shortest path between source and destination**.**