Exp No: 6

Simulation of ARP & RARP Protocols

Date: 20/09/2020

Name: Swetha Saseendran

Reg No : 185001183

Code:

SERVER.C

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SA struct sockaddr
#define SERVER_PORT 4000
#define MAX 256
#define N_CLIENTS 3


struct arpCatche
{
   char IP[20];
   char MAC[20];
};
struct arpCatche arpTable[10];

struct packet
{
   char srcIP[20];
   char srcMAC[20];
   char dstIP[20];
   char dstMAC[20];
   char data[20];
};


int error(char *msg)
{
   perror(msg);
   exit(1);
}
```

```
int lookUp(char IP[20],char MAC[20])
{
   if(strlen(MAC) == 0) //check for dest client_fd
      for (int i = 0; i < N_CLIENTS; i++)
         if(strcmp(IP,arpTable[i].IP) == 0)
            return i;

   else //check MAC and IP validity
      for (int i = 0; i < N_CLIENTS; i++)
         if(strcmp(IP,arpTable[i].IP) == 0)
            if(strcmp(MAC,arpTable[i].MAC) == 0)
               return i;
   return -1;
}


int main(int argc, char const *argv[])
{
   int server_fd, client_fd,sock_fd;
   int addrlen,index;
   int clients[10];
   char buf[MAX], packet_str[MAX];
   struct packet pkt;
   struct sockaddr_in servaddr, clientaddr;


   //IP & MAC of Server
   char serIP[]="192.168.1.1";
    strcpy(pkt.srcIP,serIP);
   char serMAC[]="AF-45-E5-00-97-12";
     strcpy(pkt.srcMAC,serMAC);

   //CREATE SOCKET
   if ((server_fd = socket(AF_INET,SOCK_STREAM,0)) <= 0)
         error("SOCKET FAILED");

   //RESET servaddr
   bzero(&servaddr, sizeof(servaddr));

      //ASSIGN IP, PORT, FAMILY
   servaddr.sin_port = htons(SERVER_PORT);
   servaddr.sin_family = AF_INET;
         add - ANY.
   //BIND TO PORT
   if ((bind(server_fd, (SA*)&servaddr, sizeof(servaddr))) != 0)
         error("BIND ERROR");
```

```
//LISTEN ON PORT
   if ((listen(server_fd, 3)) != 0)
         error("LISTEN ERROR");
   printf("SERVER LISTENING ON PORT 4000\n");

   //ACCEPT CONNECTION FROM ALL CLIENTS
   index = 0;
   while(index < N_CLIENTS)
   {
      addrlen = sizeof(clientaddr);
      if ((client_fd = accept(server_fd, (SA*)&clientaddr, &addrlen)) <= 0)
            error("ACCEPT ERROR");
      printf("\nCLIENT %d CONNECTED", index+1);
      clients[index] = client_fd;

      bzero(buf, MAX);
      read(client_fd, buf, MAX); //IP Address of Client
      strcpy(arpTable[index].IP,buf);

      bzero(buf, MAX);
      read(client_fd, buf, MAX); //MAC Address of Client
      strcpy(arpTable[index].MAC,buf);

      index ++;
   }

   //SERVER AS A HOST

   //GETTING DATA
   printf("\n\nEntering the details of the packet recieved:");

   printf("\n\tDestination IP\t: ");
   scanf("%s",pkt.dstIP);

   //IP Address of src
   printf("\tSource IP\t: %s",serIP);

   //MAC Address of src
   printf("\n\tSource MAC\t: %s",serMAC);

   //Data to be sent
   printf("\n\t16 bit data\t: ");
   scanf("%s",pkt.data);

   //ARP Request packet
   bzero(buf, MAX);
   strcat(buf,pkt.srcIP);
   strcat(buf," | ");
   strcat(buf,pkt.srcMAC);
```

```
strcat(buf," | ");
strcat(buf,pkt.dstIP);
strcpy(packet_str,buf);


//BROADCAST TO ALL CLIENTS
printf("\nDeveloping ARP Request Packet:\n\t %s",buf);
index = 0;
while(index < N_CLIENTS)
{
   write(clients[index], buf, sizeof(buf)); //packet
   write(clients[index], pkt.dstIP, sizeof(pkt.dstIP)); //dstIP
   index++;
}
printf("\nARP Request Packet broadcasted.");

//ARP:  Getting Reply
printf("\n\nWaiting for ARP Reply....");
index = lookUp(pkt.dstIP,"");

if( index == -1 )
{
   printf("\nINVALID IP");
   bzero(buf, MAX);
   strcpy(buf,"\nINVALID IP");
   write(client_fd, buf, sizeof(buf));
}

else
{
   //ARP:  Sending data to dst
   bzero(buf, MAX);
   read(clients[index], buf, sizeof(buf)); //read dstMAC
   strcpy(pkt.dstMAC,buf);
   strcat(packet_str," | ");
   strcat(packet_str,pkt.dstMAC);

   if (lookUp(pkt.dstIP,pkt.dstMAC) != -1) //Checking validity
         printf("VALID IP AND MAC ADDRESS MATCH");

   printf("\nARP reply recieved\t: %s\n",packet_str);
   printf("\n\nSending packet to\t: %s",pkt.dstMAC);
   strcat(packet_str," | ");
   strcat(packet_str,pkt.data);
   strcpy(buf,packet_str);
   write(clients[index], buf, sizeof(buf));
   printf("\nPacket sent\t: %s\n",buf);
}
```

```
    //CLOSE SOCKET
    close(server_fd);

    return 0;
}
```

CLIENT.C

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SA struct sockaddr
#define SERVER_PORT 4000
#define MAX 256

int error(char *msg)
{
    perror(msg);
    exit(1);
}

int main()
{
    int sock_fd = 0,flag = 0;
    struct sockaddr_in servaddr;
    //PACKET
    int n;
    char buf[MAX];
    //IP & MAC
    char IP[MAX];
    char MAC[MAX];

    //CREATE SOCKET
    if ((sock_fd = socket(AF_INET,SOCK_STREAM,0)) <= 0)
        error("SOCKET FAILED");

    //RESET servaddr
    bzero(&servaddr, sizeof(servaddr));

    // ASSIGN IP, PORT, FAMILY  OF SERVER
    servaddr.sin_port = htons(SERVER_PORT);
    servaddr.sin_family = AF_INET;
```

— ADD any

```
    //CONNECT
    if ((connect(sock_fd, (SA*)&servaddr, sizeof(servaddr))) != 0)
        error("CONNECT ERROR");

    printf("\nEnter IP address\t: "); //Enter IP
    scanf("%s",IP);
    write(sock_fd, IP, sizeof(IP));

    printf("\nEnter MAC address\t: "); //Enter MAC
    scanf("%s",MAC);
    write(sock_fd, MAC, sizeof(MAC));

    //Recieve ARP broadcast packet
    bzero(buf, MAX);
    read(sock_fd, buf, MAX);
    printf("\nARP Request Received\t: %s",buf);

    //Recieve dstIP-check and send MAC
    bzero(buf, MAX);
    read(sock_fd, buf, MAX);
    if(strcmp(IP,buf) == 0)
    {
        flag =1;
        printf("\nIP address matches.");
        write(sock_fd, MAC, sizeof(MAC));;
    }
    else
    {
        printf("\nIP address does not match.\n");
    }

    //Read data if dst
    if(flag == 1)
    {
        bzero(buf, MAX);
        read(sock_fd, buf, MAX);
        printf("\nReceived Packet is\t: %s\n",buf);
    }


    //CLOSE SOCKET
    close(sock_fd);
    return 0;
}
```

Sample Input Output:

SERVER SIDE:

```
swetha@swetha-VirtualBox:~/Desktop$ sudo gcc server.c -o s
swetha@swetha-VirtualBox:~/Desktop$ ./s
SERVER LISTENING ON PORT 4000

CLIENT 1 CONNECTED
CLIENT 2 CONNECTED
CLIENT 3 CONNECTED

Entering the details of the packet recieved:
        Destination IP  : 155.157.65.128
        Source IP       : 192.168.1.1
        Source MAC      : AF-45-E5-00-97-12
        16 bit data     : 1011110000101010

Developing ARP Request Packet:
         192.168.1.1 | AF-45-E5-00-97-12 | 155.157.65.128
ARP Request Packet broadcasted.

Waiting for ARP Reply....  VALID IP AND MAC ADDRESS MATCH
ARP reply recieved      : 192.168.1.1 | AF-45-E5-00-97-12 | 155.157.65.128 | 45-DA-62-21-1A-B2


Sending packet to       : 45-DA-62-21-1A-B2
Packet sent     : 192.168.1.1 | AF-45-E5-00-97-12 | 155.157.65.128 | 45-DA-62-21-1A-B2 | 1011110000101010
swetha@swetha-VirtualBox:~/Desktop$
```

CLIENT SIDE:

```
swetha@swetha-VirtualBox:~/Desktop$ sudo gcc client.c -o c1
swetha@swetha-VirtualBox:~/Desktop$ ./c1

Enter IP address        : 165.43.158.158

Enter MAC address       : 09-DF-90-26-6C-09

ARP Request Received    : 192.168.1.1 | AF-45-E5-00-97-12 | 155.157.65.128
IP address does not match.
swetha@swetha-VirtualBox:~/Desktop$
```

```
swetha@swetha-VirtualBox:~/Desktop$ sudo gcc client.c -o c2
swetha@swetha-VirtualBox:~/Desktop$ ./c2

Enter IP address        : 155.157.65.128

Enter MAC address       : 45-DA-62-21-1A-B2

ARP Request Received    : 192.168.1.1 | AF-45-E5-00-97-12 | 155.157.65.128
IP address matches.
Received Packet is      : 192.168.1.1 | AF-45-E5-00-97-12 | 155.157.65.128 | 45-DA-62-21-1A-B2 | 1011110000101010
swetha@swetha-VirtualBox:~/Desktop$
```

```
swetha@swetha-VirtualBox:~/Desktop$ sudo gcc client.c -o c3
swetha@swetha-VirtualBox:~/Desktop$ ./c3

Enter IP address        : 15.143.158.18

Enter MAC address       : 19-0F-01-63-C7-D4

ARP Request Received    : 192.168.1.1 | AF-45-E5-00-97-12 | 155.157.65.128
IP address does not match.
swetha@swetha-VirtualBox:~/Desktop$
```