**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;

    //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;

    //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

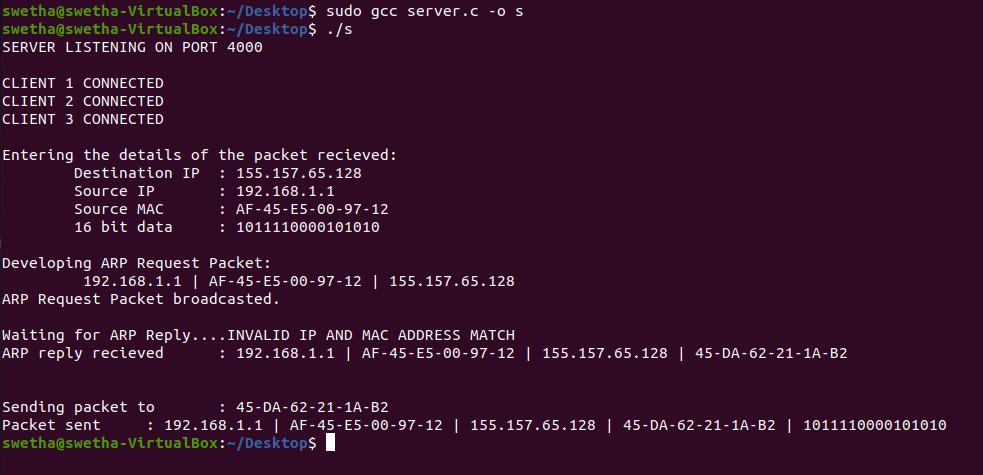
    close(sock\_fd);

    return 0;

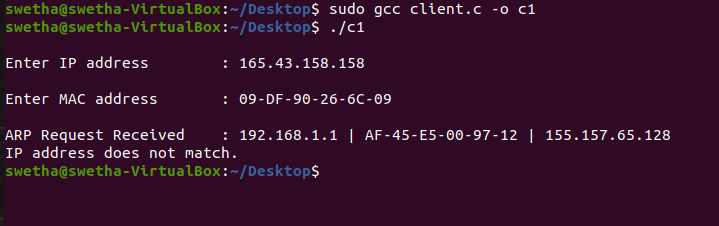
}

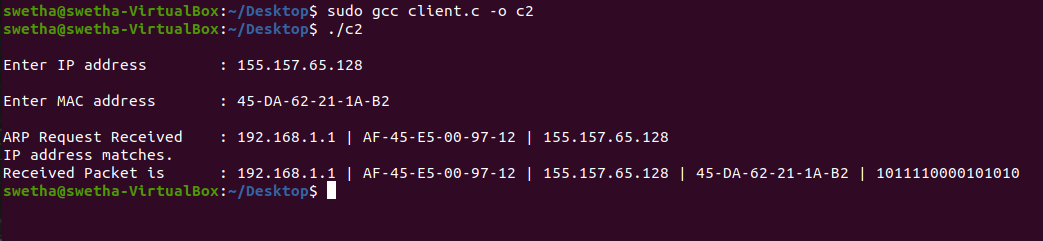
**Sample Input Output:**

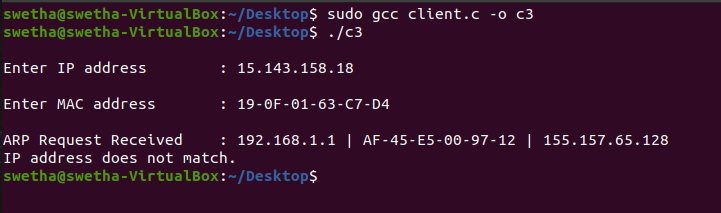
**SERVER SIDE:**

****

**CLIENT SIDE:**

****

****

****