**SSN College of Engineering, Kalavakkam**

**Department of Computer Science and Engineering**

**V Semester - CSE 'B'**

**UCS1511 NETWORKS LAB**

**Date :**08/10/2020 **Name :** Rahul Ram M

**Exercise :** 06 **Reg No :** 185001121

**ADDRESS RESOLUTION PROTOCOL**

**Learning Objective:**

To simulate ARP using socket programming.

**Algorithm for Server:**

1. Reading the value of source IP, destination IP, Source MAC and data from the user.
2. Concatenating all these to form an ARP request packet.
3. Creating a socket using the function socket(domain, type, protocol) which the returns an integer as the status of the socket creation. Here the domain is AF\_INET(iPv4 protocol), type is SOCK\_STEAM and protocol as 0.
4. Using bzero(&server\_addr, sizeof(server\_addr)) function setting values of all the socket structures to null.
5. Using bind() to binf the socket to the address and port number specified in addr(custom data structure). Here, we bind the server to the localhost, hence we use INADDR\_ANY to specify the IP address.
6. listen() function is used to set the server socket in the passive mode, where it waits for the client to approach the server to make a connection, with maximum number of connection in this case is 5.
7. Intitialize all the values of the client\_socket array to 0(means we don’t have to listen to them).
8. Setting a while loop which runs till we terminate using ctrl+z.

* Clear the list of socket descriptors to monitor using FD\_ZERO(&read\_fds).
* Add the descriptor of the server to the list using FD\_SET(sockfd, &read\_fds).
* Assign max\_sd as server\_socket.
* Using for loop over client\_socket array to select valid descriptors , add them to the list of descriptors to monitor and assign the higher number to max\_fd.
* Using select() wait for the activity on one of the sockets in the read\_fds indefinetely(timeout is NULL).
* Using FD\_ISSET(server\_socket, &read\_fds) tests for an incoming connection.
  + If it detects any incoming connection, accept the connection using accept() which creates a socket and assign the new socket any free space in client\_socket array.
  + It also sends the ARP packet request to the client using send().
* Now loop over all the valid fds in the client\_socket array
  + Using FD\_ISSET(sd, &read\_fds) tests for any message from the descriptor.
  + If the received buffer is empty means the connection is terminated by client with server. We then close the descriptor for that client and make the cleint\_sockets[i] to be zero for reuse.
  + Else the buffer contains the ARP packet sent by the client to the server.
  + Extract the MAC address of the client and print it.
  + Now add the Dara to the ARP packet using strcat and send to the client using send().

**Algorithm for Client(same for all the clients):**

1. Reading the IP address and MAC of the client from the user.
2. Creating a socket using the function socket(domain, type, protocol) which the returns an integer as the status of the socket creation. Here the domain is AF\_INET(iPv4 protocol), type is SOCK\_STEAM and protocol as 0.
3. Using bzero(&server\_addr, sizeof(server\_addr)) function setting values of all the socket structures to null.
4. The above two steps are same as the server.
5. The connect() system call connects the socket referred to by the file descriptor socket\_fd to the address specified by server\_addr. Server’s address and port is specified in server\_addr.
6. Recieve the ARP request packet from the server using recv().
7. Extract the source MAC, source IP address, destination IP address from the ARP request packet sent by the server using for loop.
8. Check if the destination IP sent by the server match with IP of this client.
9. If not, print the message and exit.
10. Else, concatenate the MAC address of this client to the ARP packet and send it to the client using send().
11. Recieve the packet from the server that contains the data from the server.

**Program for Server:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

#include <sys/socket.h>

#define PORT 8080

int main()

{

struct sockaddr\_in server\_addr, client\_addr;

char buffer[1024];

char SRC\_IP[100], DEST\_IP[100], SRC\_MAC[100], DEST\_MAC[100], DATA[100], PKT[600];

int client\_sockets[10], max\_sd, fd, sockfd, newfd, ping;

int k, i, count;

socklen\_t len;

fd\_set newfds;

printf("\nEnter the details of packet received.\n");

printf("Destination IP\t: ");

scanf("%s", DEST\_IP);

printf("Source IP\t: ");

scanf("%s", SRC\_IP);

printf("Source MAC\t: ");

scanf("%s", SRC\_MAC);

printf("16 bit data\t: ");

scanf("%s", DATA);

printf("\nDeveloping ARP Request packet\n");

strcpy(PKT, "");

strcat(PKT, SRC\_MAC);

strcat(PKT, "|");

strcat(PKT, SRC\_IP);

strcat(PKT, "|");

strcat(PKT, "00-00-00-00-00-00");

strcat(PKT, "|");

strcat(PKT, DEST\_IP);

printf("\t%s\n", PKT);

printf("\tThe ARP Request packet is broacasted.\n");

printf("Waiting for ARP Reply...\n");

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if(sockfd < 0)

{

perror("Unable to open socket.\n");

}

bzero(&server\_addr, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

server\_addr.sin\_port = htons(PORT);

if(bind(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0)

{

perror("Bind error occurred.\n");

}

listen(sockfd, 5);

for(i = 0; i < 10; i++)

{

client\_sockets[i] = 0;

}

len = sizeof(client\_addr);

while(1)

{

FD\_ZERO(&newfds); //Clears socket set.

FD\_SET(sockfd, &newfds); //Add sockfd to socket set.

max\_sd = sockfd;

for(i = 0; i < 10; i++){

fd = client\_sockets[i];

if(fd > 0){

FD\_SET(fd, &newfds);

}

if(fd > max\_sd){ //Store the max valued FD.

max\_sd = fd;

}

}

//Wait indefinitely till any client pings.

ping = select(max\_sd+1, &newfds, NULL, NULL, NULL);

if(ping < 0){

perror("Select error occurred.\n");

}

//if sockfd change => new connection request.

if(FD\_ISSET(sockfd, &newfds)){

newfd = accept(sockfd, (struct sockaddr\*)&client\_addr, &len);

if(newfd < 0){

perror("Unable to accept the new connection.\n");

}

strcpy(buffer, PKT);

send(newfd, buffer, sizeof(buffer), 0);

//Add the new client on an empty slot.

for(i = 0; i < 10; i++){

if(client\_sockets[i] == 0){

client\_sockets[i] = newfd;

break;

}

}

}

// checking for any response from any connected clients.

for(i = 0; i < 10; i++)

{

fd = client\_sockets[i];

bzero(buffer, sizeof(buffer));

//Check for change in FD

if(FD\_ISSET(fd, &newfds))

{

recv(fd, buffer, sizeof(buffer), 0);

//Check ARP response

if(buffer[0])

{

printf("\nARP Reply received: %s\n", buffer);

count = 0;

k = 0;

for(i = 0; buffer[i]; i++)

{

if(count == 0)

{

DEST\_MAC[k++] = buffer[i];

}

if(buffer[i] == '|')

{

break;

}

}

DEST\_MAC[k] = '\0';

printf("\nSending the packet to: %s\n", DEST\_MAC);

bzero(buffer, sizeof(buffer));

strcat(buffer, SRC\_MAC);

strcat(buffer, "|");

strcat(buffer, SRC\_IP);

strcat(buffer, "|");

strcat(buffer, DEST\_IP);

strcat(buffer, "|");

strcat(buffer, DEST\_MAC);

strcat(buffer, "|");

strcat(buffer, DATA);

send(newfd, buffer, sizeof(buffer), 0);

printf("\nPacket Sent: %s\n", buffer);

}

else

{

close(fd);

client\_sockets[i] = 0;

}

}

}

}

return 0;

}

**Program for Client(same for all the clients):**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#define PORT 8080

int main()

{

struct sockaddr\_in server\_addr;

char buffer[1024], dest[100], SRC\_IP[100], DEST\_IP[100], SRC\_MAC[100], DEST\_MAC[100];

int sockfd;

int i, count, k;

printf("\nEnter the IP Address\t: ");

scanf("%s", dest);

printf("\nEnter the MAC Address\t: ");

scanf("%s", DEST\_MAC);

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if(sockfd < 0)

{

perror("Unable to open socket.\n");

}

bzero(&server\_addr, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

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

server\_addr.sin\_port = htons(PORT);

connect(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

bzero(buffer, sizeof(buffer));

recv(sockfd, buffer, sizeof(buffer), 0);

printf("\nARP Request Received: %s\n", buffer);

count = 0;

k = 0;

for(i = 0; buffer[i]; i++)

{

if(buffer[i] == '|')

{

if(count == 0)

{

SRC\_MAC[k] = '\0';

}

else if(count == 1)

{

SRC\_IP[k] = '\0';

}

else if(count == 3)

{

DEST\_IP[k] = '\0';

}

count++;

k = 0;

}

else if(count == 0)

{

SRC\_MAC[k++] = buffer[i];

}

else if(count == 1)

{

SRC\_IP[k++] = buffer[i];

}

else if(count == 3)

{

DEST\_IP[k++] = buffer[i];

}

}

if(strcmp(dest, DEST\_IP) == 0)

{

bzero(buffer, sizeof(buffer));

printf("\nIP Address matches.\n");

strcat(buffer, DEST\_MAC);

strcat(buffer, "|");

strcat(buffer, DEST\_IP);

strcat(buffer, "|");

strcat(buffer, SRC\_IP);

strcat(buffer, "|");

strcat(buffer, SRC\_MAC);

send(sockfd, buffer, sizeof(buffer), 0);

printf("\nARP Reply Sent: %s\n", buffer);

bzero(buffer, sizeof(buffer));

recv(sockfd, buffer, sizeof(buffer), 0);

printf("\nReceived Packet is: %s\n", buffer);

}

else{

printf("\nIP Address does not match.\n");

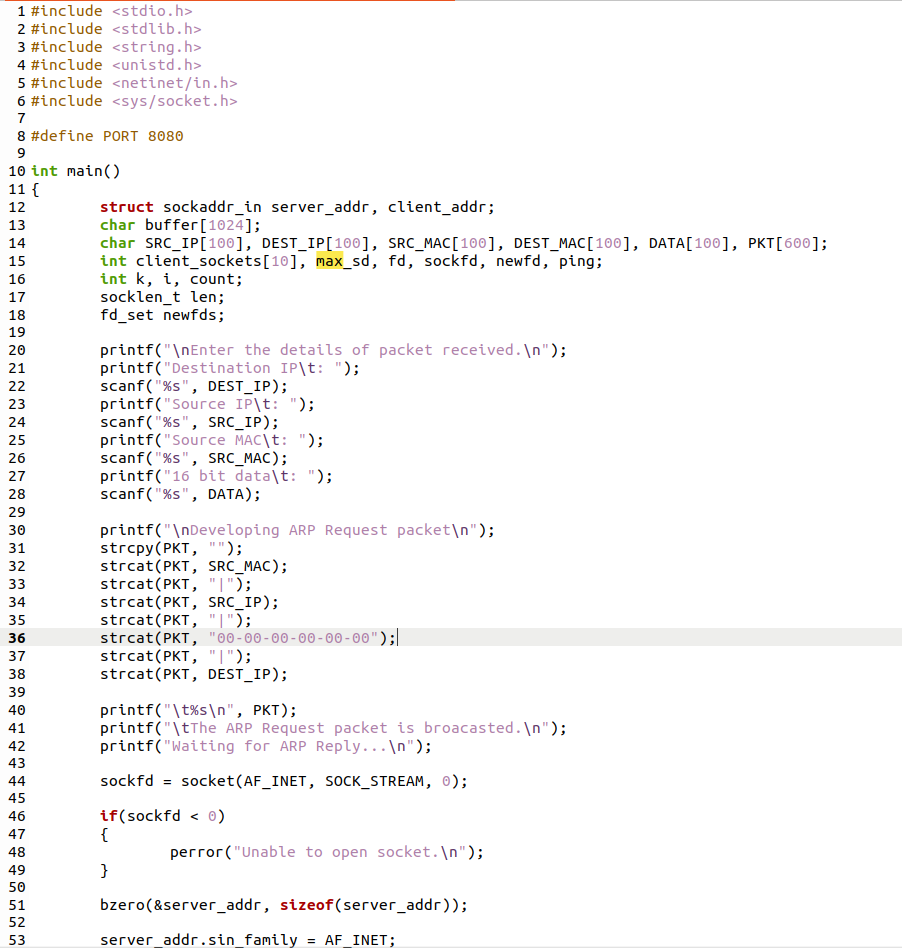
}

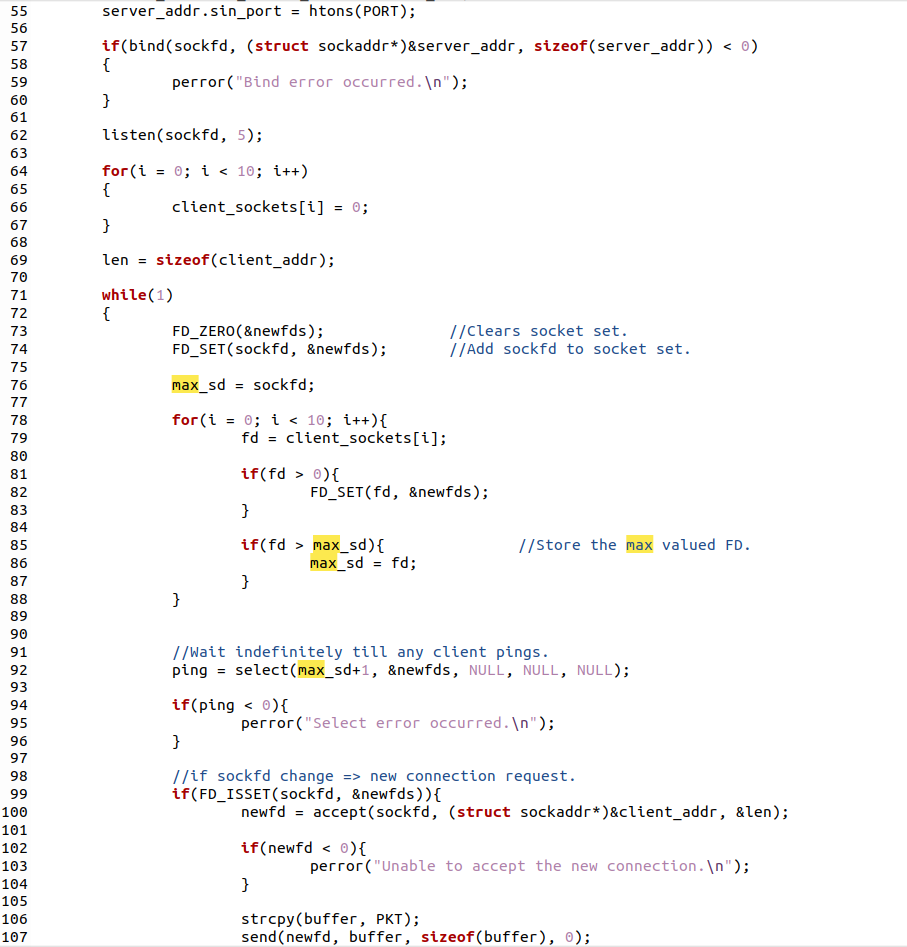
close(sockfd);

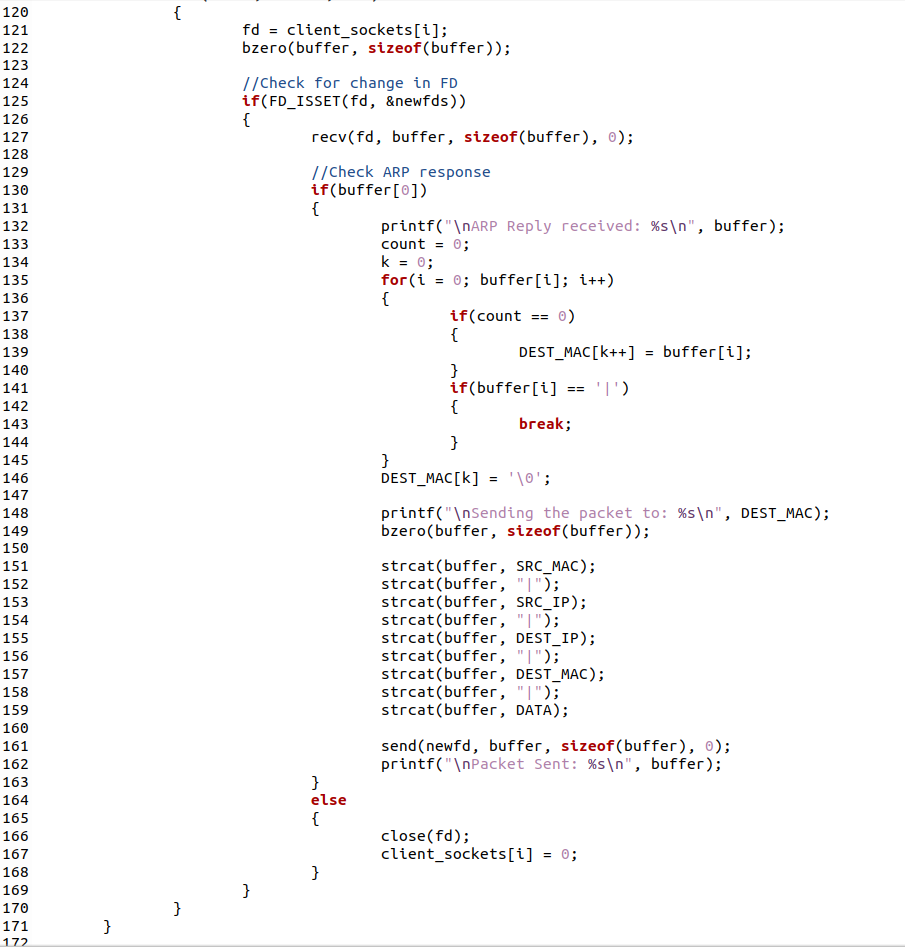
return 0;

}

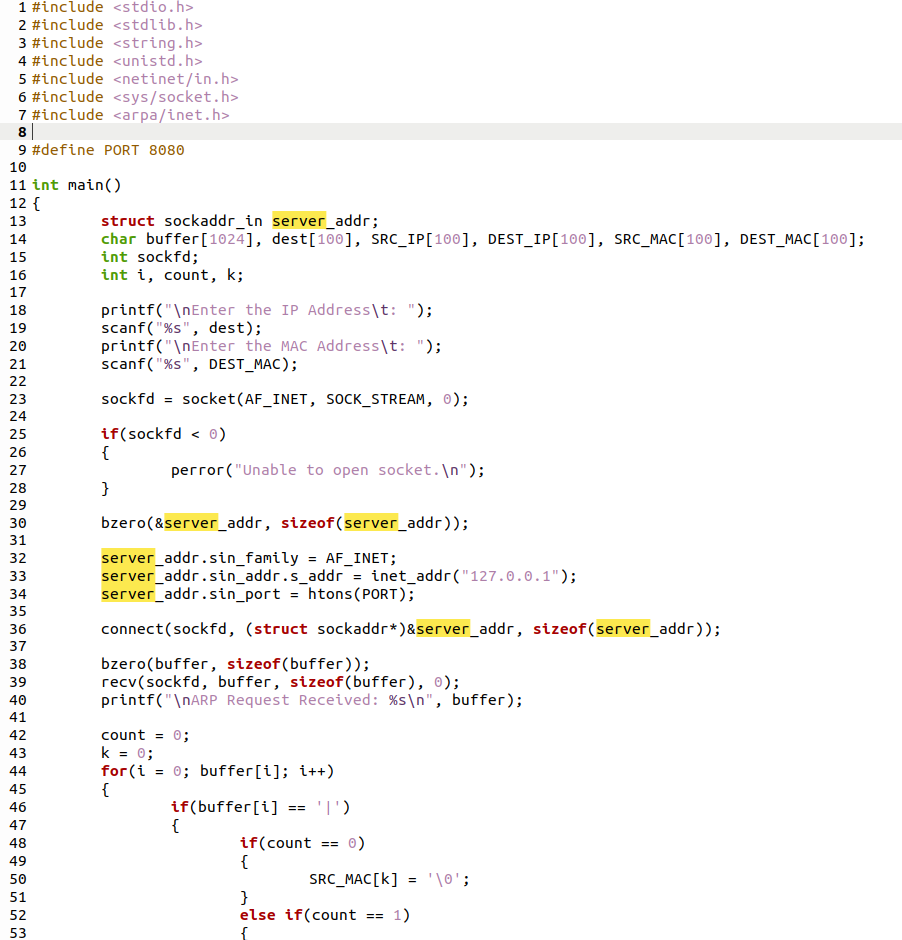
**Screenshot for Server:**

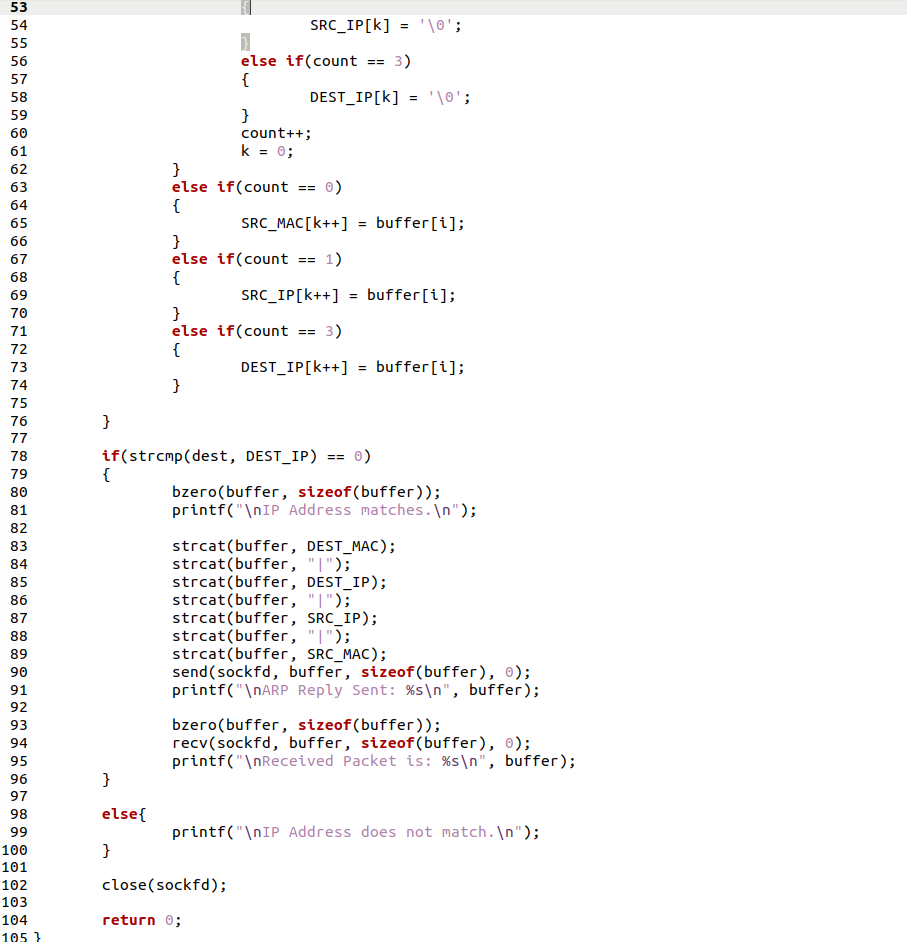
****

****

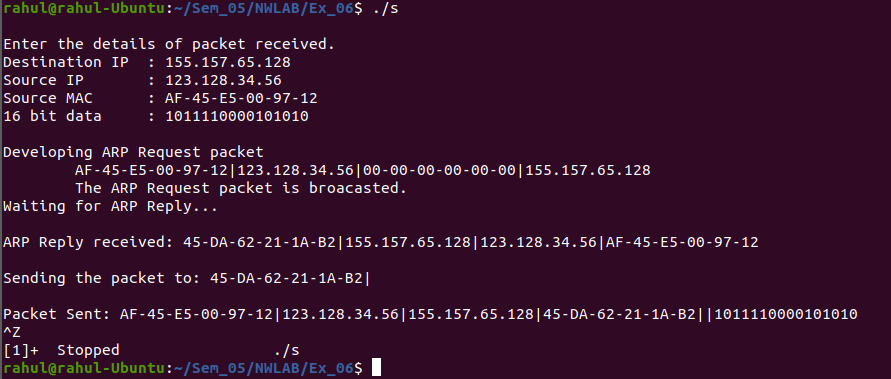
****

**Screenshot for Client:**

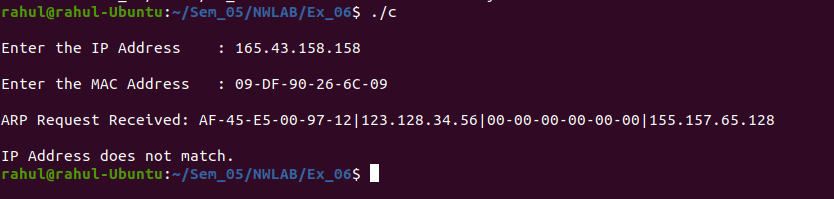
****

****

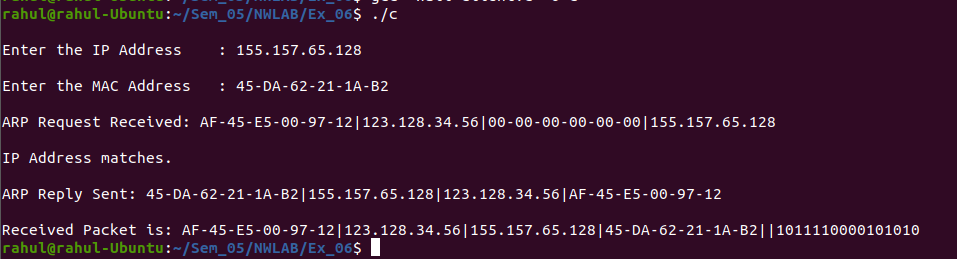
**Server Output:**

****

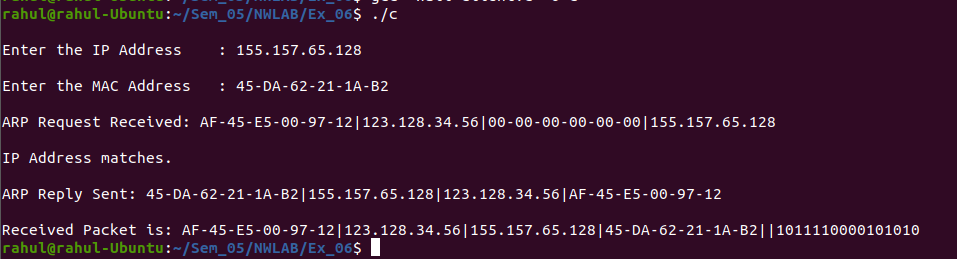
**Client1 Output:**

****

**Client2 Output:**

****

**Client3 Output:**

****

**Learning Outcomes:**

This assignment helped me to

1. Write program for server and client with socket programming.
2. Understand various functions invloved in creating, estabilishing, maintaining, Sending, recieving and termininating the connection between the server and client.
3. Connect multiple clients to the server using select() system call.
4. Understand various functions of ARP protocol.
5. Simulate the functions of ARP using socket programming.