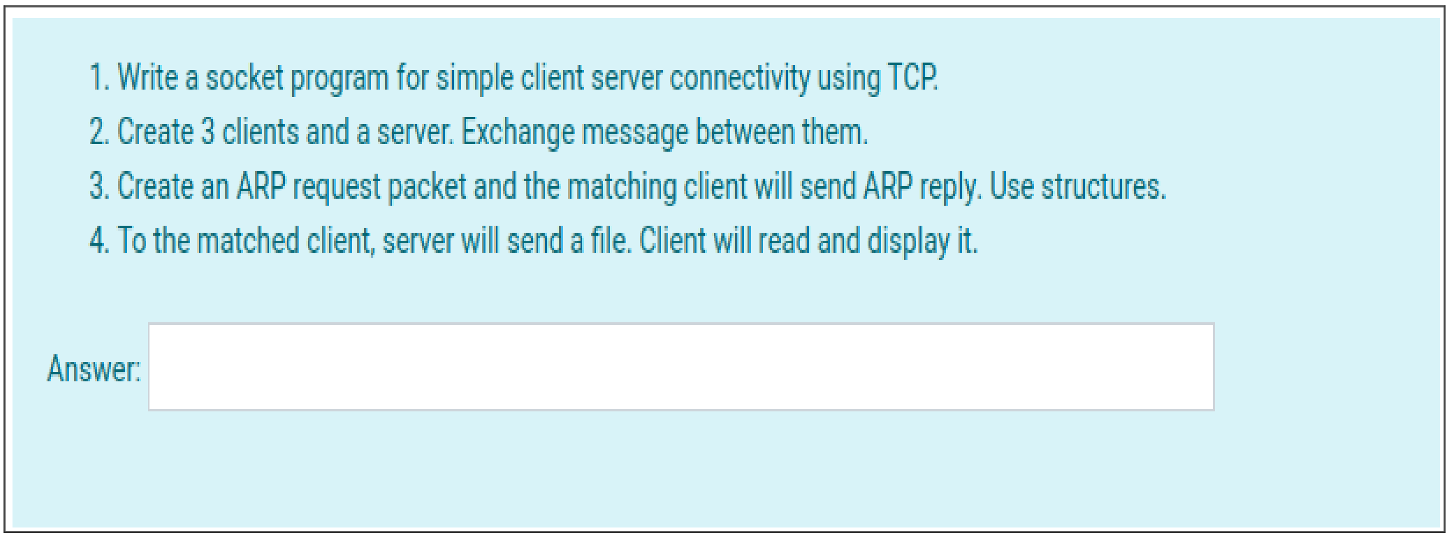
END SEMESTER PRACTICAL EXAM
NETWORKS LAB

Date	:	20 November 2020
Name	:	S. Vishakan
Class	:	CSE – C
Reg. No.	:	18 5001 196

Question:



1. Write a socket program for simple client server connectivity using TCP.
2. Create 3 clients and a server. Exchange message between them.
3. Create an ARP request packet and the matching client will send ARP reply. Use structures.
4. To the matched client, server will send a file. Client will read and display it.

Answer:

Server Program:

```
/*
1. Write a socket program for simple client server connectivity using TCP.
2. Create 3 clients and a server. Exchange message between them.
3. Create an ARP request packet and the matching client will send ARP reply. Use
structures.
4. To the matched client, server will send a file. Client will read and display it.
*/

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

struct ARP_PACKET{
        //ARP Packet Structure
        char SRC_IP[100];
        char DEST_IP[100];
        char SRC_MAC[100];
        char DEST_MAC[100];
        char FILE[100];
        char PKT[600];
};

typedef struct ARP_PACKET arp;

arp makeARPPacket(void);

int main(int argc, char **argv){
        //Usage:        gcc Server.c -o s
        //              ./s
        struct sockaddr_in server, client;
        char buffer[1024];
        int sockfd, newfd;
        int k, i, len, count, child, c_id, n;
        arp packet;

        /*---Taking the server side network details and developing an ARP request packet---*/

        packet = makeARPPacket();
        printf("\nDeveloping ARP Request packet\n");
        printf("\t%s", packet.PKT);
        printf("\n\nThe ARP Request packet is ready to be broacasted.\n");

        /*---Creating the socket and binding it to the server port---*/
```

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);             //IPv4, TCP socket

if(sockfd < 0){
        perror("Unable to open socket.\n");
}
else{
        printf("Socket opened successfully.\n");
}

bzero(&server, sizeof(server));

server.sin_family = AF_INET;                          //IPv4
server.sin_addr.s_addr = inet_addr("127.0.0.1");      //Localhost
server.sin_port = htons(7228);                        //Port Number

if(bind(sockfd, (struct sockaddr*)&server, sizeof(server)) < 0){
        perror("Bind error occurred.\n");
}
else{
        printf("Socket binded to the port successfully.\n");
}

listen(sockfd, 10);

printf("\n------------------------------Server ready------------------------------\n");

len = sizeof(client);

/*---Client-Server communication---*/

while(1){
        //To broadcast the ARP request packet to every connecting client
        newfd = accept(sockfd, (struct sockaddr *)&client, &len);
        child = fork();

        if(child == 0){
                /*---Sending acknowledgement & receiving client ID---*/

                bzero(buffer, sizeof(buffer));
                strcpy(buffer, "You are connected to server.");
                n = send(newfd, buffer, sizeof(buffer), 0);
                n = recv(newfd, &c_id, sizeof(c_id), 0);

                /*---Broadcasting ARP request packet & responding to the ARP reply
                packet, if received---*/

                bzero(buffer, sizeof(buffer));
                strcpy(buffer, packet.PKT);
                printf("\nSending ARP Request Packet to Client %d", c_id);
                n = send(newfd, buffer, sizeof(buffer), 0);
```

```c
            bzero(buffer, sizeof(buffer));
            n = recv(newfd, buffer, sizeof(buffer), 0);

            if(strcmp(buffer, "Does not match!") == 0){
                //ARP reply was not obtained
                printf("\nClient %d did not match with the destination IP
                address!\n", c_id);
            }

            else{
                //ARP reply obtained, send file details to client.
                printf("\nClient %d matched with the destination IP address!",
                c_id);
                printf("\n\nARP Reply received:\n\t%s", buffer);
                printf("\nSending the file details to Client %d...\n", c_id);

                bzero(buffer, sizeof(buffer));
                strcpy(buffer, packet.FILE);
                n = send(newfd, buffer, sizeof(buffer), 0);
            }
        }
    }

    close(newfd);
    close(sockfd);
    return 0;
}

arp makeARPPacket(void){
    //Creates the ARP Request packet
    arp packet;

    printf("\nEnter the details of packet received.\n");
    printf("Enter the Source IP\t\t: ");
    scanf(" %s", packet.SRC_IP);
    printf("Enter the Source MAC\t\t: ");
    scanf(" %s", packet.SRC_MAC);
    printf("Enter the Destination IP\t: ");
    scanf(" %s", packet.DEST_IP);
    printf("Enter the File Name\t\t: ");
    scanf(" %s", packet.FILE);

    //Creating the packet
    strcpy(packet.PKT, packet.SRC_IP);
    strcat(packet.PKT, "|");
    strcat(packet.PKT, packet.SRC_MAC);
    strcat(packet.PKT, "|");
    strcat(packet.PKT, packet.DEST_IP);

    return packet;
}
```

Client Program:

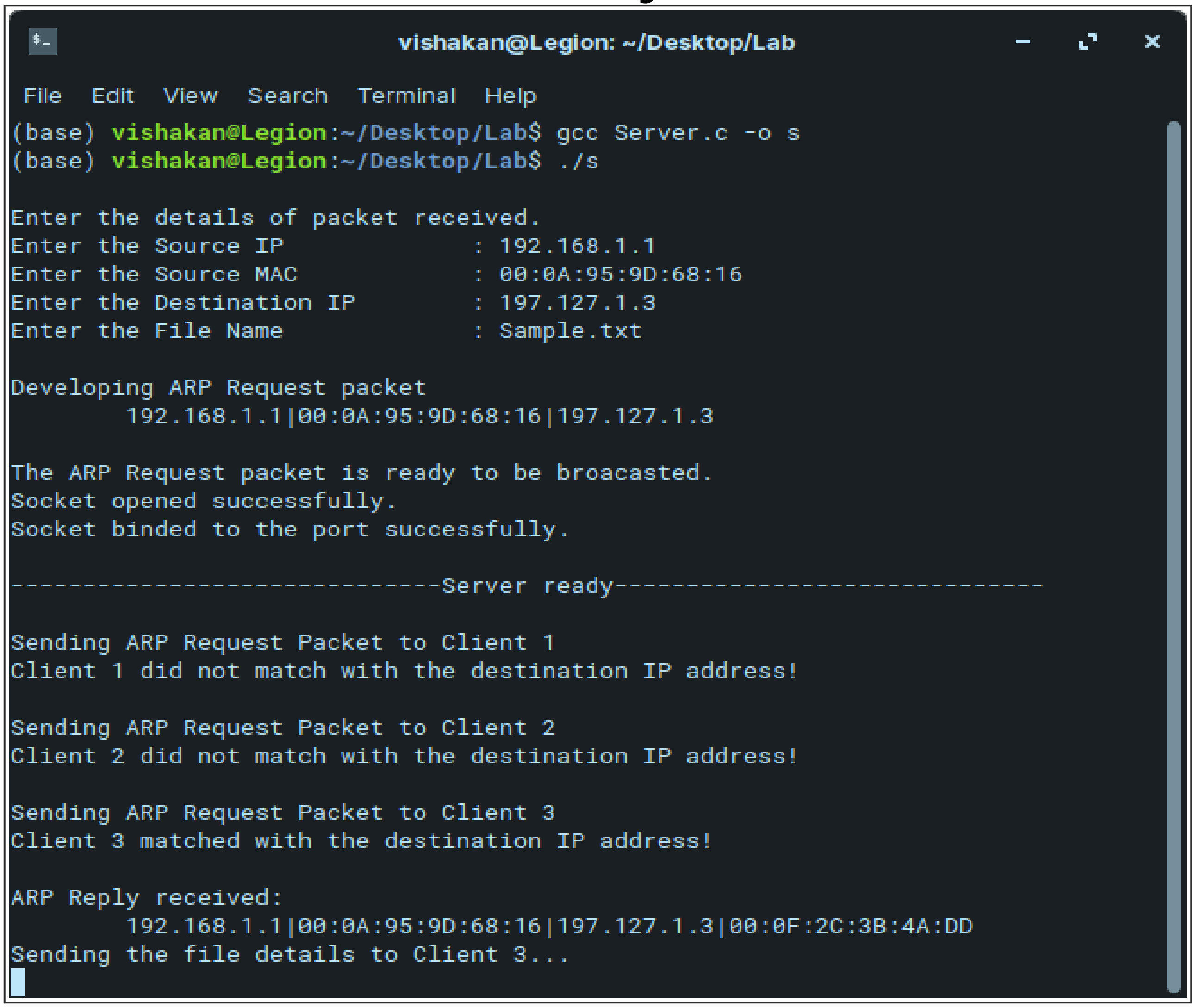
```
/*
1. Write a socket program for simple client server connectivity using TCP.
2. Create 3 clients and a server. Exchange message between them.
3. Create an ARP request packet and the matching client will send ARP reply. Use
structures.
4. To the matched client, server will send a file. Client will read and display it.
*/

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

struct ARP_PACKET{
	//ARP Packet Structure
	char SRC_IP[100];
	char DEST_IP[100];
	char SRC_MAC[100];
	char DEST_MAC[100];
	char FILE[100];
	char PKT[600];
};

typedef struct ARP_PACKET arp;

void fileDumper(char *fname);

int main(int argc, char **argv){
	//Usage:		gcc Client.c -o c
	//			./c <client_id>
	struct sockaddr_in server, client;
	char buffer[1024];
	int sockfd, newfd;
	int len, i, count, k, c_id;
	arp packet;

	/*---Taking the client side network details---*/

	c_id = atoi(argv[1]);					//Client ID
	printf("\nEnter the IP Address\t: ");			//Client IP
	scanf("%s", packet.DEST_IP);
	printf("\nEnter the MAC Address\t: ");			//Client MAC
	scanf("%s", packet.DEST_MAC);
```

```
/*---Creating the socket and connecting it to the server port---*/

sockfd = socket(AF_INET, SOCK_STREAM, 0); //IPv4, TCP socket

if(sockfd < 0){
	perror("Unable to open socket.\n");
	exit(1);
}
else{
	printf("\nSocket opened successfully.\n");
}

bzero(&server, sizeof(server));

server.sin_family = AF_INET;                              //IPv4
server.sin_addr.s_addr = inet_addr("127.0.0.1");          //Localhost
server.sin_port = htons(7228);                            //Port number

connect(sockfd, (struct sockaddr*)&server, sizeof(server));
len = sizeof(client);

printf("\n-----------------------------Client ready-----------------------------\n");

/*---Sending client identifier to server on connection---*/

bzero(buffer, sizeof(buffer));
recv(sockfd, buffer, sizeof(buffer), 0);
send(sockfd, &c_id, sizeof(c_id), 0);
printf("\nServer ACK: %s\n", buffer);

/*---Receiving ARP request packet & responding to it---*/

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++){
	//Getting the IP from the received ARP packet
	if(count == 2){
		packet.SRC_IP[k++] = buffer[i];
	}
	if(buffer[i] == '|'){
		count++;
	}
}

packet.SRC_IP[k] = '\0';
```

```c
	if(strcmp(packet.SRC_IP, packet.DEST_IP) == 0){
		//Comparing the IP addresses, case for matched destination IP address
		printf("\nDestination IP Address matches.\n");
		strcat(buffer, "|");
		strcat(buffer, packet.DEST_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 File is: %s\n", buffer);
		fileDumper(buffer);
	}

	else{
		//Case for unmatched destination IP address
		printf("\nDestination IP Address does not match.\n");
		bzero(buffer, sizeof(buffer));
		strcpy(buffer, "Does not match!");
		send(sockfd, buffer, sizeof(buffer), 0);
	}

	close(sockfd);

	return 0;
}

void fileDumper(char *fname){
	//To print the contents of the file specified by the server
	int src, n;
	char buffer[1024];
	src = open(fname, O_RDONLY);		//opening a file in Read-Only mode

	if(src == -1){
		printf("\nSpecified file does not exist.\n");
	}

	else{
		printf("\nFile contents:\n");

		while((n = read(src, buffer, sizeof(buffer))) != 0){
			buffer[n] = '\0';		//To terminate the string
			printf("%s", buffer);
		}
		printf("\n");
		close(src);
	}
}
```

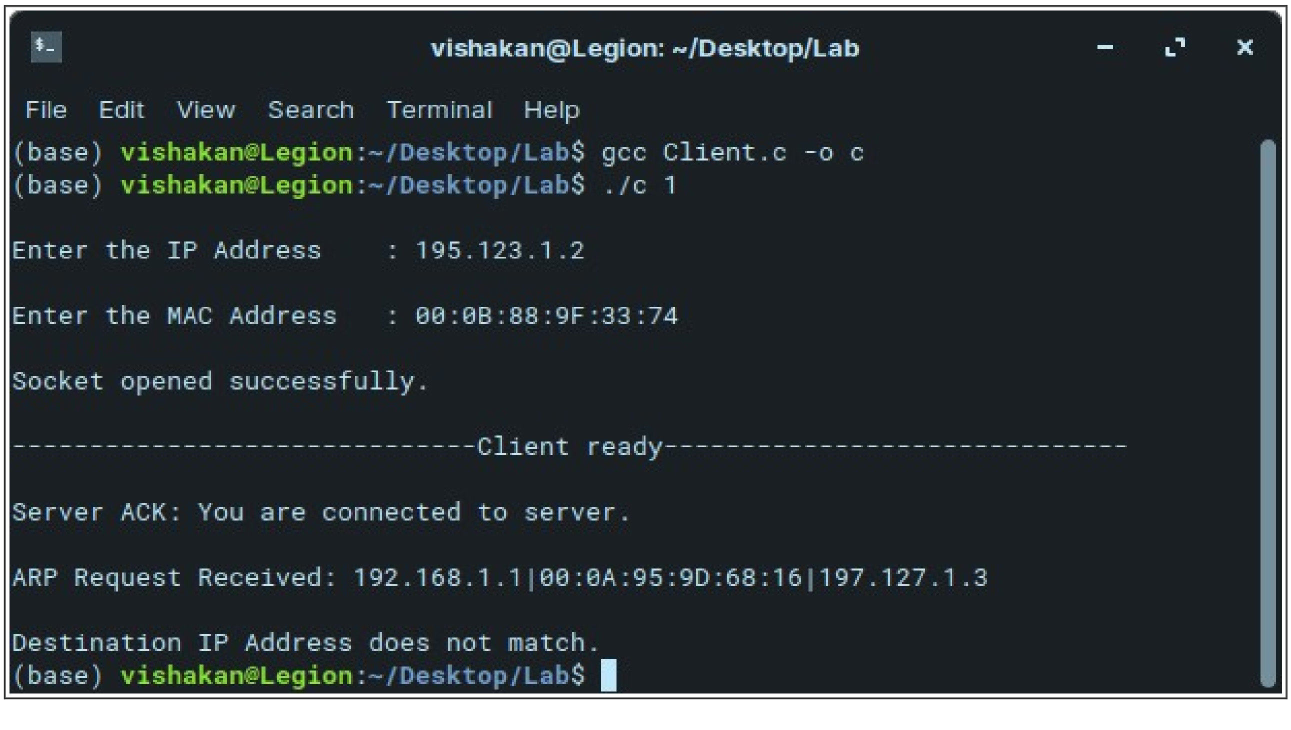
Output:

Server Program


vishakan@Legion: ~/Desktop/Lab
File Edit View Search Terminal Help

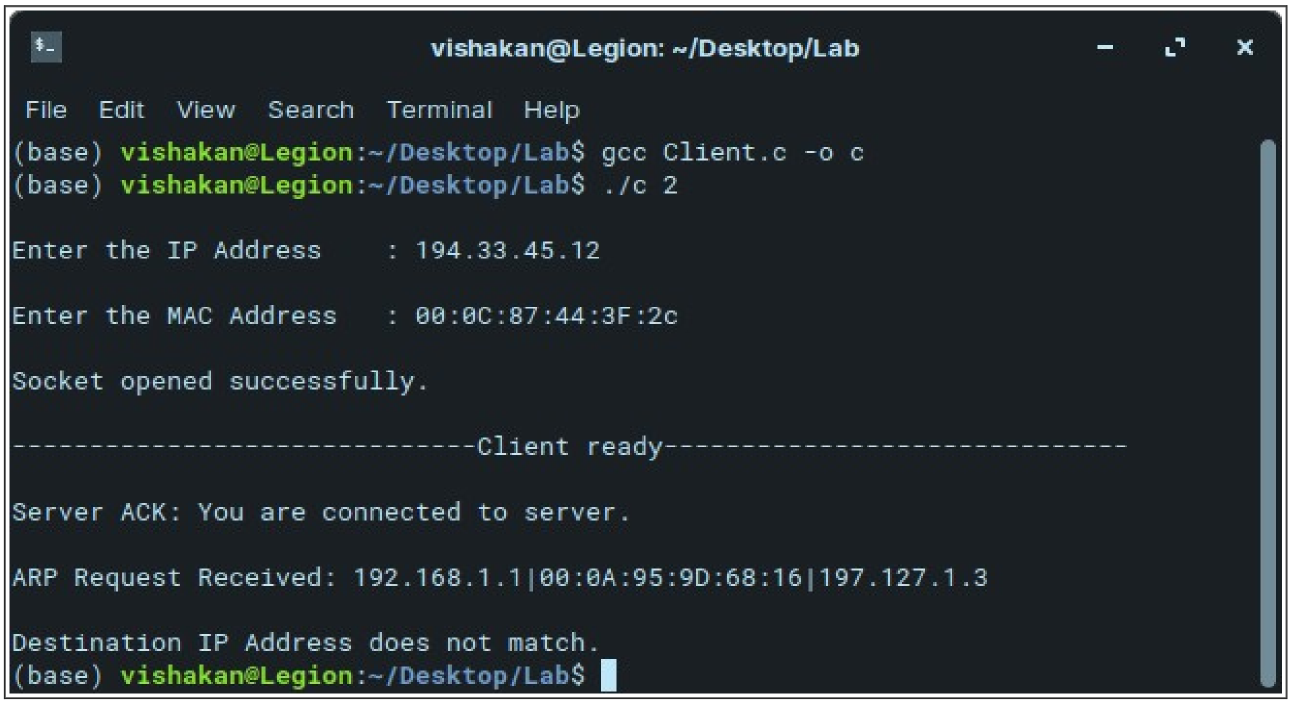
```
(base) vishakan@Legion:~/Desktop/Lab$ gcc Server.c -o s
(base) vishakan@Legion:~/Desktop/Lab$ ./s

Enter the details of packet received.
Enter the Source IP             : 192.168.1.1
Enter the Source MAC            : 00:0A:95:9D:68:16
Enter the Destination IP        : 197.127.1.3
Enter the File Name             : Sample.txt

Developing ARP Request packet
        192.168.1.1|00:0A:95:9D:68:16|197.127.1.3

The ARP Request packet is ready to be broacasted.
Socket opened successfully.
Socket binded to the port successfully.

------------------------------Server ready------------------------------

Sending ARP Request Packet to Client 1
Client 1 did not match with the destination IP address!

Sending ARP Request Packet to Client 2
Client 2 did not match with the destination IP address!

Sending ARP Request Packet to Client 3
Client 3 matched with the destination IP address!

ARP Reply received:
        192.168.1.1|00:0A:95:9D:68:16|197.127.1.3|00:0F:2C:3B:4A:DD
Sending the file details to Client 3...
```

Client 1 - Not the destination client


vishakan@Legion: ~/Desktop/Lab
File Edit View Search Terminal Help

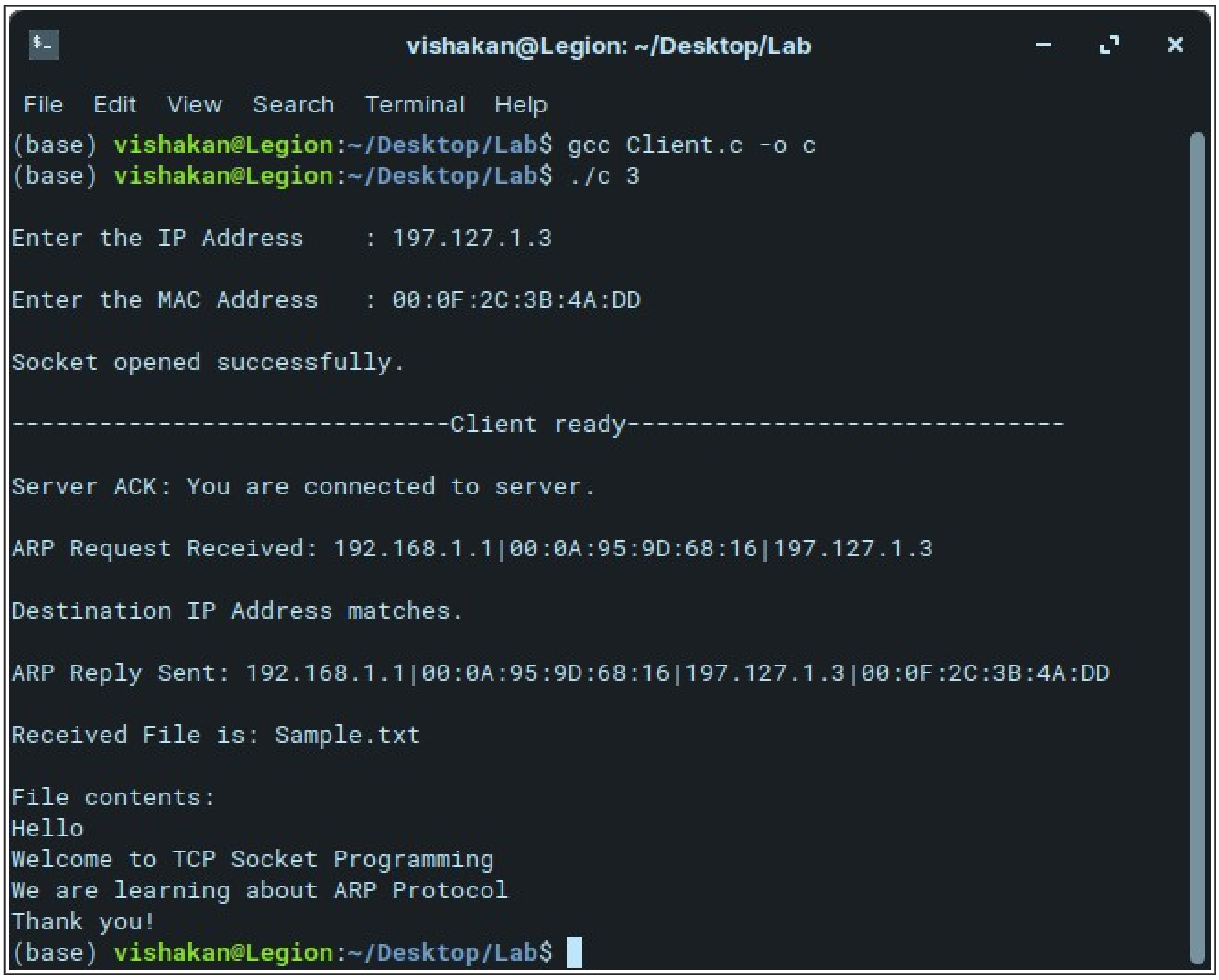
```
(base) vishakan@Legion:~/Desktop/Lab$ gcc Client.c -o c
(base) vishakan@Legion:~/Desktop/Lab$ ./c 1

Enter the IP Address    : 195.123.1.2

Enter the MAC Address   : 00:0B:88:9F:33:74

Socket opened successfully.

------------------------------Client ready------------------------------

Server ACK: You are connected to server.

ARP Request Received: 192.168.1.1|00:0A:95:9D:68:16|197.127.1.3

Destination IP Address does not match.
(base) vishakan@Legion:~/Desktop/Lab$
```

Client 2 - Not the destination client


vishakan@Legion: ~/Desktop/Lab
File Edit View Search Terminal Help

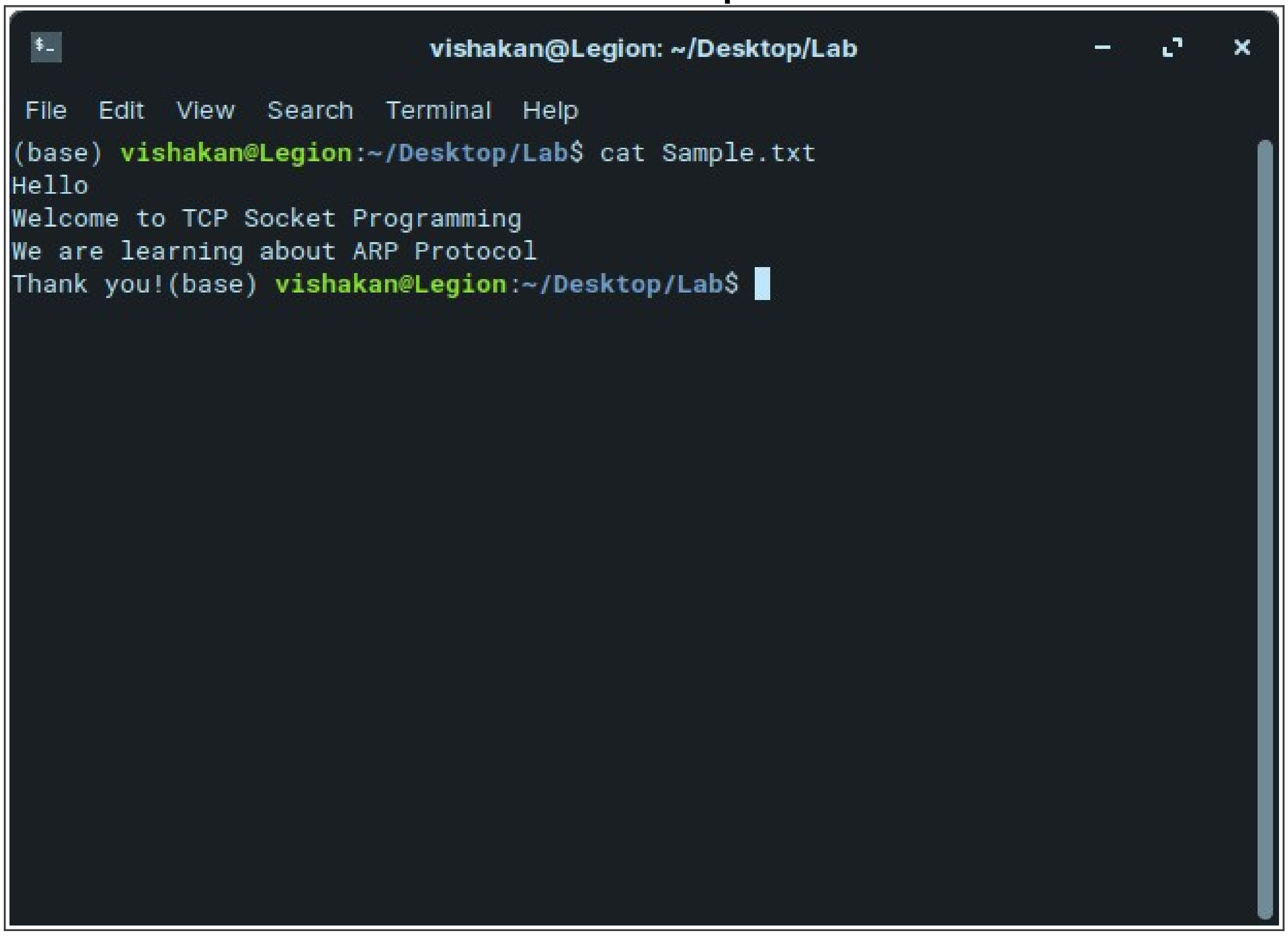
```
(base) vishakan@Legion:~/Desktop/Lab$ gcc Client.c -o c
(base) vishakan@Legion:~/Desktop/Lab$ ./c 2

Enter the IP Address    : 194.33.45.12

Enter the MAC Address   : 00:0C:87:44:3F:2c

Socket opened successfully.

------------------------------Client ready------------------------------

Server ACK: You are connected to server.

ARP Request Received: 192.168.1.1|00:0A:95:9D:68:16|197.127.1.3

Destination IP Address does not match.
(base) vishakan@Legion:~/Desktop/Lab$ 
```

Client 3 – Intended destination client


vishakan@Legion: ~/Desktop/Lab
File Edit View Search Terminal Help

```
(base) vishakan@Legion:~/Desktop/Lab$ gcc Client.c -o c
(base) vishakan@Legion:~/Desktop/Lab$ ./c 3

Enter the IP Address    : 197.127.1.3

Enter the MAC Address   : 00:0F:2C:3B:4A:DD

Socket opened successfully.

------------------------------Client ready------------------------------

Server ACK: You are connected to server.

ARP Request Received: 192.168.1.1|00:0A:95:9D:68:16|197.127.1.3

Destination IP Address matches.

ARP Reply Sent: 192.168.1.1|00:0A:95:9D:68:16|197.127.1.3|00:0F:2C:3B:4A:DD

Received File is: Sample.txt

File contents:
Hello
Welcome to TCP Socket Programming
We are learning about ARP Protocol
Thank you!
(base) vishakan@Legion:~/Desktop/Lab$ 
```

Contents of Sample.txt


vishakan@Legion: ~/Desktop/Lab
File Edit View Search Terminal Help

```
(base) vishakan@Legion:~/Desktop/Lab$ cat Sample.txt
Hello
Welcome to TCP Socket Programming
We are learning about ARP Protocol
Thank you!(base) vishakan@Legion:~/Desktop/Lab$ 
```

Result:

A simple client-server connectivity based TCP socket program was developed and expanded to the multi-client, single-server paradigm. Then, using the multi-client single-server program, a simulation of the ARP protocol was incrementally developed, using a structure for the ARP packet.

To the client that sent the ARP reply back to the server(which sent an ARP request packet before), a file was sent by the server. The file was opened in the matched client program and its contents were read and displayed in the terminal by the client.

The output was verified and the simulation was successful.