Pradeep KM
3122225001092

Exercise 5: ARP and RARP

ARP

Server algorithm

1. Create UDP socket for broadcasting.
2. Set socket option for broadcasting.
3. Broadcast message with source IP, source MAC, and destination IP.
4. Wait for incoming UDP message.
5. If destination IP matches, create TCP socket.
6. Connect to the client via TCP.
7. Send data (ARP reply) to the client via TCP.
8. Close TCP and UDP sockets.

Client algorithm

1. Create UDP socket to listen for broadcast.
2. Wait for UDP broadcast message.
3. Extract source IP, source MAC, and destination IP from the message.
4. If destination IP matches, create TCP socket.
5. Connect to the server via TCP.
6. Receive ARP reply from server.
7. Close the TCP and UDP sockets.

server.c

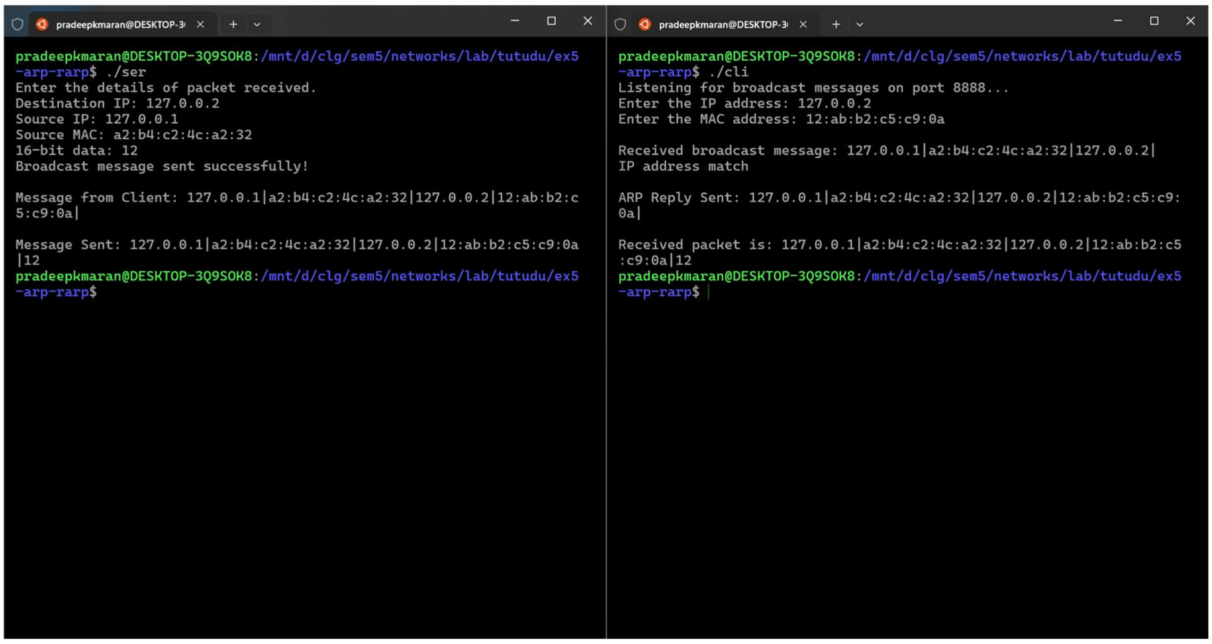
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define BROADCAST_IP "255.255.255.255"
#define BROADCAST_PORT 8888
#define MESSAGE "This is a broadcast message!"

typedef struct {
   char src_ip[16];
   char src_mac[18];
   char dest_ip[16];
   char dest_mac[18];
   char data[17];
} Packet;

int main() {
   int sockfd;
   struct sockaddr_in broadcast_addr;
```

```
    int broadcast_enable = 1;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &broadcast_enable,
sizeof(broadcast_enable));

    memset(&broadcast_addr, 0, sizeof(broadcast_addr));
    broadcast_addr.sin_family = AF_INET;
    broadcast_addr.sin_port = htons(BROADCAST_PORT);
    broadcast_addr.sin_addr.s_addr = inet_addr(BROADCAST_IP);

    Packet packet;
    printf("Enter the details of packet received.\n");
    printf("Destination IP: ");
    scanf("%s", packet.dest_ip);
    printf("Source IP: ");
    scanf("%s", packet.src_ip);
    printf("Source MAC: ");
    scanf("%s", packet.src_mac);
    printf("16-bit data: ");
    scanf("%s", packet.data);

    char msg[1000];
    strcpy(msg, packet.src_ip);
    strcat(msg, "|");
    strcat(msg, packet.src_mac);
    strcat(msg, "|");
    strcat(msg, packet.dest_ip);
    strcat(msg, "|");

    sendto(sockfd, msg, strlen(msg), 0, (struct sockaddr *)&broadcast_addr, sizeof(broadcast_addr));

    printf("Broadcast message sent successfully!\n");

    int len;
    int sockfd1, newfd, n;
    struct sockaddr_in servaddr, cliaddr;
    char buff[1024];
    char str[1000];

    sockfd1 = socket(AF_INET, SOCK_STREAM, 0);
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(7228);
    bind(sockfd1, (struct sockaddr *)&servaddr, sizeof(servaddr));

    listen(sockfd1, 2);

    len = sizeof(cliaddr);
    newfd = accept(sockfd1, (struct sockaddr *)&cliaddr, &len);

    n = read(newfd, buff, sizeof(buff));
    printf("\nMessage from Client: %s\n", buff);

    char newstr[1000];
    strcpy(newstr, buff);
    strcat(newstr, packet.data);
    printf("\nMessage Sent: %s\n", newstr);
    n = write(newfd, newstr, sizeof(newstr));

    close(sockfd1);
    close(newfd);

    close(sockfd);

    return 0;
}
```

client.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define LISTEN_PORT 8888
#define BUFFER_SIZE 1024
#define PORT 8888

int main() {
    int sockfd;
    struct sockaddr_in recv_addr, cliaddr;
    char buffer[BUFFER_SIZE];
    socklen_t addr_len = sizeof(recv_addr);
    char client_ip[16], client_mac[18];

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&cliaddr, 0, sizeof(cliaddr));
    cliaddr.sin_family = AF_INET;
    cliaddr.sin_addr.s_addr = INADDR_ANY;
    cliaddr.sin_port = htons(PORT);

    memset(&recv_addr, 0, sizeof(recv_addr));
    recv_addr.sin_family = AF_INET;
    recv_addr.sin_port = htons(LISTEN_PORT);
    recv_addr.sin_addr.s_addr = INADDR_ANY;
    bind(sockfd, (struct sockaddr *)&recv_addr, sizeof(recv_addr));
    printf("Listening for broadcast messages on port %d...\n", LISTEN_PORT);

    printf("Enter the IP address: ");
    scanf("%s", client_ip);
    printf("Enter the MAC address: ");
    scanf("%s", client_mac);

    char src_ip[16], src_mac[18], dest_ip[16];

    while (1) {
        int recv_len = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr *)&recv_addr,
&addr_len);

        if (recv_len > 0) {
            buffer[recv_len] = '\0';
            printf("\nReceived broadcast message: %s\n", buffer);

            sscanf(buffer, "%[^|]|%[^|]|%[^|]", src_ip, src_mac, dest_ip);

            if (strcmp(dest_ip, client_ip) == 0) {
                printf("IP address match\n");

                int len;
                int sockfd1, n, newfd;
                struct sockaddr_in servaddr;
                char str[1000];
                char buff[1024];
                char newbuff[1024];

                sockfd1 = socket(AF_INET, SOCK_STREAM, 0);
                if (sockfd1 < 0)
                    perror("\nCannot create socket\n");

                bzero(&servaddr, sizeof(servaddr));
                servaddr.sin_family = AF_INET;
                servaddr.sin_addr.s_addr = inet_addr(src_ip);
                servaddr.sin_port = htons(7228);

                connect(sockfd1, (struct sockaddr *)&servaddr, sizeof(servaddr));

                snprintf(buffer, sizeof(buffer), "%s|%s|%s|%s|", src_ip, src_mac, dest_ip, client_mac);
                n = write(sockfd1, buffer, sizeof(buffer));
                printf("\nARP Reply Sent: %s\n", buffer);

                n = read(sockfd1, newbuff, sizeof(newbuff));
                printf("\nReceived packet is: %s \n", newbuff);

                close(sockfd1);
                close(newfd);
            } else {
                printf("IP address not matched\n");
            }
            break;
        } else {
            break;
        }
    }
    close(sockfd);

    return 0;
}
```

Output



```
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp$ ./ser
Enter the details of packet received.
Destination IP: 127.0.0.2
Source IP: 127.0.0.1
Source MAC: a2:b4:c2:4c:a2:32
16-bit data: 12
Broadcast message sent successfully!

Message from Client: 127.0.0.1|a2:b4:c2:4c:a2:32|127.0.0.2|12:ab:b2:c
5:c9:0a|

Message Sent: 127.0.0.1|a2:b4:c2:4c:a2:32|127.0.0.2|12:ab:b2:c5:c9:0a
|12
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp$
```

```
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp$ ./cli
Listening for broadcast messages on port 8888...
Enter the IP address: 127.0.0.2
Enter the MAC address: 12:ab:b2:c5:c9:0a

Received broadcast message: 127.0.0.1|a2:b4:c2:4c:a2:32|127.0.0.2|
IP address match

ARP Reply Sent: 127.0.0.1|a2:b4:c2:4c:a2:32|127.0.0.2|12:ab:b2:c5:c9:
0a|

Received packet is: 127.0.0.1|a2:b4:c2:4c:a2:32|127.0.0.2|12:ab:b2:c5
:c9:0a|12
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp$
```

Pradeep KM
3122225001092

RARP

Server algorithm

1. Create UDP socket.
2. Bind to UDP port.
3. Wait for incoming message.
4. Receive MAC address from client.
5. Check MAC in the IP mapping.
6. If MAC found, prepare corresponding IP.
7. Send IP back to the client.
8. If MAC not found, send default IP.

Client algorithm

1. Create UDP socket.
2. Prepare MAC address to send.
3. Send MAC address to the server.
4. Wait for the server's response.
5. Receive IP address from server.
6. Display the received IP.
7. Close the socket.

server.c

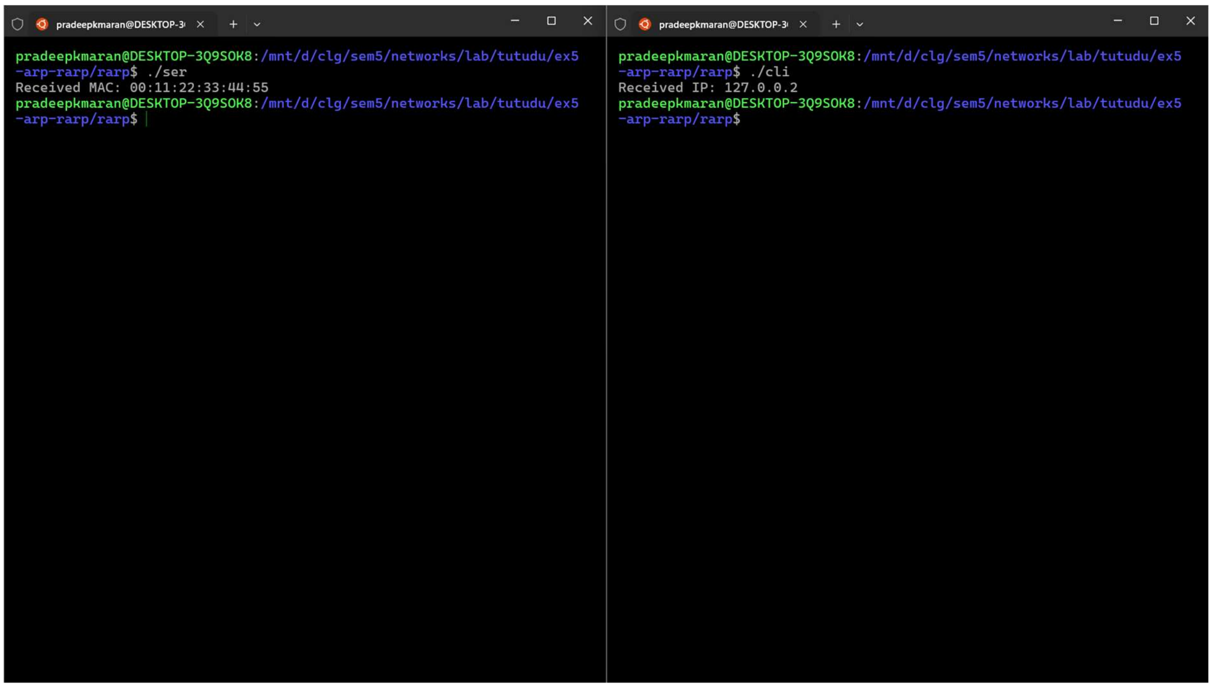
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define UDP_PORT 8888
#define BUFFER_SIZE 1024

typedef struct {
   char mac[18];
   char ip[16];
} Mac_Ip_Map;

Mac_Ip_Map mappings[] = {
   {"00:11:22:33:44:55", "127.0.0.2"},
   {"11:22:33:44:55:66", "127.0.0.3"},
   {"22:33:44:55:66:77", "127.0.0.4"},
   {"33:44:55:66:77:88", "127.0.0.5"}
};

int find_ip_for_mac(const char mac[]) {
   for (int i = 0; i < 4; i++) {
      if (strcmp(mac, mappings[i].mac) == 0) {
         return i;
      }
```

```
    }
    return -1;
}

int main() {
    int udp_sock, tcp_sock, client_sock;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char client_mac[18];
    char client_ip[16];

    udp_sock = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(UDP_PORT);

    bind(udp_sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
    recvfrom(udp_sock, client_mac, sizeof(client_mac), MSG_WAITALL, (struct sockaddr*)
&client_addr, &addr_len);
    client_mac[strlen(client_mac)] = '\0';
    printf("Received MAC: %s\n", client_mac);

    int index = find_ip_for_mac(client_mac);
    if (index != -1) {
        strcpy(client_ip, mappings[index].ip);
    } else {
        strcpy(client_ip, "0.0.0.0");
    }

    sendto(udp_sock, client_ip, strlen(client_ip) + 1, 0, (struct sockaddr*)&client_addr, addr_len);
    close(udp_sock);

    return 0;
}
```

client.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define UDP_PORT 8888
#define BUFFER_SIZE 1024

int main() {
    int udp_sock;
    struct sockaddr_in server_addr;
    char client_mac[18] = "00:11:22:33:44:55";
    char server_ip[16];
    int addr_len = sizeof(server_addr);

    udp_sock = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&server_addr, 0, addr_len);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(UDP_PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    sendto(udp_sock, client_mac, strlen(client_mac) + 1, MSG_CONFIRM, (struct
sockaddr*)&server_addr, addr_len);

    recvfrom(udp_sock, server_ip, sizeof(server_ip), MSG_WAITALL, (struct
sockaddr*)&server_addr, &addr_len);

    printf("Received IP: %s\n", server_ip);

    close(udp_sock);
    return 0;
}
```

Output



```
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp/rarp$ ./ser
Received MAC: 00:11:22:33:44:55
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp/rarp$
```

```
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp/rarp$ ./cli
Received IP: 127.0.0.2
pradeepkmaran@DESKTOP-3Q9SOK8:/mnt/d/clg/sem5/networks/lab/tutudu/ex5
-arp-rarp/rarp$
```