ADDRESS RESOLUTION PROTOCOL (ARP)

Simulate ARP using socket programming.

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]; // 16-bit data
} Packet;

int main() {
    int sockfd;
    struct sockaddr_in broadcast_addr;
    int broadcast_enable = 1;

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Set socket options to allow broadcast
    if (setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST,
&broadcast_enable,
                   sizeof(broadcast_enable)) < 0) {
        perror("setsockopt failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    // Define broadcast address
```

```
    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);

    // Input packet details
    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, "|");

    // Send broadcast message
    if (sendto(sockfd, msg, strlen(msg), 0, (struct
sockaddr *)&broadcast_addr, sizeof(broadcast_addr)) < 0)
{
        perror("sendto failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

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

    // Listen for reply
    int len;
    int sockfd1, newfd, n;
    struct sockaddr_in servaddr, cliaddr;
    char buff[1024];
```

```
    char str[1000];

    // socket
    sockfd1 = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd1 < 0)
        perror("cannot create socket");

    // bind
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(7228);
    if (bind(sockfd1, (struct sockaddr *)&servaddr,
sizeof(servaddr)) < 0)
        perror("Bind error");

    // listen
    listen(sockfd1, 2);

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

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

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

    // close
    close(sockfd1);
    close(newfd);

    // Close socket
    close(sockfd);

    return 0;
}
```

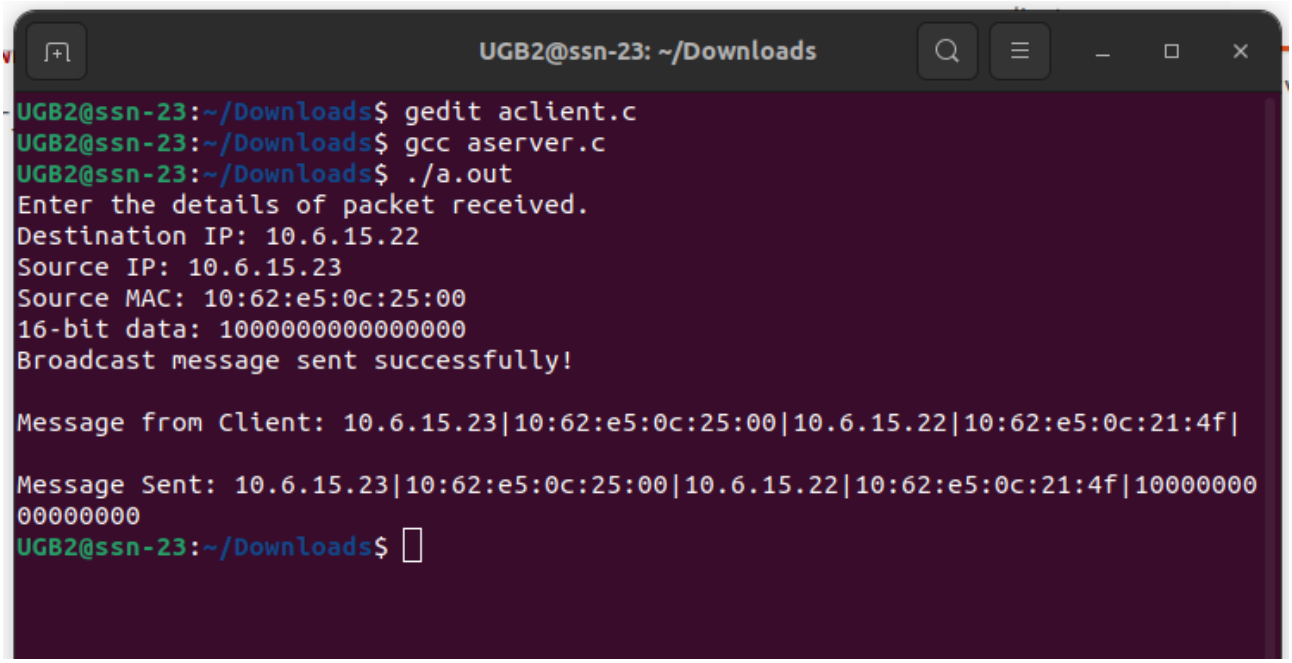
client.c

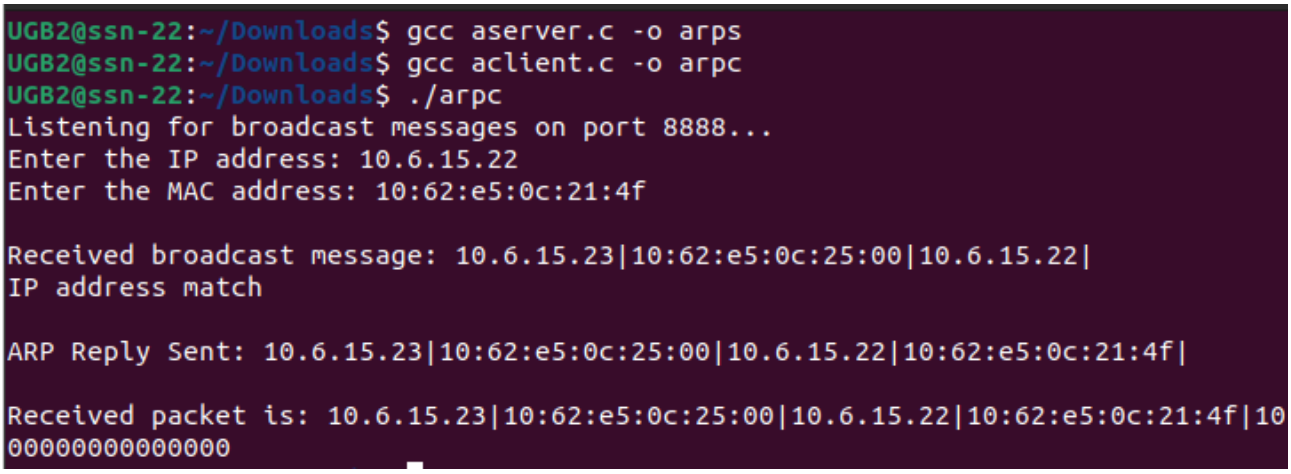
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#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];

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

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

    // Bind to 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;

    if (bind(sockfd, (struct sockaddr *)&recv_addr,
sizeof(recv_addr)) < 0) {
        perror("bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }
```

```
    printf("Listening for broadcast messages on port
%d...\n", LISTEN_PORT);

    // Input client's own IP and MAC
    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];

    // Receive message
    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'; // Null-terminate
the received data
            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));

                // Send ARP reply
                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 {
            perror("recvfrom failed");
            return 0;
        }
    }

    // Close socket
    close(sockfd);

    return 0;
}
```

Output:


UGB2@ssn-23: ~/Downloads
UGB2@ssn-23:~/Downloads$ gedit aclient.c
UGB2@ssn-23:~/Downloads$ gcc aserver.c
UGB2@ssn-23:~/Downloads$ ./a.out
Enter the details of packet received.
Destination IP: 10.6.15.22
Source IP: 10.6.15.23
Source MAC: 10:62:e5:0c:25:00
16-bit data: 1000000000000000
Broadcast message sent successfully!
Message from Client: 10.6.15.23|10:62:e5:0c:25:00|10.6.15.22|10:62:e5:0c:21:4f|
Message Sent: 10.6.15.23|10:62:e5:0c:25:00|10.6.15.22|10:62:e5:0c:21:4f|1000000000000000
UGB2@ssn-23:~/Downloads$


UGB2@ssn-22:~/Downloads$ gcc aserver.c -o arps
UGB2@ssn-22:~/Downloads$ gcc aclient.c -o arpc
UGB2@ssn-22:~/Downloads$ ./arpc
Listening for broadcast messages on port 8888...
Enter the IP address: 10.6.15.22
Enter the MAC address: 10:62:e5:0c:21:4f
Received broadcast message: 10.6.15.23|10:62:e5:0c:25:00|10.6.15.22|
IP address match
ARP Reply Sent: 10.6.15.23|10:62:e5:0c:25:00|10.6.15.22|10:62:e5:0c:21:4f|
Received packet is: 10.6.15.23|10:62:e5:0c:25:00|10.6.15.22|10:62:e5:0c:21:4f|1000000000000000