
Assignment 4 – Address Resolution Protocol (ARP)

Date: 09/09/2022

Aim:

To simulate ARP using socket programming.

Algorithm:

Server:

1. Start
2. Get the host's or routers' IP address and MAC address.
3. Listen for any number of clients (for broadcasting purpose).
4. Enter the packet details received from a host or its own packet to send to a destination. The details are:
    a. Source IP address
    b. Source MAC address
    c. Destination IP address
    d. 16-bit data
5. Develop an ARP Request packet which is to be broadcasted to all clients.
6. Query packet should contain

    ARPOperation| SourceMAC | SourceIP | DestinationMAC | DestinationIP

7. When an ARP Reply is received with the Destination MAC address, send the packet to the corresponding destination. Also check the validity of IP and MAC addresses.
8. Stop

Client:

1. Start
2. Enter the client's own IP and MAC addresses.
3. When an ARP Request packet is received, check whether the Destination IP is its own IP by comparing the addresses.
4. If not, send no reply.
5. If yes, respond with an ARP Reply packet.

    ARPOperation|SourceMAC | SourceIP | DestinationMAC | DestinationIP

6. Then receive the packet from the server and display it.
7. Stop

Code:

```
//ARP structure and operations - ADT

typedef char string[50];

#define REQ 1
#define ACK 2
#define DATA 3

typedef struct ARP_PACKET{
        int mode;
   string src_ip;
   string dest_ip;
   string src_mac;
   string dest_mac;
   string data;
}arp;

arp createARPPacket(int mode){
   arp packet;
        bzero(&packet, sizeof(packet));

        packet.mode = mode;
   printf("\nEnter the details of packet.\n");
   printf("Source IP\t: ");
   scanf(" %s", packet.src_ip);
   printf("Source MAC\t: ");
   scanf(" %s", packet.src_mac);
   printf("Destination IP\t: ");
   scanf(" %s", packet.dest_ip);
   printf("16 bit data\t: ");
   scanf(" %s", packet.data);

   return packet;
}

void printPacket(arp packet){
        if (packet.mode == REQ)
```

```
        printf("%d|%s|%s|%s|%s\n", packet.mode, packet.src_mac, packet.src_ip,
"00:00:00:00:00:00", packet.dest_ip);
        else if (packet.mode == ACK)
        printf("%d|%s|%s|%s|%s\n", packet.mode, packet.src_mac, packet.src_ip, packet.dest_ip,
packet.dest_mac);
        else
        printf("%d|%s|%s|%s|%s|%s\n", packet.mode, packet.src_mac, packet.src_ip,
packet.dest_ip, packet.dest_mac, packet.data);

}


//Simulation of Address Resolution Protocol (ARP) using Socket Programming: Server

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

#include "arp.h"

int main(int argc, char **argv){

        if (argc < 2){
        fprintf(stderr, "Enter port number as second argument!\n");
        exit(EXIT_FAILURE);
        }

        int PORT = atoi(argv[1]);

   struct sockaddr_in server, client;
   char buffer[1024];
   int client_sockets[10] = {0}, max, fd, sockfd, newfd, activity;
   int k, i, len, count;
   fd_set newfds;

   arp packet, recv_packet;

   packet = createARPPacket(REQ);
   printf("\nDeveloping ARP Request packet\n");
```

```
printPacket(packet);
printf("\nThe ARP Request packet has been broacasted.\n");
printf("Waiting for ARP Reply...\n");

        sockfd = socket(AF_INET, SOCK_STREAM, 0);

if(sockfd < 0){
         perror("Unable to open socket.\n");
        exit(EXIT_FAILURE);
}

bzero(&server, sizeof(server));

server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(PORT);

if(bind(sockfd, (struct sockaddr*)&server, sizeof(server)) < 0){
         perror("Bind error occurred.\n");
        exit(EXIT_FAILURE);
}

listen(sockfd, 10);

len = sizeof(client);

while(1){
        FD_ZERO(&newfds);                       //Clears socket set.
        FD_SET(sockfd, &newfds);    //Add sockfd to socket set.

        max = sockfd;

        for(i = 0; i < 10; i++){
                fd = client_sockets[i];

                if(fd > 0){
                        FD_SET(fd, &newfds);
                }

                if(fd > max){               //Store the max valued FD.
                        max = fd;
                }
        }
```

```
//Wait indefinitely till any client pings.
activity = select(max+1, &newfds, NULL, NULL, NULL);

if(activity < 0){
        perror("Select error occurred.\n");
exit(EXIT_FAILURE);
}

//if sockfd change => new connection request.
if(FD_ISSET(sockfd, &newfds)){
        newfd = accept(sockfd, (struct sockaddr*)&client, &len);

        if(newfd < 0){
                perror("Unable to accept the new connection.\n");
        exit(EXIT_FAILURE);
        }

        send(newfd,(void*)&packet, sizeof(packet), 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;
                }
        }
}

//Broadcast on all established connections
for(i = 0; i < 10; i++){
        fd = client_sockets[i];
        bzero((void*)&recv_packet, sizeof(recv_packet));

        //Check for change in FD
        if(FD_ISSET(fd, &newfds)){
                recv(fd, (void*)&recv_packet, sizeof(recv_packet), 0);

                //Check ARP response
                if(recv_packet.mode == ACK){
                        printf("\nARP Reply received: \n");
        printPacket(recv_packet);
```

```
                strcpy(packet.dest_mac, recv_packet.src_mac);
                packet.mode = DATA;

                                printf("\nSending the packet to: %s\n", packet.dest_mac);

                                send(newfd, (void*)&packet, sizeof(packet), 0);
                printf("\nPacket sent: \n");
                printPacket(packet);
                exit(EXIT_SUCCESS);
                        }
                 }
         }

   }
        close(sockfd);
   return 0;
}


//Simulation of Address Resolution Protocol (ARP) using Socket Programming: Client

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

#include "arp.h"

int main(int argc, char **argv){
        if (argc < 2){
        fprintf(stderr, "Enter port number as second argument!\n");
        exit(EXIT_FAILURE);
        }

        int PORT = atoi(argv[1]);

   struct sockaddr_in server, client;
   char buffer[1024];
   int sockfd, newfd;
```

```
int len, i, count, k;
arp packet, recv_packet;

printf("\nEnter the IP Address\t: ");
scanf("%s", packet.src_ip);
printf("Enter the MAC Address\t: ");
scanf("%s", packet.src_mac);

sockfd = socket(AF_INET, SOCK_STREAM, 0);

if(sockfd < 0){
        perror("Unable to open socket.\n");
}

bzero(&server, sizeof(server));

server.sin_family = AF_INET;
server.sin_addr.s_addr = inet_addr("127.0.0.1");
server.sin_port = htons(PORT);

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

bzero(&recv_packet, sizeof(recv_packet));
recv(sockfd, (void*)&recv_packet, sizeof(recv_packet), 0);
printf("\nARP Request Received: \n");
        printPacket(recv_packet);

        if(strcmp(packet.src_ip, recv_packet.dest_ip) == 0){
         printf("\nIP Address matches.\n");
        packet.mode = ACK;
        strcpy(packet.dest_ip, recv_packet.src_ip);
        strcpy(packet.dest_mac, recv_packet.src_mac);

         send(sockfd, (void*)&packet, sizeof(packet), 0);
         printf("\nARP Reply Sent: \n");
        printPacket(packet);

         bzero(&recv_packet, sizeof(recv_packet));
         recv(sockfd, (void*)&recv_packet, sizeof(recv_packet), 0);
         printf("\nReceived Packet is: \n");
        printPacket(recv_packet);
}
```

```
    else{
            printf("\nIP Address does not match.\n");
    }

    close(sockfd);

    return 0;
}
```

Output:

Server side:

```
root@spl21:~/kri# ./server 8080

Enter the details of packet.
Source IP       : 123.128.34.56
Source MAC      : AF-45-E5-00-97-12
Destination IP  : 155.157.65.128
16 bit data     : 1011110000101010

Developing ARP Request packet
1|AF-45-E5-00-97-12|123.128.34.56|00:00:00:00:00:00|155.157.65.128

The ARP Request packet has been broacasted.
Waiting for ARP Reply...

ARP Reply received:
2|45-DA-62-21-1A-B2|155.157.65.128|123.128.34.56|AF-45-E5-00-97-12

Sending the packet to: 45-DA-62-21-1A-B2

Packet sent:
3|AF-45-E5-00-97-12|123.128.34.56|155.157.65.128|45-DA-62-21-1A-B2|1011110000101010
root@spl21:~/kri#
```

Client-1 side:

```
root@spl21:~/kri# ./client 8080

Enter the IP Address    : 165.43.158.158
Enter the MAC Address   : 09-DF-90-26-6C-09

ARP Request Received:
1|AF-45-E5-00-97-12|123.128.34.56|00:00:00:00:00:00|155.157.65.128

IP Address does not match.
root@spl21:~/kri#
```

Client-2 side:

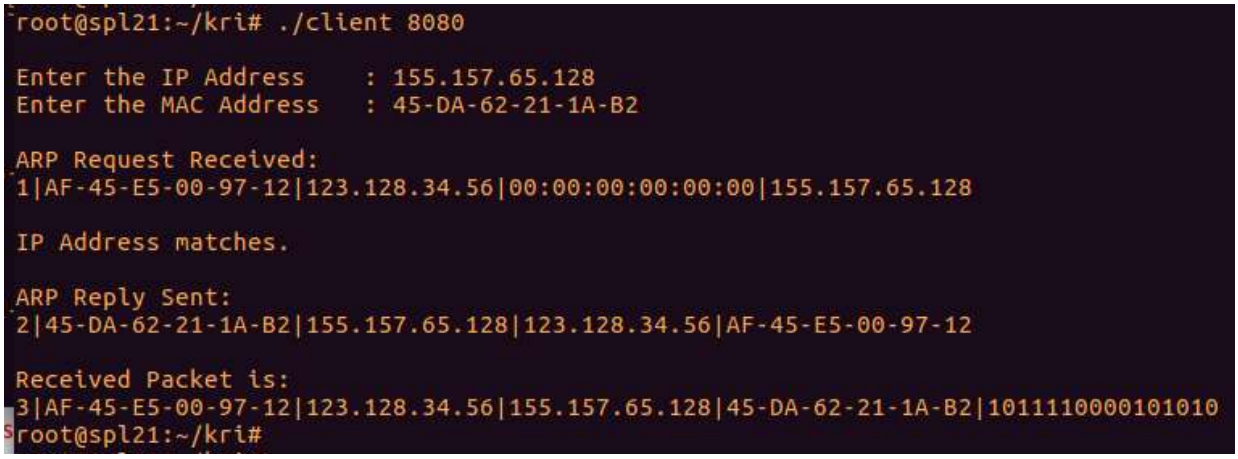
```
root@spl21:~/kri# ./client 8080

Enter the IP Address    : 15.143.158.18
Enter the MAC Address   : 19-0F-01-63-C7-D4

ARP Request Received:
1|AF-45-E5-00-97-12|123.128.34.56|00:00:00:00:00:00|155.157.65.128

IP Address does not match.
root@spl21:~/kri#
```

Client-3 side:



Learning outcomes:

- The Address Resolution Protocol was understood and implemented.
- A client-server communication was established between a server and multiple clients where an ARP Reply Packet is sent as acknowledgement only if the client's MAC address matches the required destination address.
