

Computer Networks Lab

Week-9

Oct 11, 2021

Venkata Naga Sai Ram Nomula

RA1911033010021

L2 - SWE

ARP IMPLEMENTATION USING UDP

GIVEN REQUIREMENTS:

There is a single host. The IP address of any Client in the network is given as input and the corresponding hardware address is given as the output.

TECHNICAL OBJECTIVE:

Address Resolution Protocol (ARP) is implemented through this program. The IP address of any Client is given as the input. The ARP cache is looked up for the corresponding hardware address. This is returned as the output. Before compiling that Client is pinged.

METHODOLOGY:

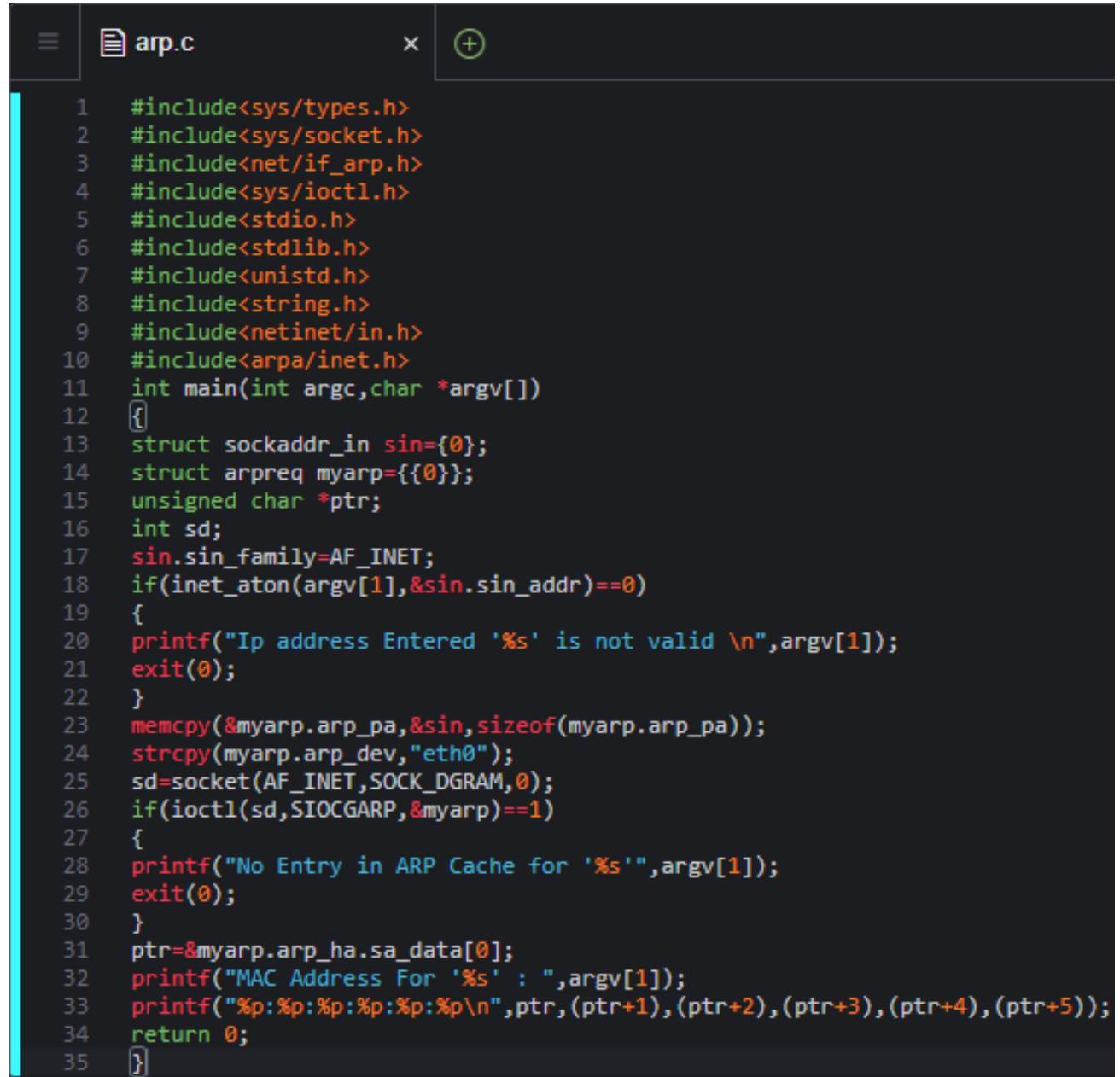
- Include the necessary header files.
- Create a socket using the socket function with family AF_INET, type as SOCK_DGRAM.
- Declare structures arpreq (as NULL structure, if required) and sockaddr_in.
- Initialize server address to 0 using the bzero function.
- Assign the sin_family to AF_INET and sin_addr using inet_aton().
- Using the object of arpreq structure, assign the name of the Network Device to the data member arp_dev like, arp_dev="eth0".
- Ping the required Client.
- Using the ioctl() we get the ARP cache entry for the given IP address.
- The output of the ioctl() function is stored in the sa_data[0] datamember of the arp_ha

- structure which is in turn a data member of structure arpreq.
- Print the hardware address of the given IP address on the output console.

Code:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<net/if_arp.h>
#include<sys/ioctl.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<netinet/in.h>
#include<arpa/inet.h>
int main(int argc,char *argv[])
{
    struct sockaddr_in sin={0};
    struct arpreq myarp={{0}};
    unsigned char *ptr;
    int sd;
    sin.sin_family=AF_INET;
    if(inet_aton(argv[1],&sin.sin_addr)==0)
    {
        printf("Ip address Entered '%s' is not valid \n",argv[1]);
        exit(0);
    }
    memcpy(&myarp.arp_pa,&sin,sizeof(myarp.arp_pa));
    strcpy(myarp.arp_dev,"eth0");
    sd=socket(AF_INET,SOCK_DGRAM,0);
    if(ioctl(sd,SIOCGARP,&myarp)==1)
    {
        printf("No Entry in ARP Cache for '%s'",argv[1]);
        exit(0);
    }
    ptr=&myarp.arp_ha.sa_data[0];
```

```
printf("MAC Address For '%s' : ",argv[1]);
printf("%p:%p:%p:%p:%p:%p\n",ptr,(ptr+1),(ptr+2),(ptr+3),(ptr+4),(ptr+5));
return 0;
}
```

A screenshot of a code editor window titled 'arp.c'. The editor has a dark background with light blue and green syntax highlighting. The code is a C program that checks if an IP address is in the ARP cache. It includes headers for system types, sockets, ARP, ioctl, stdio, stdlib, unistd, string, and network-related functions. The main function takes an IP address as an argument. It first checks if the IP is valid using inet_aton. If not, it prints an error and exits. If valid, it creates an ARP request structure, sets the device to 'eth0', and sends it via ioctl. If the response indicates no entry in the ARP cache, it prints a message and exits. Otherwise, it prints the MAC address for the given IP. The code is numbered from 1 to 35.

```
1  #include<sys/types.h>
2  #include<sys/socket.h>
3  #include<net/if_arp.h>
4  #include<sys/ioctl.h>
5  #include<stdio.h>
6  #include<stdlib.h>
7  #include<unistd.h>
8  #include<string.h>
9  #include<netinet/in.h>
10 #include<arpa/inet.h>
11 int main(int argc,char *argv[])
12 {
13     struct sockaddr_in sin={0};
14     struct arpreq myarp={{0}};
15     unsigned char *ptr;
16     int sd;
17     sin.sin_family=AF_INET;
18     if(inet_aton(argv[1],&sin.sin_addr)==0)
19     {
20         printf("Ip address Entered '%s' is not valid \n",argv[1]);
21         exit(0);
22     }
23     memcpy(&myarp.arp_pa,&sin,sizeof(myarp.arp_pa));
24     strcpy(myarp.arp_dev,"eth0");
25     sd=socket(AF_INET,SOCK_DGRAM,0);
26     if(ioctl(sd,SIOCGARP,&myarp)==1)
27     {
28         printf("No Entry in ARP Cache for '%s'",argv[1]);
29         exit(0);
30     }
31     ptr=&myarp.arp_ha.sa_data[0];
32     printf("MAC Address For '%s' : ",argv[1]);
33     printf("%p:%p:%p:%p:%p:%p\n",ptr,(ptr+1),(ptr+2),(ptr+3),(ptr+4),(ptr+5));
34     return 0;
35 }
```

Result:

```
bash - "ip-172-31-9-200" x
RA1911033010017:~/environment $ cd RA1911033010021
RA1911033010017:~/environment/RA1911033010021 $ cd 'ARP Implementation'
RA1911033010017:~/environment/RA1911033010021/ARP Implementation $ ping 172.31.9.200
PING 172.31.9.200 (172.31.9.200) 56(84) bytes of data.
64 bytes from 172.31.9.200: icmp_seq=1 ttl=64 time=0.029 ms
64 bytes from 172.31.9.200: icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from 172.31.9.200: icmp_seq=3 ttl=64 time=0.025 ms
64 bytes from 172.31.9.200: icmp_seq=4 ttl=64 time=0.027 ms
64 bytes from 172.31.9.200: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 172.31.9.200: icmp_seq=6 ttl=64 time=0.027 ms
^C
--- 172.31.9.200 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5127ms
rtt min/avg/max/mdev = 0.025/0.028/0.035/0.007 ms
RA1911033010017:~/environment/RA1911033010021/ARP Implementation $ cc arp.c
RA1911033010017:~/environment/RA1911033010021/ARP Implementation $ ./a.out 172.31.9.200
MAC Address For '172.31.9.200' : 0x7ffc3fdf3612:0x7ffc3fdf3613:0x7ffc3fdf3614:0x7ffc3fdf3615:0x7ffc3fdf3616:0x7ffc3fdf3617
```

INFERENCE:

Thus the ARP implementation is developed to get the MAC address of the remote machine's IP address from ARP cache and prints it.