Experiment-5
Simulation of ARP/RARP

Aim: Simulation ARP/RARP

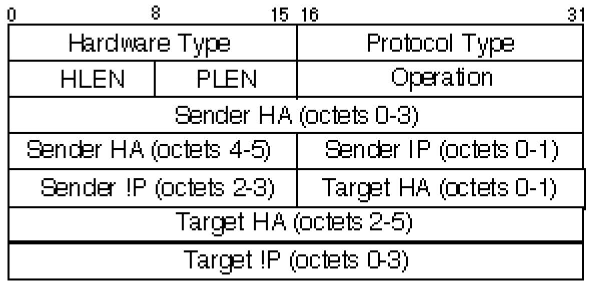
Description: ARP (Address Resolution Protocol) is a protocol used by Internet Protocol (IP). It is used to map IP address to the corresponding hardware address(MAC) used by the data link protocol. It operates below the network layer as a part of the interface between the OSI network and link layer. It is used when IPv4 is used over Ethernet.

To resolve an IP address ARP uses four types of messages that may be sent by the ARP protocol. These are identified by four values in the “operation” field of an arp message. The messages are:

1. ARP request
2. ARP reply
3. RARP request
4. RARP reply

The format of ARP message is shown in Figure 5.1

:



0	8 15	16 31
Hardware Type		Protocol Type
HLEN	PLEN	Operation
Sender HA (octets 0-3)		
Sender HA (octets 4-5)		Sender IP (octets 0-1)
Sender IP (octets 2-3)		Target HA (octets 0-1)
Target HA (octets 2-5)		
Target IP (octets 0-3)		

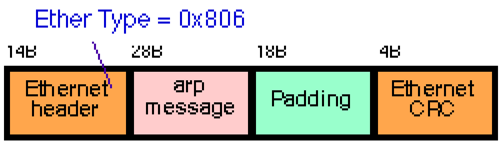
Figure 5.1: ARP header format

To reduce the number of address resolution requests, a client normally caches resolved addresses for a (short) period of time. The arp cache is of a finite size, and would become full of incomplete and obsolete entries for computers that are not in use if it was allowed to grow without check. The arp cache is therefore periodically flushed of all entries. This deletes unused entries and frees space in the cache. It also removes any unsuccessful attempts to contact computers which are not currently running.

If a host changes the MAC address it is using, this can be detected by other hosts when the cache entry is deleted and a fresh arp message is sent to establish the new association. The use of gratuitous arp (e.g. triggered when the new NIC interface is enabled with an IP address) provides a more rapid update of this information.

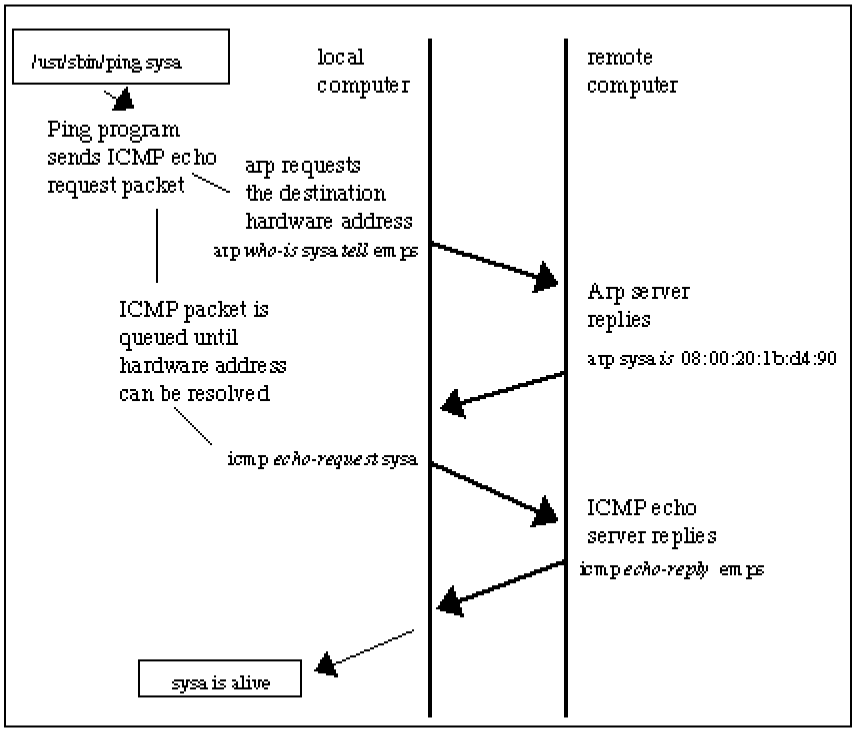
Example of use of the Address Resolution Protocol (arp)

The figure below shows the use of arp when a computer tries to contact a remote computer on the same LAN (known as "sysa") using the "ping" program. It is assumed that no previous IP datagrams have been received form this computer, and therefore arp must first be used to identify the MAC address of the remote computer.



The arp request message ("who is X.X.X.X tell Y.Y.Y.Y", where X.X.X.X and Y.Y.Y.Y are IP addresses) is sent using the Ethernet broadcast address, and an Ethernet protocol type of value 0x806. Since it is broadcast, it is received by all systems in the same collision domain (LAN). This is ensures that is the target of the query is connected to the network, it will receive a copy of the query. Only this system responds. The other systems discard the packet silently.

The target system forms an arp response ("X.X.X.X is hh:hh:hh:hh:hh:hh", where hh:hh:hh:hh:hh:hh is the Ethernet source address of the computer with the IP address of X.X.X.X). This packet is unicast to the address of the computer sending the query (in this case Y.Y.Y.Y). Since the original request also included the hardware address (Ethernet source address) of the requesting computer, this is already known, and doesn't require another arp message to find this out.


/usr/sbin/ping sysa
local computer
remote computer
Ping program sends ICMP echo request packet
arp requests the destination hardware address
arp who-is sysa tell emps
Arp server replies
arp sysa is 08:00:20:1b:d4:90
ICMP packet is queued until hardware address can be resolved
icmp echo-request sysa
ICMP echo server replies
icmp echo-reply emps
sysa is alive

Proxy ARP

Proxy ARP is the name given when a node responds to an arp request on behalf of another node. This is commonly used to redirect traffic sent to one IP address to another system.

Proxy ARP can also be used to subvert traffic away from the intended recipient. By responding instead of the intended recipient, a node can pretend to be a different node in a network, and therefore force traffic directed to the node to be redirected to itself. The node can then view the traffic (e.g. before forwarding this to the originally intended node) or could modify the traffic. Improper use of Proxy ARP is therefore a significant security vulnerability and some networks therefore implement systems to detect this. Gratuitous ARP can also help defend the correct IP to MAC bindings.

server.c ARP is

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "sys/types.h"
#include "sys/socket.h"
#include "arpa/inet.h"
#include "netinet/in.h"
#define SA struct sockaddr

struct IPmac {
                        char ip[100];
                        char mac[100];
           };


int main() {
           int sockfd,len,i;
           struct sockaddr_in servaddr;
           char buff[30],temp[30],ip[30],mac[30];
           int flag=0;

           struct IPmac in[3]={
                        {"10.1.1.8","44:dd:22:11:33"},
                        {"127.0.0.1","33:aa:fe:4e:2d"},
                        {"10.1.8.5","23:a3:5d:33:9d"}
            };

           //printing table
           printf("ip\t\tmac\n");
           for(i=0;i<3;i++)
           {
                        printf("%s\t%s\n",in[i].ip,in[i].mac);
           }

           //create socket
           sockfd = socket(AF_INET,SOCK_DGRAM,0);

           //fill structure
           servaddr.sin_family = AF_INET;
           servaddr.sin_port = htons(9999);
           servaddr.sin_addr.s_addr = INADDR_ANY;

           //bind
           bind(sockfd,(SA*)&servaddr,sizeof(servaddr));

           //get ip from client
           len=sizeof(servaddr);
```

```
            recvfrom(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,&len);

            for(i=0;i<strlen(ip)-1;i++) {
                        temp[i]=ip[i];
            }
            temp[i]='\0';
            printf("received IP :%s\n",temp);

            //searching in table for equivalent mac
            for(i=0;i<3;i++) {
                        if(strcmp(temp,in[i].ip)==0) {
                                    strcpy(mac,in[i].mac);
                                    break;
                        }
            }

            printf("mac address is %s\n",mac);
            sendto(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,len);

            //rarp simulation
            //recv mac address
            bzero(mac,sizeof(mac));
            recvfrom(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,&len);
            printf("received mac address :%s",mac);
            //store in temp
            bzero(temp,sizeof(temp));
            for(i=0;i<strlen(mac)-1;i++) {
                        temp[i]=mac[i];
            }
            temp[i]='\0';
            bzero(ip,sizeof(ip));

            //check in table
            for(i=0;i<3;i++) {
                        if(strcmp(temp,in[i].mac)==0) {
                                    strcpy(ip,in[i].ip);
                                    break;
                        }
            }
            printf("ip address :%s\n",ip);
            sendto(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,len);
            return 0;
}
```

client.c

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "sys/types.h"
#include "sys/socket.h"
#include "arpa/inet.h"
#include "netinet/in.h"
#define SA struct sockaddr

int main() {
```

```
            int sockfd,len;
            char ip[30],mac[30];
            struct sockaddr_in servaddr;

            //creating socket
            sockfd = socket(AF_INET,SOCK_DGRAM,0);

            //fill structure
            servaddr.sin_family = AF_INET;
            servaddr.sin_port = htons(9999);
            servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");

            //send ip address
            printf("ARP SIMULATION\n");
            printf("enter ip address :");
            fgets(ip,sizeof(ip),stdin);
            sendto(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,sizeof(servaddr));
            len=sizeof(servaddr);
            recvfrom(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,&len);
            printf("MAC address is: %s\n",mac);

            printf("RARP simulation\n");
            printf("enter mac address :");
            bzero(mac,sizeof(mac));
            fgets(mac,sizeof(mac),stdin);
            sendto(sockfd,mac,sizeof(mac),0,(SA*)&servaddr,len);
            recvfrom(sockfd,ip,sizeof(ip),0,(SA*)&servaddr,&len);
            printf("IP address is: %s\n",ip);

            return 0;
}
```

RESULTS AND DISCUSSIONS:

CONCLUSSIONS:

References: