

INSTITUTO POLITÉCNICO NACIONAL

ESCOM

Redes

PRÁCTICA 3

"Protocolo ARK"

INTEGRANTES :

González Mora Erika Giselle

Olivares Ménez Gloria Oliva

GRUPO: 2CV16

Índice

| | |
|--|---|
| 1. Introducción | 3 |
| 1.1. ¿Qué es el ARP (Address Resolution Protocol)? | 3 |
| 1.2. Definición del protocolo ARP | 3 |
| 1.3. ¿Cómo funciona el ARP? | 3 |
| 2. Desarrollo | 4 |
| 2.1. Funcionamiento del programa | 4 |
| 3. Conclusiones | 6 |
| 3.1. González Mora Erika Giselle | 6 |
| 3.2. Olivares Ménez Gloria Oliva | 6 |
| 4. Bibliografía | 6 |
| 5. Anexos | 7 |
| a) Programa | 7 |

1. Introducción

1.1. ¿Qué es el ARP (Address Resolution Protocol)?

Para poder enviar paquetes de datos en redes TCP/IP, un servidor necesita, sobre todo, tres datos de dirección sobre el host al que se dirige: la máscara de subred, la dirección IP y la dirección MAC (también conocida como dirección de hardware o dirección física). Los dispositivos reciben la máscara de red y la dirección IP de manera automática y flexible cuando se establece la conexión con una red. Con este objetivo, los dispositivos de comunicación mediadores como routers o concentradores (hubs) recurren al protocolo DHCP. En las redes locales se pueden introducir ambos datos manualmente. El fabricante del dispositivo correspondiente otorga la dirección de hardware, que queda vinculada a una dirección IP con ayuda del llamado Address Resolution Protocol (ARP).

1.2. Definición del protocolo ARP

El Address Resolution Protocol (protocolo de resolución de direcciones) fue especificado en 1982 en el estándar RFC 826 para llevar a cabo la resolución de direcciones IPv4 en direcciones MAC. ARP es imprescindible para la transmisión de datos en redes Ethernet por dos razones: por un lado, las tramas de datos (también tramas Ethernet) de los paquetes IP solo pueden enviarse con ayuda de una dirección de hardware a los hosts de destino, pero el protocolo de Internet no puede obtener estas direcciones físicas por sí mismo. Por el otro, y debido a su limitada longitud, el protocolo IPv4 carece de la posibilidad de almacenar las direcciones de los dispositivos. Con un mecanismo de caché propio, el protocolo ARP también es, aquí, la solución más adecuada. IPv6, por su parte, adopta las funciones del Neighbor Discovery Protocol (NDP).

1.3. ¿Cómo funciona el ARP?

A la hora de asignar direcciones por medio del Address Resolution Protocol hay que distinguir si la dirección IP del host de destino se encuentra en la misma red local o en otra subred. Así, en caso de asignar una dirección MAC a una determinada dirección IP, antes de nada se lleva a cabo una revisión de la máscara de subred. Si la IP se encuentra en la red local, el primer paso es controlar si ya existe una entrada para ella en la cache del ARP.

Si una dirección IP ya tiene asignada una dirección física, es esta la que se utiliza para el direccionamiento. En caso contrario, el remitente envía una solicitud ARP (ARP Request) con la dirección IP de destino a todos los hosts de la red. Para tal fin, el emisor utiliza la dirección de broadcast de ARP FF:FF:FF:FF:FF:FF como dirección del destinatario. Cada una de las estaciones compara la dirección IP indicada en la petición con las suyas propias y rechaza la solicitud si no hay coincidencia. Si una estación percibe que se

trata de la dirección propia, reacciona con una respuesta ARP (ARP Reply) en la que, entre otros datos, también transmite la dirección MAC. Ambas partes pueden incorporar la dirección MAC y la IP de la otra parte en la memoria caché, sentando las bases para el intercambio de datos.

Si el host de destino no se encuentra en la misma subred, el remitente se dirige a la puerta de enlace estándar (en la mayoría de los casos un router). Puede acceder a ella mediante la combinación de dirección MAC e IP, por lo que aquí también se necesita el Address Resolution Protocol. Una vez resueltas las direcciones, la puerta de enlace recibe el paquete de datos y a continuación lo envía al host de destino. Para ello esta pasarela de enlace analiza la cabecera IP para obtener los datos necesarios. A continuación, sirviéndose de las posibilidades del protocolo ARP, resuelve la dirección física directamente cuando esta se encuentra en una subred adyacente, o resuelve la dirección de hardware de otra puerta de enlace cuando el ordenador de destino se encuentra en una subred remota y no se puede determinar la trayectoria del paquete con ayuda de la tabla de enrutamiento.

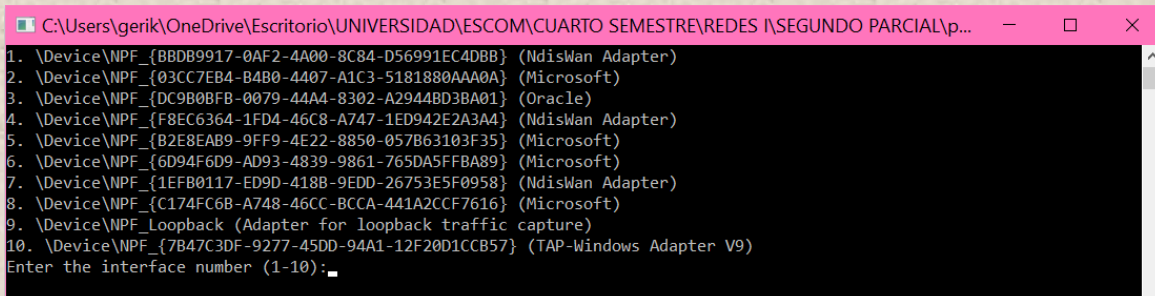
En el último caso, el proceso se repite tantas veces como sea necesario hasta que el paquete de datos llegue a su destino o hasta que el campo TTL (Time to Live) haya adoptado el valor 0 en la cabecera IP.

2. Desarrollo

Para implementar el protocolo ARP se desarrolló una aplicación en C, la cual se encarga de capturar tramas al vuelo, producidas por la tarjeta de red instalada en nuestra computadora. Al recibir un paquete, el programa busca el campo longitud, el cual se ubica en los bytes 13 y 14 de la trama Ethernet, si este campo es equivalente a 0x0806, o bien 2054 en el sistema decimal, se trata de una trama ARP y posteriormente el programa desplegará la información contenida en el mensaje ARP.

2.1. Funcionamiento del programa

Al ejecutar el programa se nos preguntará con cuál de las tarjetas instaladas en nuestra computadora queremos realizar la captura de tramas al vuelo.



```
C:\Users\gerik\OneDrive\Escritorio\UNIVERSIDAD\ESCOM\CUARTO SEMESTRE\REDES I\SEGUNDO PARCIAL\p... - □ ×
1. \Device\NPF_{BBDB9917-0AF2-4A00-8C84-D56991EC4DBB} (NdisWan Adapter)
2. \Device\NPF_{03CC7EB4-B4B0-4407-A1C3-5181880AAA0A} (Microsoft)
3. \Device\NPF_{DC9B0BFB-0079-44A4-8302-A2944BD3BA01} (Oracle)
4. \Device\NPF_{F8EC6364-1FD4-46C8-A747-1ED942E2A3A4} (NdisWan Adapter)
5. \Device\NPF_{B2E8EAB9-9FF9-4E22-8850-057B63103F35} (Microsoft)
6. \Device\NPF_{6D94F6D9-AD93-4839-9861-765DA5FFBA89} (Microsoft)
7. \Device\NPF_{1EFB0117-ED9D-418B-9EDD-26753E5F0958} (NdisWan Adapter)
8. \Device\NPF_{C174FC6B-A748-46CC-BCCA-441A2CCF7616} (Microsoft)
9. \Device\NPF_{Loopback} (Adapter for loopback traffic capture)
10. \Device\NPF_{7B47C3DF-9277-45DD-94A1-12F20D1CCB57} (TAP-Windows Adapter V9)
Enter the interface number (1-10):
```

Una vez que seleccionamos la tarjeta de red, comenzará a realizar la captura de tramas al vuelo, pero sólo mostrará las tramas ethernet de longitud 2054, las cuales se tratan de tramas ARP. En la información de la trama podremos ver la trama en crudo, en seguida se nos muestra la dirección MAC destino (bits 0 a 5), la dirección MAC origen (bits 6 a 11), y el campo tipo (bits 12 y 13), los cuáles se obtienen de la trama recibida. En el protocolo ARP tenemos dos tipos de mensajes, los cuales se definen por los bits 20 y 21 de la trama; si en los bits 20 y 21 tenemos el número 0x0001 se trata de un mensaje ARP de pregunta, mientras que si tenemos el número 0x0002 se tratará de un mensaje ARP de respuesta. A continuación se muestra el mensaje de pregunta y su respectiva respuesta:

```
Trama ARP
3c f0 11 52 07 0b 40 2b 50 f0 9b 86 08 06 00 01
08 00 06 04 00 01 40 2b 50 f0 9b 86 c0 a8 00 01
00 00 00 00 00 00 c0 a8 00 0a 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00

MAC destino: 3c:f0:11:52:07:0b
MAC origen: 40:2b:50:f0:9b:86
Tipo: ARP (2054)
Hardware type: Ethernet (1)
Protocol type: IPv4 (2048)
Hardware size: 6
Protocol size: 4
Opcode: ARP request (1)
Sender MAC address: 40:2b:50:f0:9b:86
Sender IP address: 192.168.0.1
Target MAC address: 00:00:00:00:00:00
Target IP address: 192.168.0.10
```

```
Trama ARP
40 2b 50 f0 9b 86 3c f0 11 52 07 0b 08 06 00 01
08 00 06 04 00 02 3c f0 11 52 07 0b c0 a8 00 0a
40 2b 50 f0 9b 86 c0 a8 00 01

MAC destino: 40:2b:50:f0:9b:86
MAC origen: 3c:f0:11:52:07:0b
Tipo: ARP (2054)
Hardware type: Ethernet (1)
Protocol type: IPv4 (2048)
Hardware size: 6
Protocol size: 4
Opcode: ARP reply (2)
Sender MAC address: 3c:f0:11:52:07:0b
Sender IP address: 192.168.0.10
Target MAC address: 40:2b:50:f0:9b:86
Target IP address: 192.168.0.1
```


3. Conclusiones

3.1. González Mora Erika Giselle

En esta práctica, pudimos implementar el análisis para la resolución de direcciones, mediante el protocolo ARP, y su comportamiento ante la captura de varias tramas. Sin embargo, lo que realmente comprendimos, fue que el protocolo ARP tiene un papel clave entre los protocolos de capa de Internet relacionados con el protocolo TCP/IP, ya que permite que se conozca la dirección física de una tarjeta de interfaz de red correspondiente a una dirección IP. Lo diferente a las 2 prácticas que hemos realizado anteriormente es que el protocolo ARP utiliza un formato simple de mensaje para enviar solicitudes o responder a estas. Aunque inicialmente estaba previsto para direcciones IPv4 y direcciones MAC, también son válidos otros protocolos de red, de ahí que existan campos para el tipo y tamaño de las direcciones de hardware y de protocolo. Como consecuencia, también puede haber diferencias en el tamaño total de los paquetes ARP.

3.2. Olivares Ménez Gloria Oliva

En redes de computadoras podemos decir que el protocolo de resolución de direcciones (ARP) es un protocolo muy útil para encontrar la dirección de hardware (dirección MAC) que corresponde a una dirección IP determinada. Como sabemos, en las tramas Ethernet se envía información que proviene de un dispositivo y que se espera que la reciba otro dispositivo, pero será el tipo de trama lo que determinará la información que se lleva. Las tramas de tipo 0x0806 se clasifican como tramas ARP y su funcionamiento es muy simple de entender. Lo que hace una trama ARP es enviar un paquete de pregunta (identificado como tipo 0x0001 en los bits 20 y 21) a la dirección de broadcast de nuestra red (FF:FF:FF:FF:FF:FF) y en este paquete se contiene la dirección IP por la que se está preguntando, y esta misma espera a que la máquina que contiene esa IP, o incluso otra máquina, respondan mediante un paquete ARP de respuesta (identificado como tipo 0x0002 en los bits 20 y 21) con la dirección MAC que le corresponde a esa dirección IP. Cada máquina guarda en una memoria caché las direcciones que fueron traducidas para poder reducir la búsqueda en una futura trama ARP de pregunta. Podemos encontrar el protocolo ARP documentado en el RFC 826.

4. Bibliografía

1&1 IONOS Inc. (2020, 2 noviembre). *ARP: resolución de direcciones en la red*. IONOS Digitalguide. <https://www.ionos.mx/digitalguide/servidores/know-how/arp-resolucion-de-direcciones-en-la-red/>

5. Anexos

a) Programa

```
#ifndef _MSC_VER
#define _CRT_SECURE_NO_WARNINGS
#endif
#include <pcap.h>
#include <stdlib.h>
#include <stdio.h>
#define LINE_LEN 16
#define PAQUETES 500

void packet_handler(u_char *param, const struct pcap_pkthdr
*header, const u_char *pkt_data);
void imprimirTrama(const struct pcap_pkthdr *, const u_char *);
void analizarARP(const u_char *);
void imprimirHarwareType(unsigned short);
void imprimirProtocolType(unsigned short);
void imprimirOpcode(unsigned short);
void imprimirMac(const u_char *, int, int);
void imprimirIp(const u_char *, int, int);

//Variables globales
int ARP = 0, noARP = 0;

int main()
{
    pcap_if_t *alldevs;
    pcap_if_t *d;
    int inum;
    int i = 0;
    pcap_t *adhandle;
    char errbuf[PCAP_ERRBUF_SIZE];

    /* Retrieve the device list */
    if(pcap_findalldevs(&alldevs, errbuf) == -1)
    {
        fprintf(stderr, "Error in pcap_findalldevs: %s\n",
errbuf);
        exit(1);
    }

    /* Jump to the selected adapter */
    for(d=alldevs; d; d=d->next)
    {
        printf("%d. %s", ++i, d->name);
        if (d->description)
            printf(" (%s)\n", d->description);
    }
}
```

```

        else
            printf(" (No description available)\n");
    }

    if(i == 0)
    {
        printf("\nNo interfaces found! Make sure WinPcap is
installed.\n");
        return -1;
    }

    printf("Enter the interface number (1-%d):",i);
    scanf("%d", &inum);

    if(inum < 1 || inum > i)
    {
        printf("\nInterface number out of range.\n");
        /* Free the device list */
        pcap_freealldevs(alldevs);
        return -1;
    }

    /* Jump to the selected adapter */
    for(d=alldevs, i=0; i< inum-1 ;d=d->next, i++);

    /* Open the device */
    /* Open the adapter */

    if ((adhandle= pcap_open_live(d -> name,    // name of the
device
                                65536,          //
portion of the packet to capture.
                                //
65536 grants that the whole packet will be captured on all the
MACs.
                                1,              //
promiscuous mode (nonzero means promiscuous)
                                1000,           // read
timeout
                                errbuf         //
error buffer
                                )) == NULL)
    {
        fprintf(stderr, "\nUnable to open the adapter. %s is not
supported by WinPcap\n", d->name);
        /* Free the device list */
        pcap_freealldevs(alldevs);
        return -1;
    }

    printf("ANALIZADOR DEL PROTOCOLO ARP");

```



```

    /* At this point, we don't need any more the device list.
Free it */
    pcap_freealldevs(alldevs);

    /* start the capture */
    pcap_loop(adhandle, PAQUETES, packet_handler, NULL);

    printf("\n\n\nNumero de tramas ARP encontradas: %d", ARP);
    printf("\nNumero de tramas no ARP encontradas: %d\n", noARP);

    pcap_close(adhandle);

    return 0;
}

/* Callback function invoked by libpcap for every incoming packet
*/
void packet_handler(u_char *param, const struct pcap_pkthdr
*header, const u_char *pkt_data)
{
    int j = 0, k = 0;

    switch ((pkt_data[12] << 8) + pkt_data[13])
    {
        case 2054:
            printf("\n\n\nTrama ARP \n\n");
            imprimirTrama(header, pkt_data);
            analizarARP(pkt_data);
            ARP++;
            break;
        default:
            noARP++;
            break;
    }
}

void imprimirTrama(const struct pcap_pkthdr *header, const u_char
*pkt_data)
{
    int i;

    for (i = 1; (i < header -> caplen + 1); i++)
    {
        printf("%.2x ", pkt_data[i - 1]);

        if ( (i % LINE_LEN) == 0)
            printf("\n");
    }
}

void analizarARP(const u_char *pkt_data)
{

```

```

        //MAC destino
        printf("\n\nMAC destino: ");
        imprimirMac(pkt_data, 0, 5);

        //MAC origen
        printf("\n\nMAC origen: ");
        imprimirMac(pkt_data, 6, 11);

    printf("\nTipo: ");
    imprimirProtocolType((pkt_data[12] << 8) + pkt_data[13]);

    //Tipo de hardware
    printf("\nHardware type: ");
    imprimirHarwareType((pkt_data[14] << 8) + pkt_data[15]);

    //Tipo de protocolo
    printf("\nProtocol type: ");
    imprimirProtocolType((pkt_data[16] << 8) + pkt_data[17]);

    //Tamaño de hardware
    printf("\nHardware size: ");
    printf("%d", pkt_data[18]);

    //Tamaño de protocolo
    printf("\nProtocol size: ");
    printf("%d", pkt_data[19]);

    //Opcode
    printf("\nOpcode: ");
    imprimirOpcode((pkt_data[20] << 8) + pkt_data[21]);

    //MAC address del remitente
    printf("\nSender MAC address: ");
    imprimirMac(pkt_data, 22, 27);

    //IP address del remitente
    printf("\nSender IP address: ");
    imprimirIp(pkt_data, 28, 31);

    //MAC address del objetivo
    printf("\nTarget MAC address: ");
    imprimirMac(pkt_data, 32, 37);

    printf("\nTarget IP address: ");
    imprimirIp(pkt_data, 38, 41);
}

void imprimirHarwareType(unsigned short tipo)
{
    switch(tipo)
    {
        case 1:

```



```

        printf("Ethernet");
        break;
    case 6:
        printf("IEEE 802 Networks");
        break;
    case 7:
        printf("ARCNET");
        break;
    case 15:
        printf("Frame relay");
        break;
    case 16:
    case 19:
        printf("Asynchronous Transfer Mode (ATM)");
        break;
    case 17:
        printf("HDLC");
        break;
    case 18:
        printf("Fibre Channel");
        break;
    case 20:
        printf("Serial Line");
        break;
    default:
        printf("Unknown");
        break;
}

printf(" (%d)", tipo);
}

void imprimirProtocolType(unsigned short tipo)
{
    switch(tipo)
    {
        case 2048:
            printf("IPv4");
            break;
        case 2054:
            printf("ARP");
            break;
        default:
            printf("Unknown");
            break;
    }

    printf(" (%d)", tipo);
}

void imprimirOpcode(unsigned short tipo)
{

```

```

switch(tipo)
{
    case 1:
        printf("ARP request");
        break;
    case 2:
        printf("ARP reply");
        break;
    case 3:
        printf("RARP request");
        break;
    case 4:
        printf("RARP reply");
        break;
    default:
        printf("Unknown");
        break;
}

printf(" (%d)", tipo);
}

void imprimirMac(const u_char *pkt_data, int inicio, int fin)
{
    int i = inicio;

    for(; i <= fin; i++)
    {
        printf("%02x", pkt_data[i]);

        if(i < fin)
            printf(":");
    }
}

void imprimirIp(const u_char *pkt_data, int inicio, int fin)
{
    int i = inicio;

    for(; i <= fin; i++)
    {
        printf("%d", pkt_data[i]);

        if(i < fin)
            printf(".");
    }
}

```