**Arquitectura de Redes 2**

**Práctica 2: ARP e IP**

**Introducción**

El objetivo de esta práctica es familiarizarse con los protocolos ARP e IP, realizando una implementación básica de cada uno de ellos.

La práctica está dividida en dos partes: En la primera parte se debe implementar el protocolo ARP para encontrar la dirección *Ethernet* local a la que se debe enviar un paquete destinado a una determinada dirección IP. En la segunda parte se familiarizará con el protocolo IP, con el que podrá implementar un *talk* entre dos máquinas.

Cabe destacar que la implementación y desarrollo de las dos partes de la práctica es incremental. Esto es, para la realización de la segunda parte es necesario tener implementada la primera.

**Tabla de Contenidos de la práctica**

**Parte A. ARP.**

* Fundamentos del protocolo
* Implementación
* Criterios de evaluación

**Parte B. IP.**

* Fundamentos del protocolo
* Criterios de evaluación y fechas de entrega/calificaciones

**PARTE A: FUNDAMENTOS DEL PROTOCOLO ARP**

El protocolo ARP (*Address* *Resolution Protocol*, [RFC 826](http://www.faqs.org/rfcs/rfc826.html)) es un protocolo utilizado en la resolución de direcciones de enlace (direcciones físicas) a partir de direcciones de red. ARP está diseñado para soportar diferentes protocolos de enlace y de red, pero en esta práctica solamente trabajaremos con los habituales, Ethernet para realizar el enlace e IP (Internet Protocol) como protocolo de red.

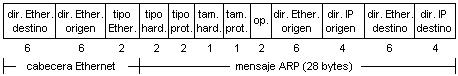
***Nota:*** El protocolo IP será estudiado en la segunda parte de la práctica y de momento no es necesario conocerlo para la implementación de esta primera parte.

Por lo tanto, el protocolo ARP es un sencillo protocolo que a partir de una dirección de red (dirección IP) obtiene su correspondiente dirección física (dirección Ethernet o también llamada: 'MAC address').

Cuando una máquina se dispone a enviar información a otra situada en la misma red de área local (LAN), debe obtener su dirección física, a partir de su dirección IP. Esto ocurre siempre así cuando un host usando TCP/IP quiere enviar información a otro host de su red local. Al host que quiere enviar la información no le basta con conocer la dirección IP del host destino, ya que la tarjeta de red no entiende de direcciones IP, sino que tan sólo reconoce direcciones definidas por la interfaz de acceso al medio que implementa, por ejemplo Ethernet.

De esta manera ARP permite obtener una dirección Ethernet (6 bytes) a partir de una dirección IP (4 bytes). Obviamente la máquina cuya dirección Ethernet se busca ha de estar conectada a la misma red de área local que la máquina que hace la solicitud.

ARP sobre Ethernet

La estructura de una trama Ethernet con un paquete ARP es la siguiente:

El significado y codificación de los campos es el siguiente:

* *dir. Ether. destino, dir. Ether. origen:* Direcciones Ethernet del receptor y del emisor del mensaje.
* *tipo Ether.:* El tipo definido para el protocolo ARP es el **0x806.**
* *tipo hard.:* Tipo de direcciones "hardware". Su valor es **1** para Ethernet.
* *tipo prot.:* Tipo de direcciones "de protocolo". Su valor es **0x800** para direcciones IP.
* *tam. hard.:* Tamaño de las direcciones "hardware". Su valor es **6** para Ethernet.
* *tam. prot.:* Tamaño de las direcciones "de protocolo". Su valor es **4** para IP.
* *op.:* Operación efectuada por el mensaje. Petición (Request) = **1**, Respuesta (Reply) = **2.**
* *dir. IP origen*, *dir. IP destino*: Direcciones IP del emisor y del receptor del mensaje.

Protocolo

Cuando un host A quiere conocer la dirección Ethernet de otro host B conectado a su misma red, envía un mensaje ARP de petición en modo 'broadcast' (*dirección Ethernet destino = FF:FF:FF:FF:FF:FF*).  
En dicho mensaje ARP repite su dirección Ethernet, ya presente en la cabecera Ethernet, en el cuerpo del mensaje (*dir. Ether. origen*) e indica su dirección IP (*dir. IP origen*). Rellena *dir. IP destino* con la dirección IP de B y deja *dir. Ether. destino* como *FF:FF:FF:FF:FF:FF*.  
B, al recibir este paquete (lo cual ocurrirá siempre que su tarjeta tenga habilitada la recepción de tramas broadcast), responde con un mensaje ARP de respuesta, para lo cual le basta con cambiar el código op. de 1 a 2, rellenar su propia dirección Ethernet e intercambiar los campos. Esta vez el mensaje irá dirigido directamente a A en lugar de ser difundido.

La máquina A tomará nota de la información obtenida, apuntando la correspondencia de direcciones IP-Ethernet de B en una caché, con lo cual se evitarán futuras peticiones para mensajes IP dirigidos a la misma máquina. Por otra parte, B, al recibir la petición de A, tomará nota en su caché de la dirección Ethernet de esta máquina, si no la tenía ya almacenada.

La caché mantiene una lista de las direcciones obtenidas por las sucesivas peticiones y respuestas recibidas. Se suele implementar un tiempo vida (TTL, *time tolive*) para cada entrada de la caché, borrándose dicha entrada cuando su tiempo de vida expira. Esto fuerza un cierto refresco en las cachés de los hosts de una red, de forma que entradas no usadas o incorrectas acaban siendo eliminadas. Por otro lado, si la caché es de un tamaño limitado, puede resultar necesario sustituir una entrada por otra; una política común es eliminar aquella de uso menos reciente, o más sencillo, aquella a la que le quede un menor tiempo de vida.

Algunos sistemas operativos emiten un ARP al arrancar la maquina, en este ARP se pregunta por su propia dirección IP (denominado *ARP gratuito*), si alguna maquina les contesta podemos decir que existen dos maquinas con la misma IP, al detectar esto los SO no inicializan los protocolos de comunicaciones.

***IMPLEMENTACIÓN***

Es **obligatorio** usar los siguientes recursos en la elaboración de esta parte de la práctica. Se proporcionan para facilitar el desarrollo (los prototipos y definiciones de datos deberían resultar muy útiles a la hora de plantear la práctica), las pruebas (el programa de prueba presupone el uso de arpt.c ), y evitar tentaciones de reutilizar código de otros años sin necesidad de entenderlo.

* arpt.c- Programa de prueba. No se debe modificar.
* arp.h y arp.c- Cabeceras y esqueleto de implementación para ARP. Sólo se deben modificar partes de arp.c. La versión proporcionada ya compila, e incluye una caché sencilla y perfectamente funcional.
* rc\_funcs.c y rc\_funcs.h- Funciones útiles para implementar ARP e IP. No se deben modificar.

Se recomienda seguir los pasos siguientes para hacer funcionar la primera parte:

* Prepara el entorno. Abre 3 consolas, usa una para arrancar un demonio 'ethd' para funcionamiento "en local" mediante ethd 127 0 v (máximo nivel de trazas, sin introducir errores, en modo virtual). En las otras dos consolas, usa export EHOST=1; export IPLOCAL=1.1.1.1 y export EHOST=2; export IPLOCAL=2.2.2.2 respectivamente. La variable de entorno EHOST=n hace que las aplicaciones lanzadas desde esa consola usen la dirección Ethernet "virtual" 00:00:00:00:00:0n, y la variable de entorno IPLOCAL hace lo propio para la dirección IP.
* Descomprime el fichero codigo\_arq2\_p2.tgz, y compila sus fuentes mediante "make". Lanza una aplicación arpt en cada una de las consolas anteriores. Verás que no acaba de funcionar bien (faltan varias funciones por implementar). Abre arp.c y completa aquellas funciones en las que haya comentarios de la forma //PRACTICA: implementa…
* Durante la implementación, es importante **hacer pruebas frecuentes**. Planifica el orden de implementación para facilitar estas pruebas, y haz paradas frecuentes para verificar que vas por buen camino.
* Sigue haciendo pruebas en local hasta que todo funcione. Llegado ese momento, cambia IPLOCAL para que refleje tu verdadera direccion IP (usa/sbin/ifconfig para averiguarla), apaga el demonio ethd mediante un Ctrl+C en su consola y relánzalo usando ethd 127 (máximo nivel de trazas, modo real) -- y repite las pruebas usando 2 ordenadores.

**CRITERIOS DE EVALUACIÓN**

* Manda petición propia ARP correcta (broadcast) 2 puntos
* Recibe y procesa ARP correcto (sin manejo del cache) 1,5 puntos
* Da respuestas a peticiones ARP correctas 3 puntos
* Maneja mensajes con su propia IP correctamente:

a. del mismo origen (ARP gratuito). 0,75 puntos

b. de otro origen (Errores). 0,75 puntos

* Maneja la tabla correctamente:

a. de las peticiones propias 0,5 puntos

b. de las peticiones ajenas y los broadcast 0,5 puntos

* Comunicación entre enviar y recibir mensaje:

con comunicación entre hilos correcta 0,5 puntos

* Retry y espera sin perdida de tiempo de CPU 0,5 puntos

NOTA: Las pruebas siempre se realizarán en un ENTORNO REAL. Esto es, nuestro programa “arpt” debe ser capaz de:

* Realizar peticiones ARP (y procesar la respuesta) a máquinas reales con Linux (sin ningún programa nuestro corriendo) o Windows.
* Dar respuesta a peticiones ARP de otras máquinas reales con Linux (sin ningún programa nuestro corriendo) o Windows.

**\*PARTE B: FUNDAMENTOS DEL PROTOCOLO IP**

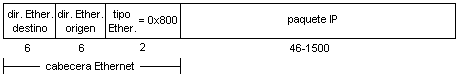
Cabe recordar que es necesario tener el protocolo ARP en perfecto funcionamiento de la parte A.

El protocolo IP (Internet Protocol, [RFC 791](http://www.faqs.org/rfcs/rfc791.html)) es el principal del conjunto conocido como TCP/IP. Se trata de un protocolo de red que permite la comunicación de ordenadores conectados a redes diferentes. Otros protocolos como TCP, UDP, ICMP, etc. usan datagramas IP para transportar su información. Tres características son especialmente significativas en este protocolo:

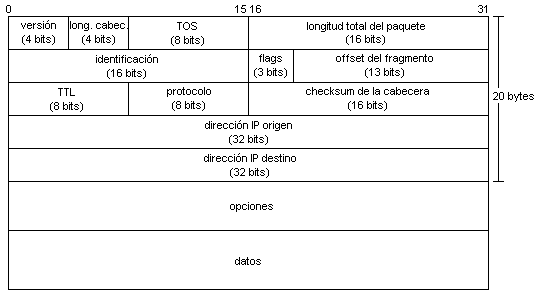
* Es un protocolo **no orientado a conexión** (uso de datagramas). No se guarda información sobre los sucesivos datagramas que se envían o reciben. Esto conlleva que si dos paquetes cambian su orden relativo en el camino entre un host y otro, llegando antes el enviado más tarde debido a la variación en la ruta seguida, serán pasados desordenados a la capa superior.
* Es un **protocolo no fiable**, no se garantiza que un paquete que se pase al protocolo para ser enviado llegue al otro extremo. Los paquetes pueden perderse o corromperse sin generar por ello comunicaciones de error ni retransmisiones. Si se detecta un error en un datagrama, el datagrama simplemente se descarta. La fiabilidad en la comunicación la debe aportar una capa superior (por ejemplo TCP, protocolo de transporte orientado a conexión).
* Puede **adaptarse a las diferentes características** de las redes por las que ha de pasar un paquete, ya que permite la fragmentación de un datagrama en varios fragmentos. Los fragmentos deberán recomponerse de nuevo en un datagrama único al llegar al host destinatario. De nuevo aquí puede producirse un desorden en el orden de llegada de los fragmentos. Esta vez la capa IP deberá ser capaz de restablecer el orden adecuado, pues la fragmentación ha de ser totalmente transparente a la capa superior. Eso sí, como es lógico no se ejerce aquí tampoco ningún control sobre las perdidas o corrupciones de los fragmentos, que provocarán el descarte de un datagrama entero.

IP sobre Ethernet

Los paquetes IP se transmiten encapsulados directamente en una (o varias) trama(s) Ethernet:



El formato de un paquete es el siguiente:



El paquete se transmite en el siguiente orden respecto a la figura anterior: De izquierda a derecha y de arriba hacia abajo. Los valores de más de 8 bits se transmiten con el byte/bit más significativo primero y el menos significativo el último.  
Nótese que este criterio es el contrario al utilizado por los microprocesadores x86 de Intel, con lo cual no se puede hacer copia directa de memoria desde variables numéricas a un array con la trama: hay que invertir el orden de los bytes. Las direcciones, como es habitual, se transmiten con el mismo orden con que se escriben.

El significado y codificación de los campos es el siguiente:

* versión: Versión del protocolo IP usado. La extendida actualmente es la codificada con el 4.
* long. cabec.: Longitud de la cabecera, en palabras de 32 bits. El mínimo es 5 (20 bytes) y el máximo queda limitado a 15 (60 bytes), dejando un espacio de 40 bytes para las opciones.
* TOS: Type Of Service. Conjunto de flags que indican el tipo de servicio de comunicación que se desea, en cuanto a las características deseables de la ruta: minimizar retardo, maximizar velocidad, maximizar fiabilidad y minimizar coste. Estos cuatro bits-flags van precedidos de tres bits no usados normalmente y seguidos de un bit que debe ser 0. La [RFC 1340](http://www.faqs.org/rfcs/rfc1340.html) da valores recomendados para las aplicaciones más habituales. Si no se usa este campo, se dejará a cero.
* longitud total del paquete: En bytes, incluye cabecera y datos. El tamaño de este campo limita pues el datagrama IP más grande a 65.535 bytes (como máximo 65.515 de datos), pero un host no está obligado a aceptar paquetes de tamaño superior a 576 bytes, por lo que para mandar paquetes más largos debe tenerse la certeza de que el otro extremo permita su recepción.

NOTA: Aunque la tarjeta de red pueda decirnos qué longitud tiene una trama, si existe una longitud mínima de trama, como en Ethernet, se hace necesario este campo para saber hasta dónde llega el mensaje real.

Si la MTU (Maximum Transmission Unit) del enlace por el que deba circular este datagrama es menor que el tamaño de este, el datagrama será fragmentado. En este caso, cada uno de los fragmentos generados llevarán como valor de este campo la longitud total del fragmento (cabecera + datos) en lugar de la longitud total del datagrama original.

* identificación: Este campo lo usará IP para identificar cada datagrama que envía, incrementando normalmente su valor de uno en uno. La utilidad principal se obtiene con la fragmentación, al permitir a IP identificar los diferentes fragmentos de un mismo datagrama (todos llevarán el mismo valor en este campo). Aunque la [RFC 791](http://www.faqs.org/rfcs/rfc791.html) sugiere que sea la capa superior la que dé este valor a la función de envío de IP, en la práctica es la propia capa IP la que suele asignar números consecutivos a este identificador.
* flags: Tienen la siguiente disposición: [0][DF][MF]. El primer bit, reservado, debe dejarse a 0. DF (Don't Fragment) lo activa el host que envía el datagrama si quiere impedir que este se fragmente. Si el paquete llega a un router que no puede encaminar un paquete tan grande sin fragmentarlo, este router devolverá un mensaje de error ICMP al host que lo envió. MF (More Fragments) se usa para identificar el último fragmento de un datagrama: Todos los fragmentos llevan este bit a 1 menos el último.
* offset del fragmento: El valor de este campo, multiplicado por 8, nos da el offset del fragmento en el datagrama al que pertenece. Esta forma de codificación obliga a la capa IP a dividir los datagramas en fragmentos de tamaño múltiplo de 8 bytes (salvo el último fragmento).
* TTL (Time To Live): El host que envía el paquete IP rellena este campo con un valor inicial. Cada router por el que pasa debe decrementar este campo una unidad. Si un router recibe un paquete con un valor TTL=1 y el paquete no va destinado a él (pues el router puede ser un host), en vez de encaminarlo debe descartarlo y devolver al host remitente un mensaje de error ICMP. Si TTL=0 lo descartará directamente (esta situación no debería darse nunca). De esta forma se limita la "vida" de un paquete en la red, evitando que, por ejemplo por errores en tablas de rutado, un paquete pueda dar vueltas indefinidamente, sobrecargando el tráfico en la red.
* protocolo: Es un valor que identifica qué protocolo de nivel superior ha pasado los datos a IP para su transmisión. Por ejemplo, ICMP usa el valor 1, TCP el valor 6 y UDP el valor 17. Estos valores se encuentran en la [RFC 1700](http://www.faqs.org/rfcs/rfc1700.html): *Assigned Numbers*.
* checksum de la cabecera: Provee un método simple de detección de errores en la cabecera. Hay una implementación en rc\_funcs.c, llamada calcula\_checksum\_ip ().

Para calcularlo, se debe asumir que el propio campo *checksum* está a cero. Para verificarlo, el checksum de todo, incluyendo el propio checksum, debe ser igual a 0.

* dirección IP origen, dirección IP destino: Direcciones de los hosts que llevan a cabo la comunicación IP, extremo a extremo.
* opciones: Este campo permite añadir más información a la cabecera IP. El tamaño de esta información adicional viene determinado por el campo longitud de la cabecera pues el resto de esta es de tamaño fijo (20 bytes, dejando hasta 40 para las opciones). Las diferentes opciones (record route, timestamp, etc.) y su formato pueden consultarse en la [RFC 791](http://www.faqs.org/rfcs/rfc1700.html).
* datos: Los pasados a IP por una capa superior.

Protocolo

El módulo IP ofrece a los módulos de capas superiores una interfaz para enviar y recibir datagramas.

En el **envío**, analizando la dirección IP del host destinatario se determinará si este se encuentra en la misma red o en otra, en cuyo caso se enviará el datagrama a un router por defecto (ver más abajo como obtener la dirección de este router y la máscara de subred). Se usará el módulo ARP desarrollado en la práctica anterior para obtener las direcciones Ethernet.

En la **recepción**, se usará el campo protocolo de la cabecera IP para dirigir el datagrama recibido a una u otra capa superior.

***IMPLEMENTACIÓN***

Se debe partir de las definiciones incluídas en el fichero de cabecera ip.h, implementándolas en el correspondiente .c, junto con toda aquella función auxiliar, variable o estructura global que se considere apropiada. Para hacer pruebas de este módulo, se deberá usar la implementación de talk.c suministrada junto a este enunciado.

Otras consideraciones:

* La máscara de subred y el router por defecto se deberán obtener de sendas variables locales, llamadas IPMASK e IPROUTER. Estas variables se deben leer y asignar durante la llamada a ip\_inicializa().

**CRITERIOS DE EVALUACIÓN**

* ARP se integra con esta parte (El ARP de la parte anterior pide la dirección MAC) 2 puntos
* Envía paquete IP correctamente (versión, longitud cabecera, longitud total, números de secuencia crecientes y correctos, checksum correcto) 4 puntos
* Recibe y procesa paquete IP de talk correctamente (imprime mensaje por pantalla) 4 puntos

**EXAMEN**

Se realizará un examen INDIVIDUAL.

El examen consistirá en una cuestión sobre la implementación concreta de la práctica por parte de cada pareja de prácticas.

La calificación del examen será APTO/NO APTO. En caso ser NO APTO, la calificación de la práctica para ese miembro de la pareja será de 0 (cero).

**PONDERACIÓN DE CADA PRÁCTICA**

El peso de cada parte respecto al total de las práctica es el siguiente:

* 2A (ARP): 30%
* 2B (IP): 70%