**Arquitectura de Redes 2**

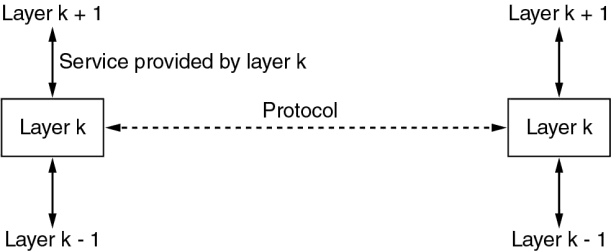
**Descripción de la interfaz de programación de nivel 1**[Introducción](https://moodle.uam.es/mod/resource/view.php?id=705284#INTRO)  
  
[Los servicios del nivel inferior: Nivel 1](https://moodle.uam.es/mod/resource/view.php?id=705284#NIVEL1)  
    [IniciarNivel1](https://moodle.uam.es/mod/resource/view.php?id=705284#INIC)  
   [FinalizarNivel1](https://moodle.uam.es/mod/resource/view.php?id=705284#FIN)  
   [EnviarTramaNivel1](https://moodle.uam.es/mod/resource/view.php?id=705284#ENVIAR)  
   [RecibirTramaNivel1](https://moodle.uam.es/mod/resource/view.php?id=705284#RECIBIR)  
   [EsperaNuevaTrama](https://moodle.uam.es/mod/resource/view.php?id=705284" \l "ESPERA)  
   [ObtenerDirMAC](https://moodle.uam.es/mod/resource/view.php?id=705284" \l "GETIFHWADDR)  
   [ActivarTrazas](https://moodle.uam.es/mod/resource/view.php?id=705284" \l "TRAZAS)   [Función de notificación definida por el usuario](https://moodle.uam.es/mod/resource/view.php?id=705284#auxil).

**Introducción**

Las arquitecturas que definen el marco para realizar la transmisión de la información en las redes de ordenadores, conforme se ha visto en las clases de teoría, se basan en una serie de elementos fundamentales:

* Las distintas funciones que se realizan en un nodo de red para transmitir los datos se agrupan en distintos **niveles** (*layers*).
* Los niveles se agrupan en una estructura jerárquica, en forma de torre.
* En cada nivel se ejecutan procesos o **entidades**.
* Los procesos de nivel **k** se comunican entre sí mediante un **protocolo de nivel k**. La unidad básica de intercambio recibe el nombre de **Unidad de Datos del Protocolo *(Protocol Data Unit, PDU)***.
* Cada nivel proporciona al nivel superior una serie de **servicios**. A estos servicios se accede mediante unas funciones primitivas, que constituyen su ***Application Programming Interface, API***.
* Cada nivel ve la red exclusivamente a través de los servicios que le proporciona el nivel inmediatamente inferior a él.

Estos conceptos se pueden resumir en la siguiente figura:

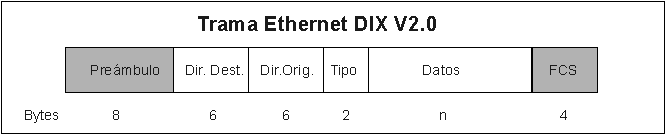
   
(Fuente: Tanenbaum, A., *Computer Networks*, Prentice Hall, 2003. 4ª ed. Pág. 37)

En las prácticas de Redes de Comunicaciones I, el nivel en el que vamos a trabajar es el denominado *Medium Access Control, MAC* de una Red de Área Local Ethernet.

La red Ethernet es una red de *difusión*: cuando un ordenador desea transmitir una información, la envía a un medio que es *común* para todas las estaciones, especificando en los datos transmitidos la estación que debe recibir dicha comunicación. Todas las estaciones conectadas al medio detectan la transmisión, pero sólo la que está especificada como destino del mensaje lo recibe. Para más detalles sobre el funcionamiento de la red Ethernet, consultar la bibliografía de la asignatura (por ejemplo, Tanenbaum, *Computer Networks*, Prentice Hall, 2003. 4ª ed. Pp. 271 y ss).

El nivel MAC es un subnivel perteneciente al nivel de enlace, cuya misión fundamental es garantizar el acceso correcto al medio de comunicación compartido sin que se produzcan colisiones entre las distintas estaciones que desean transmitir.

Dentro de este nivel las Unidades de Datos del Protocolo reciben el nombre de *tramas*. Entre las posibles tramas que se soportan en este nivel vamos a trabajar con la de formato *DIX Versión 2.0*. que tiene la siguiente estructura:

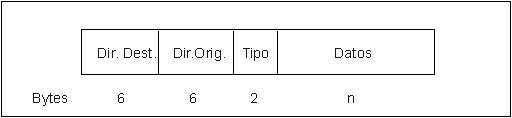


donde se pueden distinguir los siguientes campos:

* **Preámbulo:** Son 8 caracteres que se envían para permitir la sincronización del receptor con el transmisor.
* **Dir.Dest.:**Dirección de la estación que debe recibir la trama.
* **Dir.Orig.:**Dirección de la estación que envía la trama.
* **Tipo:**Campo *Ethertype*. Tipo de trama que se esta usando. Sirve para poder identificar la aplicación destino de la trama dentro de las múltiples que pueden estar transmitiendo datos por la red en el ordenador destino de la misma.
* **Datos:**Información que se transmite por la red.
* **FCS:***Frame Check Sequence*. Es un código de redundancia cíclica (CRC) calculado sobre el resto de la trama.

El tamaño de la trama se considera como la suma de los tamaños de todos los campos. Este tamaño debe ser mayor o igual a 72 bytes y menor o igual que 1526 bytes. Para garantizar el tamaño mínimo puede ser necesario completar los datos a transmitir con caracteres de *relleno*.

En nuestro caso, dentro de los servicios que presta el nivel sobre el que vamos a acceder a la red se encuentra la generación de  los campos que aparecen sombreados en la figura: Preámbulo y FCS. Las primitivas que implementan los servicios de transmisión y recepción intercambiarían sólo, por tanto, los campos representados en fondo blanco. Nos queda de este modo una *trama* con un tamaño mínimo de 60 bytes y máximo de 1514 bytes, que será de la siguiente forma:



La conexión de las estaciones a la red se realiza a través de un adaptador. Cada adaptador lleva grabada internamente una dirección determinada, que es única. Estas direcciones, que como se puede ver en la estructura de la trama son de 6 bytes, se representan por los valores hexadecimales de cada byte, separados por dos puntos. Así, por ejemplo, una dirección Ethernet puede ser la siguiente:

00:C0:F0:40:A7:23

Todas la tramas que circulan por la red cuya dirección destino coincida con la grabada internamente en el adaptador serán recibidas por el mismo, siendo copiadas a su memoria interna . Un caso especial para Dirección Destino es el valor FF:FF:FF:FF:FF:FF, que es la dirección de difusión *(broadcast)*, que se emplea cuando una trama quiere enviarse a todas las estaciones de la red.

La comunicación entre la tarjeta y los programas del usuario se realiza a través del servidor ethd. Ver en la página de [descripción del entorno de trabajo del laboratorio](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-entorno.html) su función y modo de arranque.

**Los servicios del nivel inferior: Nivel 1.**

Como se ha dicho anteriormente, una entidad o programa ve la red de comunicaciones a través de los servicios que le presenta el nivel inmediatamente inferior mediante un conjunto de funciones (API). Mediante el uso de estos servicios puede comunicar unidades básicas de protocolo (PDUs) con otra entidad o proceso de su mismo nivel en otro ordenador perteneciente a la red.

En nuestro caso la red se va a ver a través del nivel que denominaremos **Nivel1**. Las funciones que implementan los servicios de este nivel se encuentran en la librería **libnivel1**, y su definición, en el fichero **/usr/local/include/nivel1.h**, donde también se almacenan algunas definiciones auxiliares.

#define ETH\_OK 0

#define ETH\_ERROR -1

#define ETH\_TIMEOUT -2

typedef unsigned char BYTE;

typedef int (\*tpfNotificacionDesdeNivel1) (int, const BYTE \*);

[int IniciarNivel1 ( int num\_types, const BYTE \*ether\_types,](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#INIC) [tpfNotificacionDesdeNivel1 func\_notificacion, int timer\_intr );](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#INIC)  
  
[int FinalizarNivel1 ( void );](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#FIN)  
  
[int EnviarTramaNivel1 ( int len, const BYTE \*data );](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#ENVIAR)  
  
[int RecibirTramaNivel1( BYTE \* buffer );](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#RECIBIR)  
  
[int EsperaNuevaTrama ( int t );](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#ESPERA)  
  
[int ObtenerDirMAC ( BYTE \* ifhwddr );](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#GETIFHWADDR)  
  
[int ActivarTrazas ( int tipo, char \*fichero );](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#TRAZAS)

El programa de aplicación constará de un módulo principal (main) que incluirá llamadas a estas funciones, según se necesite, y una [función de notificación](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#auxil) que es definida por el usuario:

int Nombre\_de\_Funcion(int , const BYTE \*)

El nivel 1 ejecutará esta función para notificar a nuestro programa de aplicación que se ha recibido una trama. Entonces, el programa de aplicación podrá realizar manipulaciones o validaciones sobre la trama recibida, para aceptarla o rechazarla, según sea necesario.

int IniciarNivel1( int num\_types, const BYTE \*ether\_types, tpfNotificacionDesdeNivel1 func\_notificacion, int timer\_intr);

**ACCIÓN:** Esta función inicia la interfaz con el servidor  ethd, reservando un buffer interno para almacenar las tramas y además indica el tipo de tramas que podrán ser recibidas por el programa. Esta función deberá ser llamada una única vez y antes que el resto de las funciones.

**PARÁMETROS:**

* **int num\_types**: Indica el número de tipo *(ethertype)* de tramas que se habilitan para recepción. Los valores permitidos son:
  + 0: No se recibirán ningún tipo de trama.
  + 1: Se podrán recibir tramas de un único tipo.
  + 2: Se podrán recibir tramas de dos tipos diferentes.
* **const BYTE \* ether\_types**: Puntero a un buffer que contiene los *ethertype*de las tramas que se habilitan para recepción. Deberá haber tantos *ethertype*consecutivos como indique el parámetro anterior.
* **tpfNotificacionDesdeNivel1 func\_notificacion**: Puntero a una [función de notificación](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#auxil), definida por el usuario, que será llamada cada vez que se reciba una trama. Si se pone como valor 0 o NULL no se realizará esta llamada.
* **int timer\_intr**: Representa el periodo de tiempo, en milisegundos, en el que level1 va a llamar a la función de notificación proporcionada por el usuario en el parámetro anterior, aunque no haya llegado ninguna trama. En este caso, la llamada a la función de notificación lleva como longitud de la trama 0, y NULL como puntero al buffer de la trama, para poder distinguir las llamadas debidas a la expiración de este temporizador de las llamadas por llegada de trama. Si en este parámetro se pasa el valor 0 o un valor negativo no se producen llamadas por temporizador. Se garantiza que no se realiza una llamada por temporizador a la función de notificación mientras se está dentro de ella por cualquier caso, para evitar problemas de reentradas.

**VALOR DE RETORNO:**

* ETH\_OK: La operación de inicialización es correcta.
* ETH\_ERROR: La operación falla. En este caso no se deben utilizar el resto de las funciones.  
    
    
    
    
    
    
  **Ejemplo:**

#include "nivel1.h"

BYTE EtherType[ETH\_TLEN\*2]; // Memoria para 2 Ethertypes, de longitud ETH\_TLEN

int nEtherType, rc;

int Notificacion (int longitud, const BYTE \* trama)

{

/\* Proceso a realizar \*/

}

void main(void)

{

... codigo del programa ...

// Preparar array EtherType para recibir tramas de tipos 2098 y 843F

EtherType[0]=0x20;

EtherType[1]=0x98;

EtherType[2]=0x84;

EtherType[3]=0x3F;

nEtherType=2; // Se recibirán 2 Ethertypes distintos

// Llamar a la función de iniciación.

rc = IniciarNivel1(nEtherType,EtherType, Notificacion, 0);

... codigo del programa ...

}

Este ejemplo habilita la recepción de dos tipos de tramas:

* Tramas con Ethertype 2098.
* Tramas con Ethertype 843F.

También se asigna *Notificacion* como la función que se llamará cada vez que se reciba una trama.

int FinalizarNivel1 (void);

**ACCION:** Esta función libera los *buffers*interno reservados en la iniciación del nivel 1 y deshabilita la recepción de tramas. Esta función deberá ser llamada una única vez al finalizar el programa.

**PARAMETROS:** No tiene.

**VALOR DE RETORNO:**

* ETH\_OK: La operación es correcta.
* ETH\_ERROR: La operación falla.

int EnviarTramaNivel1(int len, const BYTE \*data);

**ACCIÓN:** Esta función envía a la red un trama completa (direcciones, *ethertype*, datos).

**PARÁMETROS:**

* **int len**: Indica el tamaño de la trama.
* **const BYTE \*data**: Puntero a la trama que se quiere enviar.

**VALOR DE RETORNO:**

* ETH\_OK: La trama se envió a la red. Esto no garantiza que haya sido recibida por el receptor.
* ETH\_ERROR: La operación fallo.

int RecibirTramaNivel1 (BYTE \*buffer);

**ACCIÓN:**Esta función copia al buffer pasado como parámetro una trama recibida por el nivel 1.

La trama siempre será de alguno de los tipos (*EtherTypes*) indicados al inicializar el nivel 1 en la función IniciarNivel1. En el ejemplo anterior, sólo se recibirán tramas que tengan un EtherType 2098 ó 843F.

Esta función no espera a que llegue una trama. Si no se ha recibido ninguna, devuelve control inmediatamente indicando esta situación a través del valor de retorno.

**PARÁMETROS:**

* **BYTE \*buffer:**Puntero al buffer donde se desea que level1 copie la trama que se ha recibido. **Es responsabilidad del programa que realiza la llamada reservar suficiente espacio de memoria en este buffer para que quepa cualquier trama que se pueda recibir.**

**VALOR DE RETORNO:**

* 0: No se recibido ninguna trama.
* ETH\_ERROR: La operación fallo.
* Otro valor >0: tamaño de la trama recibida.

int EsperaNuevaTrama (int t);

**ACCIÓN:** Espera a que esté disponible una nueva trama para su lectura.

**PARÁMETROS:**

* **int t**: Tiempo máximo, en milisegundos, que se realiza la espera. Transcurrido este tiempo, se retornará de la función aunque no se haya recibido ninguna trama. Si el parámetro tiene valor 0 o negativo, la espera es indefinida.

**VALOR DE RETORNO:**

* ETH\_ERROR: Trama recibida.
* ETH\_TIMEOUT: Salida por Time Out.

int ObtenerDirMAC (BYTE \*ifhwaddr);

**ACCIÓN:** Esta función devuelve la dirección MAC local del adaptador que se está utilizando.

**PARÁMETROS:**

* **BYTE \*ifhwaddr**: El parámetro pasado, a la salida de la función, contendrá la dirección MAC solicitada. Se debe pasar un puntero a BYTE que tenga reservada memoria para contener un mínimo de 6 bytes.

**VALOR DE RETORNO:**

* ETH\_OK: Operación realizada correctamente.
* ETH\_ERROR: La operación fallo.

int ActivarTrazas (int tipo, char \*fichero);

**ACCIÓN:** Activa la producción de trazas de la operación del nivel 1..

**PARÁMETROS:**

* **int tipo**: Tipo de trazas que se pide. Los tipos válidos, y sus significados, son los siguientes:
  + 0: Suprimir las trazas previamente activadas.
  + 1: Trazas de tramas que se envían y se reciben.
  + 2: Todas las trazas del tipo 1, mas las llamadas a funciones de Nivel 1.
  + 3: Todas las trazas del tipo 2, y las de comunicación con ethd.
* **char \* fichero**: Nombre del archivo donde se desea realizar las trazas. No tiene por qué existir, en cuyo caso se crea. Acepta como valores posibles "*stdout"* y "*stderr"*(cadenas de caracteres), para producir la salida, respectivamente, sobre *stdout*y *stderr*(*handlers*de los archivos de entrada/salida)*.*

Se recomienda utilizar los tipos de trazas 1 o 2 durante la fase de depuración de los programas.

**VALOR DE RETORNO:**

* ETH\_OK: Operación realizada correctamente.
* ETH\_ERROR: La operación fallo.

Función de notificación definida por el usuario

Las funciones anteriores definen las primitivas del nivel 1. Como extensión a este nivel, el programa de aplicación deberá proporcionar una función de notificación, que por tanto deberá estar programada dentro de alguno de los módulos que compongan el programa de aplicación, y a la que el nivel 1 llamará cada vez que se reciba una trama. Esta función de notificación tendrá como definición:

int Nombre\_de\_funcion (int , const BYTE \*)

Que es consistente con la declaración del tipo tpfNotificacionDesdeNivel1, presentada anteriormente.

Esta función será llamada cada vez que se reciba una trama.

**PARAMETROS:**

* El primer parámetro indica el tamaño de la trama recibida. Si se recibe 0, quiere decir que no se ha recibido ninguna trama, y la llamada a la función se ha producido como resultado de la expiración del intervalo de tiempo especificado en el parámetro 4 de la función [IniciarNivel1](http://arantxa.ii.uam.es/%7Erc2lab/html/rc2-pract-nivel1.html#INIC).
* El segundo parámetro indica el *buffer*donde está depositada la trama recibida. En caso de haber recibido como tamaño 0, este puntero será NULL.

Aunque esta función ya recibe en el *buffer*pasado como parámetro, la trama recibida y el tamaño de la misma, normalmente no se debe emplear esta función para recibir los datos en el programa de aplicación (aunque se  harán excepciones a lo largo del curso). Esta pensada para realizar comprobaciones o chequeos en la trama sin modificarla, aceptando o denegando la validez de la trama recibida. Esta situación se indica a través del código de retorno:

**VALOR DE RETORNO:**

* ETH\_OK: La trama se acepta.
* ETH\_ERROR: La trama debe rechazarse.

Puesto que retornar el valor ETH\_ERROR hace que el nivel 1 considere la trama como no válida y la descarte para su posterior proceso, este valor también podría utilizarse en caso de que la función de notificación desee realizar el proceso de almacenamiento de la trama por sí misma.