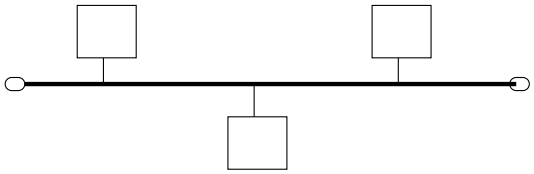
libro abierto / serie apuntes

Pablo Ruiz Múzquiz

Redes y sistemas de comunicación

0.6.0



▷ Un libro libre de Alqua

RSC	Redes y sistemas de comunicación	004.738 ALQ

† lomo para ediciones impresas

Dedicado

A mi hermana Blanca

http://alqua.org/libredoc/RSC

Pablo Ruiz Múzquiz pablo@alqua.org http://alqua.org/people/pablo

Redes y sistemas de comunicación

versión 0.6.0
15 de abril de 2004

alqua, madeincommunity

copyleft

Serie apuntes

Área informática

CDU 004.738

Editores

Pablo Ruiz Múzquiz pablo@alqua.org

Notas de producción

Plantilla `latex-book-es-b.tex, v. 0.1` (C) Álvaro Tejero Cantero.

▷compuesto con software libre◁

Índice general

1 Modelos de referencia

Hemos llegado al estado actual en donde la transmisión de información se produce a escala mundial todos los días porque en el siglo XX el tratamiento y gestión de la datos sufrieron un empuje extraordinario y porque el desarrollo de las computadoras fue impresionante. Aparecieron n-formatos en los que esa información podía estar contenida (telefonía, radio/TV, dispositivos magnéticos, etc). Desde que en los comienzos se impuso la arquitectura de *Mainframe* se ha ido evolucionando hasta descentralizar los sistemas de cálculo y potenciar las comunicaciones entre ellos[1] hasta crear lo que conocemos como **red**. Como consecuencia inmediata de este paso, el tratamiento de la información se ve en la necesidad de una unificación para organizar mejor y ser más eficiente.

Veamos un par de definiciones que nos ayudarán a aclarar conceptos de partida:

Red de computadoras es una colección interconectada de computadoras autónomas. Dos computadoras se consideran interconectadas cuando son capaces de intercambiar información y este trasvase no se realiza en un marco de dependencia Maestro-Esclavo. Es decir, en una red de computadoras, todas ellas son autónomas y podrían trabajar por sí solas si fuese necesario.

Sistema distribuido es una red de computadores gestionadas de tal manera que la red global se le esconde al usuario, que trabaja con ella de forma transparente[2].

Las redes de computadoras aportan a la empresa el poder compartir recursos y usuarios, una alta fiabilidad y el abaratamiento de costes mientras que al usuario le permite, sobre todo, el acceso a sistemas remotos.

1.1. Clasificación de redes

No existe una clasificación definitiva pero lo intentaremos mediante dos criterios generales.

1. Según la tecnología de transmisión usada

   a) Redes de difusión (también llamadas *broadcast*)
   Comparten un canal o un medio de difusión y se caracterizan por la gestión del medio compartido y el direccionamiento de la información. La ventaja inmediata es que podemos enviar información a muchas computadoras a la

[1] Entre otros motivos, el omnipresente económico
[2] Un buen ejemplo es el sistema de exportación de directorios de UNiX, el Network File System **NFS**.

vez gracias al mecanismo de *broadcast*. Este tipo de redes utiliza una topología válida para redes pequeñas ya que es vital que el retardo sea pequeño para evitar que haya colisiones entre transmisiones de diferentes computadoras[3]. Véase la figura 1.1.

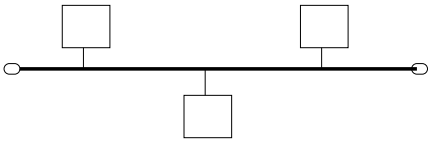


Figura 1.1: Esquema de red de difusión

b) Redes punto a punto

Se caracterizan por crear canales dedicados entre máquinas mediante el uso intensivo del *enrutamiento*. Una vez creado el canal de comunicación, los nodos intermedios sólo se ocupan de reenvíar los datos al nodo marcado por el camino. Véase la figura 1.2 para un detalle gráfico.

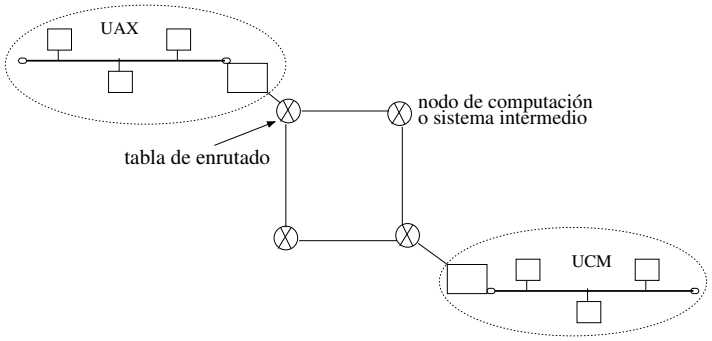


Figura 1.2: Esquema de una red Punto a Punto unida a dos redes de difusión

2. Según la escala

a) Redes de área local (LAN)

- Longitud: 10m-1km
- Velocidad de transmisión: 10Mbps, 100Mbps, Gbps.
- Modelo de transmisión: Generalmente se suele usar el modelo de red de difusión.

[3] Si usáramos esta arquitectura de red para una red interoceánica muy probablemente varios ordenadores a la vez creerían tener el canal libre ya que el *net-lag* no sería despreciable lo que provocaría un caos de colisiones entre diferentes informaciones enviadas.

- Topología: Lo más sencillo sería un bus, pero también es posible una de estrella (equivalente a una en forma de bus[4]) y otra de anillo (topológicamente diferente de las otras dos)

b) Redes de área metropolitana (MAN). Son muy similares a las LAN pero las clasificamos en un grupo aparte porque tienen un estándar definido para ellas. Este *estándar* es la norma 802.6 (IEEE). Se denominan también Bus Local de Área Distribuida (DQDB). Para más información, consultar el capítulo 4 del Tannembaum.

- Longitud: 1km-10km
- Topología: Definida por el estándar. Está formanda por dos buses unidireccionales (simplex), en donde las máquinas están conectadas como se muestra en la figura 1.3. Si una máquina decide emitir a una máquina situada a un lado concreto utilizará la interfaz de red correspondiente (en la figura 1.3 las hemos representado por pequeñas flechas que salen de los hosts). Lo mismo se aplica para recibir información de otros hosts.

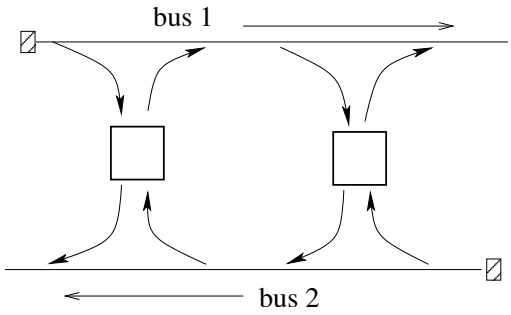


Figura 1.3: Esquema de una red metropolitana

c) Redes de área extensa (WAN). Están compuestas principalmente por líneas de transmisión del tipo que sean y dispositivos de conmutación. En esta clasificación encajan bien las redes punto a punto. En este tipo de redes los nodos de conmutación suelen ser máquinas dedicadas[5].

- Longitud: >10km

1.2. Técnicas de conmutación

Conmutar El procesamiento que realiza un nodo que recibe información de una línea por una determinada interfaz y la reenvía por otra interfaz, con el objetivo de que llegue a un destinatario final (direccionamiento).

[4] un simple cable, generalmente coaxial, que une a todos los ordenadores.
[5] En este caso, entendemos como máquinas dedicadas aquéllas que están expresamente diseñadas a nivel *hardware* para ejercer una función determinada.

Los criterios para conmutar son múltiples y los veremos más adelante pero en relación a las técnicas de conmutación sabemos que hay dos básicas.

1. Conmutación de circuitos. Se caracteriza por reservar un camino dedicado para una determinada comunicación entre un terminal origen y un terminal destino. Siendo **camino** un *canal lógico* o *capacidad determinada de la red*, que puede pertenecer tanto a una línea dedicada como a una compartida, y que en cada enlace va saltando de nodo a nodo desde el origen hasta el destino.

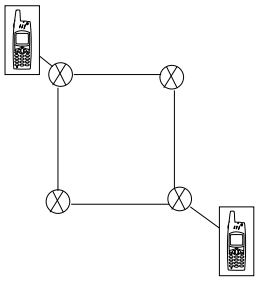


Figura 1.4: La red telefónica no es más que una red Punto a Punto

Ejemplo La línea de teléfono es un claro ejemplo de conmutación de circuitos. La línea no es un cable físicamente reservado (aunque en algunos tramos sí lo sea) sino que se reparte la capacidad del canal. Para esto existen dos técnicas, FDM (multiplexación por división de frecuencia) y TDM (multiplexación por división de tiempo).

Existen varias fases en este proceso

a) Establecimiento de comunicación al nodo más cercano. Éste nodo realiza una conmutación de la transmisión hacia el siguiente nodo más cercano, etc. Finalmente llegamos al nodo destino, que devuelve un asentimiento.

b) Transmisión de la información en modo Full-Duplex (simultáneamente y en los dos sentidos).

c) Liberar recursos de camino para cerrar el circuito. Puede cerrarlo quien desee de los dos terminales.

En este tipo de comunicación existe cierta ineficiencia ya que hay vacíos de información en el circuito que no se aprovechan. Sin embargo, una vez establecido el circuito (se pierde algo de tiempo para calcular el camino óptimo) los tiempos de retardo son muy cortos. Como restricción importante podemos decir que es obligatorio que ambos terminales emitan a una **misma velocidad**.

Este tipo de transmisión es bueno para unos servicios (voz) y malo para otros (cuando es necesario el uso intensivo de las capacidades de la línea).

2. Conmutación de paquetes. Los mensajes se parten en unidades más pequeñas denominadas *paquetes*. Ahora los enlaces físicos no se reparten de forma estática sino de forma dinámica y se van asignando recursos según se van necesitando. De esta forma, los nodos se convierten en unidades más complejas que *gestionan colas* y según vayan teniendo capacidad van enviando paquetes a diferentes nodos.

   Las características fundamentales de este tipo de conmutación son

   a) Eficiencia: si no hay transmisión de una comunicación, no se reserva ningún recurso y éste queda libre para otra comunicación.

   b) Los terminales finales no están obligados a transmitir a idénticas velocidades. Los nodos intermedios se encargan de adecuar velocidades (concepto de *buffer*[6]).

   c) No existe una saturación propiamente dicha (aunque el recurso físico puede llegar a provocarla), sino un aumento del retardo en la transmisión de paquetes. A más demanda, mayor tiempo de retardo.

   d) No todos los paquetes son iguales (hay algunos de ellos que tienen más prioridad que otros) y eso los nodos lo sabrán gestionar.

   Debido al tipo de tráfico de computadoras (*a ráfagas*: con tiempos muertos y tiempos de transmisión) este sistema de conmutación es el más adecuado para ellas ya que no es tan importante que el tiempo de retardo sea alto. Sin embargo, las necesidades de la industria comienzan a mirar a las redes de conmutación de paquetes para servicios tradicionalmente asociados a conmutación de circuitos. Esto implicará que haya que identificar estos servicios de alguna forma para que los nodos entiendan que estos paquetes *nuevos* presentan una gran prioridad frente a otros. Esta gestión del ancho de banda es muy complicada pero se intenta priorizar el tráfico gracias a una correcta identificación de paquetes o *agrupaciones* por parte de grandes nodos pseudointeligentes (en el apartado de software aún se está muy verde y las protoimplementaciones se realizan actualmente mediante soluciones hardware).

1.3. Jerarquía de protocolos y arquitectura de red

Evidentemente, es necesario que todos los terminales utilicen el mismo protocolo para comunicarse. Como esta tarea parecía inabarcable se ideó una estructura de niveles, cada uno de los cuales está especializado en una función, confiando en que esta disección del problema ayudase a implementar una arquitectura viable.

Cada una de las entidades pares (extremos asociados) que han intervenido en la comunicación lo han he hecho mediante un protocolo (Véase la figura 1.5). Una capa inferior

[6] Si un terminal emite a menor velocidad que el otro, el nodo intermedio *acumulará* datos para realizar envíos mayores. Si se da el caso contrario, el nodo intermedio fragmentará lo recibido en otros pedazos y los enviará espaciados en el tiempo.

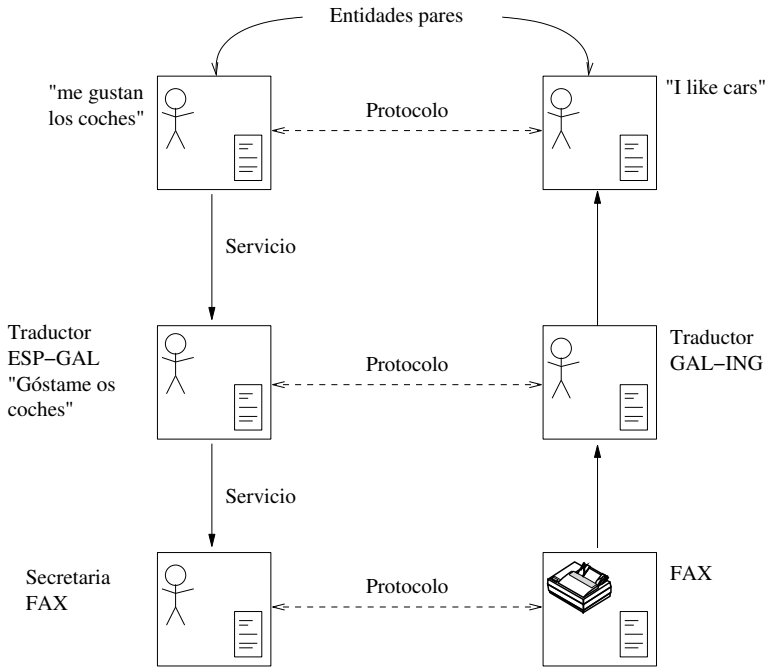


Figura 1.5: Esquema de una estructura por niveles

de la transmisión ofrece un servicio a la capa superior. Toda capa es usuaria de la abajo y proveedora de la de arriba, siendo el número de capas N. Como puede apreciarse, el objetivo es dividir la complejidad.

Vamos a definir una arquitectura de red como *el conjunto de protocolos y las capas que nos permiten la comunicación entre máquinas.*

Otras dos definiciones importantes son:

Protocolo es un formato de mensaje más una regla de intercambio de ese mensaje entre entidades pares.

Servicio es la capacidad de comunicación que ofrece una capa inferior o proveedora a una capa superior o usuaria.

Según vamos subiendo en las capas, las funcionalidades se van volviendo más potentes y menos relacionadas con la comunicación en sí (es decir, la abstracción va aumentando). Por otro lado, en lo más bajo de la estructura a capas tenemos el medio físico, que es aquel medio sobre el que se realiza la comunicación.

> El diseño de mi torre de protocolos *debería* ser de tal forma que el cambio de unos de los protocolos no afecte a la comunicación. En la práctica sucede pocas veces pero los protocolos diseñados de esta manera disfrutan de una mejor *calidad de vida*. Veremos más sobre esto en referencia al modelo OSI y al modelo actual.

1.3.1. Terminología de unidades de datos

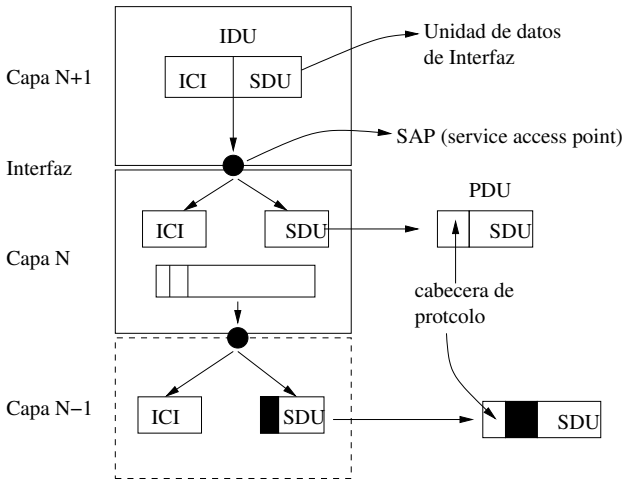


Figura 1.6: Esquema de capas

La capa N ofrecerá un servicio a la capa $N+1$ a través de una interfaz que contiene un **SAP** (Service Access Point). Los servicios de la capa N, por tanto, están disponibles a través del SAP (que tiene asignada una dirección) para la capa $N+1$.

Llamaremos *unidad de datos de interfaz* **IDU** a lo que la capa superior envía al SAP para requerir el servicio de comunicación. Esta IDU está compuesta por la ICI (información de control de interfaz, que lleva el control del envío) y la SDU (Unidad de datos del servicio, lo que propiamente se desea enviar).

En la capa inferior tenemos la ICI y la SDU enviados. A la SDU le colocamos una cabecera específica de protocolo. A esa suma de SDU y cabecera de protocolo la designamos con PDU (unidad de datos de protocolo).

Ejemplo En una red telefónica el SAP sería la roseta de la pared con dirección el número de teléfono.

Las capas proveedoras pueden ofrecer dos tipos de servicios a las capas usuarias superiores

1. Servicio orientado a conexión (CO). Son aquellos servicios en los que hay un establecimiento previo de conexión antes de la transmisión de datos y desconexión. Es una idea similar al funcionamiento de la conmutación de circuitos pero orientada a los servicios.

2. Servicio no orientado a conexión (CL). Son aquellos servicios en donde no existe una conexión entre terminales.

   Ejemplo El envío de una carta por correo postal, que se deja en el buzón y uno *se olvida* de ella.

De hecho, lo primero que se introduce no tiene por qué ser lo primero que llega, todo lo contrario que en CO.

Fiabalidad de un servicio Un servicio es fiable cuando éste garantiza la correcta entrega de la información. Esto se traduce en la necesidad de implementar lo siguiente: *el receptor notifica el haber recibido el mensaje (asentimiento)*. En general, los servicios CO son fiables mientras que los CL no lo son.

Ejemplo Si estamos sirviendo páginas de un libro para que sean leídas, deberían dar un servicio CO para que las páginas lleguen por orden y el servicio sea fiable dado que parece fundamental recibir el OK del receptor tras cada página leída.

Ejemplo Servir vídeo digital bajo demanda. Ha de ser CO para poder asegurarse la proyección secuencial pero no queremos que sea fiable ya que el tráfico generado *por asentimiento* no es rentable al consumir una cantidad significativa de ancho de canal y nadie desea ver, de pronto, fotogramas pasados que fueron enviados incorrectamente en su momento.

Ejemplo Correo electrónico. Es un claro ejemplo de servicio CL. "Ya llegará". En principio no es fiable pero se puede implementar un acuse de recibo y pasaría a ser fiable.

1.3.2. Primitivas de servicio

Un servicio ¿cómo se implementa formalmente? Con un conjunto de operaciones que denominaremos *primitivas de servicios*. Estas primitivas son como llamadas al servicio demandando funciones o acciones, o bien informes de acciones ya realizadas[7].

Existen cuatro tipos de primitivas de servicio (para un ejemplo gráfico véase la figura 1.7)

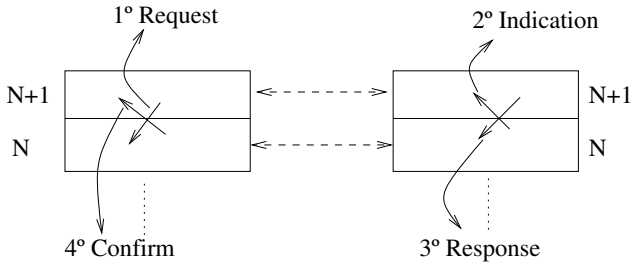


Figura 1.7: Esquema de un request-indication-response-confirm

1. Peticiones (**request**): Una entidad le pide al servicio que haga algo. "Deseo conectarme".

[7] Estaría bien realizar una analogía entre estas primitivas y las primitivas de sistema, que encontramos en los sistemas operativos. ¿Comparten la atomicidad?

2. Indicaciones (**indication**): Se informa a una entidad que ha sucedido algo (generalmente, una petición). “Alguien desea conectarse”.

3. Respuestas (**response**): La respuesta de una entidad a un suceso. “Dile que le he oído”.

4. Confirmaciones (**confirm**): Informar a un entidad de que ha llegado una respuesta de un petición suya anterior. “El de allí me dice que ha recibido el mensaje”.

Un servicio fiable utilizará las cuatro primitivas mientras que los no fiables sólo emplean los dos primeros.

Ejemplo Queremos un servicio orientado a conexión que tendrá las siguientes 8 primitivas divididas en la parte de conexión CONNECT, la de transmisión de información DATA y la de desconexión DISCONNECT.

1. CONNECT.request
2. CONNECT.indication
3. CONNECT.response
4. CONNECT.confirm
5. DATA.request
6. DATA.indication
7. DISCONNECT.request
8. DISCONNECT.indication

El usuario de la computadora 1 envía desde la capa $N+1$ un **request** y la computadora 2 recibe un **indication**, devuelve un **response** y la computadora 1 recibe un **confirm**.

Para entender el proceso entero, nos hemos valido de una supuesta conversación entre dos personas mediante intermediarios (véase la figura 1.8)

1. Quiero saber si estás ahí [1]
2. Quieren saber si estás ahí [2]
3. Diles que estoy aquí [3]
4. Me dicen que te diga que está ahí [4]
5. Quiero que venga a cenar [5]
6. Me dicen que él desea que vayas a cenar [6]
7. Deseo desconectar [7]
8. Desea cortar la comunicación [8]

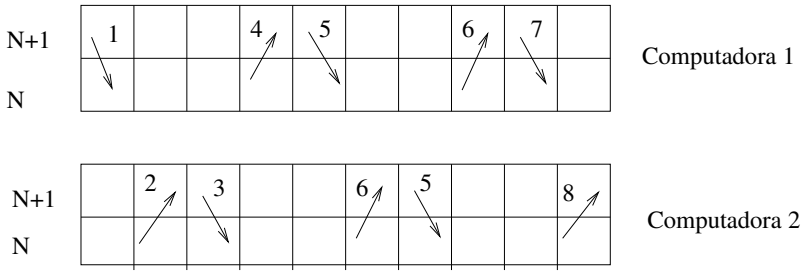


Figura 1.8: Esquema de las capas en el ejemplo de una conversación con intermediarios

1.4. Diseño de capas para desarrollar una arquitectura de red

Los siguientes elementos son necesarios para la correcta especificación de una red

1. **Direccionamiento**: Básicamente, nos proporciona direcciones de interfaces. Cuanto más subimos en el protocolo más subirán las funcionalidades y menos tendrán que ver con la comunicación física en sí.

2. **Transferencia de información**: Debemos fijar reglas de transferencia de información (en una dirección -simplex-, ambas -half-duplex-, ambas simultáneamente -full-duplex-). Generalmente, la transmisión será siempre full-duplex salvo en capas inferiores y casos puntuales en donde el medio físico no aguante ese sistema.

3. **Detección y corrección de errores**: Serán útiles *códigos redundantes* para detectar errores u obligar a una *respuesta,* para conseguir los llamados *códigos autocorregidos.*

4. **Numeración de paquetes**: Es necesario para la reconstrucción de la transmisión por el receptor de forma consistente.

5. **Mecanismo de control de flujo**: Sirven para adecuar las diferentes velocidades de los interfaces de red. Veremos varios de estos mecanismos.

6. **Mecanismos de control de congestión**: Sirven para evitar la saturación de los nodos intermedios.

7. **Mecanismo de segmentación y concatenación**: En ocasiones las diferentes capas de la arquitectura no soportan tamaños iguales de PDU de forma que será necesario proveer a nuestra red con herramientas de división de la PDU en varias SDU y a la inversa para reconstruir esa segmentación en la entidad par. Ya más raro resulta encontrar la necesidad de concatenar PDUs pequeñas en una grande pero es de gran utilidad para reducir la información de control y aprovechar al máximo los recursos del canal.

8. **Multiplexación y División**: Dividir los recursos que tiene un canal entre diferentes comunicaciones (Multiplexación) o separar diferentes comunicaciones para que vayan en diferentes canales (División). La multiplexación normalmente exige un servicio CO tanto en la capa usuaria como en la capa proveedora. No confundir Multiplexación con segmentación. Evidentemente, habrá que poder identificar correctamente las diferentes comunicaciones.

1.5. Modelo OSI

OSI quiere decir Open System Interconnection y fue definido por la ISO (International Organization of Standards) en 1983[8].

A finales de los años 60, y partiendo del entorno militar, se pensó en crear redes de conmutación de paquetes tolerante a fallos capaces de viajar por medios diferentes. Este planteamiento llegó al entorno universitario y se produjo un crecimiento rápido de sistemas de este tipo por lo que fue necesario definir un estándar para poder especificar cualquier arquitectura de red.

La propuesta OSI fue la siguiente:

Para un terminal o *host* propusieron una estructura de 7 capas como puede apreciarse en la figura 1.9.

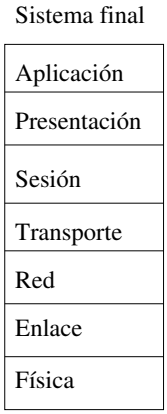


Figura 1.9: Esquema de una estructura de 7 capas para el sistema final de OSI

mientras que para un nodo intermedio propuesieron una estructura de 3 capas orientado al enturtamiento (figura 1.10) .

Además, se adjuntaron una serie de recomendaciones o reglas a seguir[9]:

1. Se debe crear una capa siempre que se necesite un nivel de abstracción diferente.

[8] Sí, las mismas siglas para todo. Esperamos que el contexto ayude.

[9] Es interesante comprobar cómo los consejos aquí enumerados guardan estrecha relación con aquellos propuestos en el desarrollo orientado a objetos en la programación. Para más información a este respecto puede consultarse [1].

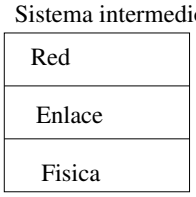


Figura 1.10: Esquema de una estructura de 3 capas para el sistema intermedio de OSI

2. Cada capa ha de realizar una función bien definida.
3. Las capas han de implementarse de tal manera que el flujo de información a través de los interfaces de capa sea el mínimo posible.
4. Las capas deben ser suficientes en número para que no exista la posibilidad de agrupar demasiadas funcionalidades en una capa individual pero asimismo no deben ser tantas que la complejidad de la arquitectura de red se dispare. Tan malo es sobrecargar una sola capa con funciones como pasar a tener muchas capas que nos dificulte la implementación.

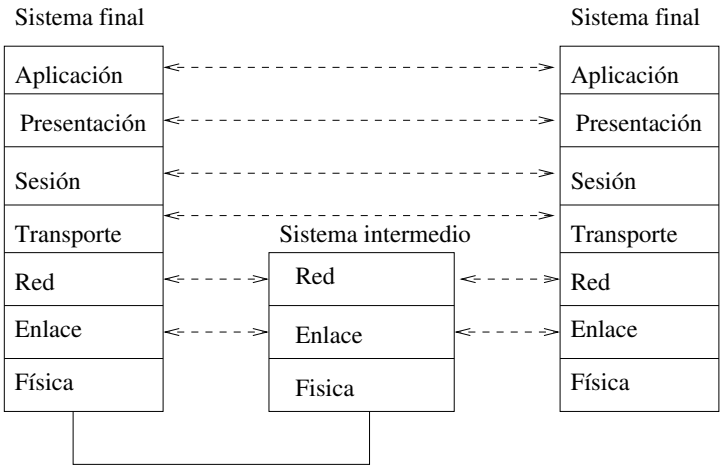


Figura 1.11: Esquema de comunicación entre terminales de OSI

1.5.1. Funciones de las capas

Veremos aquí con más detalle cada una de las capas del diseño propuesto por OSI.

- Nivel físico: El medio físico será el encargado de transmitir los bits (la PDU *será* en este caso el bit). El diseño ha de ser fiable de forma que si se mete un 1, al otro lado haya un 1 y no exista una degradación excesiva. Los parámetros que entran en juego en este nivel son:

- Voltaje que se asocia al valor lógico 1, y el voltaje asociado al valor lógico 0 (Véase la figura 1.12). Si el medio físico no es lo suficientemente fiable para poder distinguir los dos voltajes, no sabremos reconstruir correctamente la información.
- Periodo que dura el valor lógico. A menor periodo mayor aprovechamiento del canal (aunque no conviene forzar los límites de nuestros sistemas de medición al otro extremo).
- Tiempo de establecimiento de conexión, entendiendo conexión al nivel más elemental.
- Número de pines que tienen los cables.

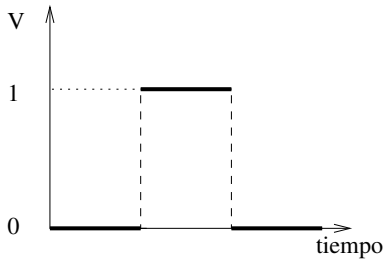


Figura 1.12: Esquema de los valores lógicos y sus voltajes correspondientes

- Nivel de enlace: Se monta directamente encima del nivel físico y transforma los 1 y 0 que le llegan en una secuencia de bits libre de errores para enviarlo al nivel de red.

  - Una función principal es detectar el inicio y fin de una PDU de ese nivel. A estas PDUs las llamaremos **Tramas**. Ese trabajo sobre las tramas consiste básicamente en **sincronizar**.
  - Realiza una importante detección de errores a nivel de trama y tendrá mecanismos de retransmisión[10].
  - Adecua velocidades entre equipos.
  - En el caso de medios compartidos es el que gestiona quién puede acceder al medio en cada momento.

- Nivel de red: Es la primera capa que ofrece un **servicio** extremo a extremo *pero* no es un protocolo extremo a extremo (ya que no conecta directamente dos entidades pares). Permite enviar información de una máquina origen a otra destino. Sus funciones principales son:

[10] a este nivel aún no podemos implementar mecanismos de autocorrección.

  - Su función principal es encaminar la información. A las PDU del nivel de red las llamaremos **paquetes**.
  - Se encarga del control de congestión de paquetes.
  - Se encarga del control de ordenación de paquetes.

- Nivel de transporte: es el primer **protocolo** extremo a extremo. Para este protocolo, al contrario de los que veníamos viendo, las entidades pares están en los dos terminales de la conexión.

  - Realiza la tarea de multiplexación de forma transparente a la capa de sesión.
  - Debería ofrecer diferentes tipos de servicio a las capas superiores en base a poder dar calidad de servicio **QoS**. No será lo mismo una videoconferencia que enviar un correo electrónico y esta capa deberá estar preparada para acomodarse a estos cambios.

- Nivel de sesión: es el encargado de la gestión del diálogo y la sincronización de las transferencias.

- Nivel de presentación: soluciona los problemas inherentes a la distinta forma de presentar la información en los terminales origen y destino. Para ello define un lenguaje de definición de interfaces abstractas, ASN-1, y unas reglas de codificación no ambigua (reglas de encodificación) de valores definidos por ASN-1, a esto se le conoce como **BER** (base encoding rules). Hoy en día las aplicaciones se encargan de esta capa.

- Nivel de aplicación: Cualquier aplicación que requiera comunicarse de forma remota. Software, en definitiva.

1.5.2. Defectos de OSI

Es un muy buen modelo didáctico y ayuda a clarificar los conceptos pero su viablidad práctica es cuestionable debido principalmente a estos factores:

1. Complejidad a la hora de implementar/especificar los protocolos. El control de flujo y sincronización es difícil de concretar.
2. Planificación: tiene una mala planificación.
3. Mala tecnología. Las capas no están bien dimensionadas. Las capas de sesión y presentación están casi vacías.
4. Se olvida totalmente de los servicios no orientados a conexión CL (debido, seguramente, al año en el que se creó).
5. Mala política debido a que la OSI fue visto siempre como imposición (al contrario que TCP/IP que venía de un entorno universitario y que estudiaremos en la siguiente sección).

1.6. Introducción a TCP/IP

A diferencia de OSI, no había un plan de trabajo definido y se va desarrollando a partir de *parches*. A pesar de tener limitaciones, es estable, econónimo y fácil de implementar. Se trata de una arquitectura de red denominada *best-effort* (proporciona un servicio según puede, no podemos exigirle nada).

Son principales características pueden resumirse en:

- Se fusionan varias capas de OSI.
- TCP es el principal protocolo de la capa de transporte y se utiliza cuando se desea realizar un servicio fiable. UDP es otro protocolo utilizado para conexiones no fiables ya que no espera respuesta a los datagramas enviados (será útil en entornos con tasa de error mínima).
- En el nivel de red el principal protocolo es el IP (se encarga de llevar los datos). Otro es el ICMP (protocolo de mensajes de control sin datos e información).

1.6.1. Defectos de TCP

- No oculta bien los protocolos, no son independientes del servicio. Un ejemplo claro de este fallo es el problema que está resultando sustituir IPv4 por IPv6.
- Carece de capacidad para acondicionar las exigencias actuales de los servicios a la propia red.

1.7. Elementos o dispositivos de interconexión

1. A nivel físico, tendremos los repetidores o Hubs.
2. A nivel de enlace, tendremos los Bridge/Switch.
3. A nivel de red, tendremos los Routers.
4. A nivel de aplicación, Gateways y Proxies.

Antes de seguir es importante detenernos en un concepto muy importante.

Dominio de colisión la región de la red que comparte un medio de difusión.

Ha de quedar claro que un medio físico no afecta para nada al dominio de colisión.

Sin embargo, un dispositivo a nivel de enlace (Switch o Bridge) es capaz de lidiar con las colisiones de forma que sí segmenta el dominio de colisión.

Dominio de emisión La región de una red delimitada por dispositivos de interconexión de nivel de red (routers). Está muy relacionado con la idea de *broadcast* (de hecho a veces se les llama dominio de broadcast). Véase la figura 1.15. Son los routers los que segmentan los dominios de emisión.

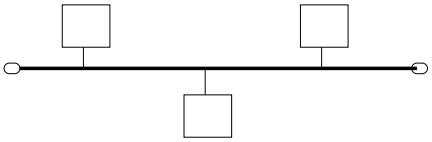


Figura 1.13: El bus de una red de difusión es un dominio de colisión

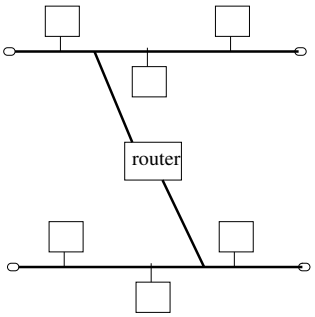


Figura 1.14: El router no afecta en absoluto al dominio de colisión porque no es un dispositivo a nivel físico

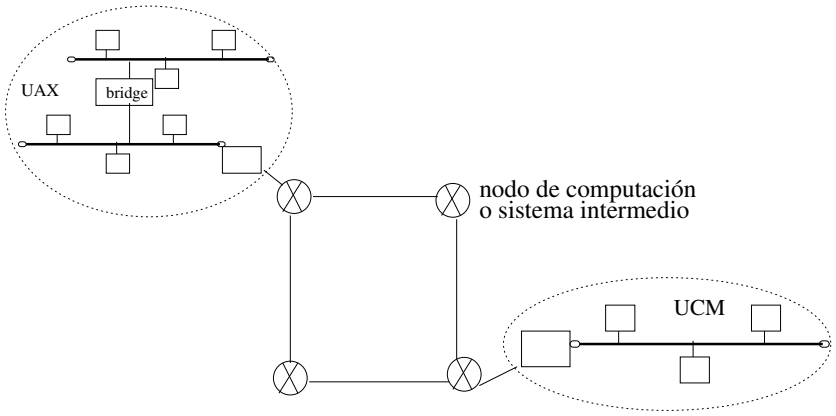


Figura 1.15: El bridge no afecta al dominio de emisión.

2 Nivel de enlace

Las funciones principales del nivel de enlace ya las vimos en el tema anterior, aquí trataremos los mecanismos de acceso al medio de que dispone antes de hablar sobre diferentes estándares utilizados.

Dentro del nivel de enlace disponemos de varios subniveles. El subnivel MAC es el que se encarga de todas las funciones que tengan que ver con el medio físico y tiene bastante sentido en las redes de difusión. El llamado LLC tiene como misión ofrecer una visión unificada del nivel de enlace a la capa superior, en este caso *de red*. La capa lógica LLC independiza la forma de gestionar el acceso (CSMA/CD, Tokeng Ring, Token Bus) de la capa de red. Véase la figura 2.1.

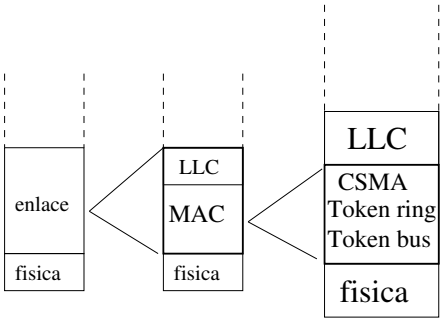


Figura 2.1: Esquema del nivel de enlace

Para poder centrar nuestro estudio del nivel de enlace es importante señalar que el estándar IEEE 802 (también conocido como ISO 8802) cubre los aspectos que han de cumplir tanto el nivel de red como el de enlace.

Dentro de IEEE 802 hay diferentes subestándares:

- 802.1 se encarga de definir las primitivas de los enlaces.
- 802.2 describe la capa lógica LLC
- 802.3 describe el estándar utilizado en redes Ethernet.
- 802.5 describe el estándar utilizado en redes Token Ring.
- 802.11b describe el estándar utilizado en redes WLAN (tan de moda últimamente con las redes *wireless*).

Una vez visto por encima el asunto de los estándares nos hacemos la siguiente pregunta ¿Cómo podemos clasificar los mecanismos de control de acceso al medio? Está claro que en un medio compartido será necesario un sistema *ordenado* de forma que no todas las máquinas salgan a enviar datos cuando les apetezca. Veremos, a continuación, varias formas de encontrar un compromiso entre las máquinas de una red para que todas puedan enviar y recibir de forma aceptable.

2.1. Mecanismos de acceso al medio

Existen dos tipos mecanimos a estudiar; los estáticos y los dinámicos, divididos a su vez en varios subtipos.

2.1.1. Estáticos (TDM,FDM)

Tenemos un **medio físico** que hay que **compartir**. Una forma de hacerlo es repartir ese medio físico y hacerlo de forma estática. Por ejemplo, podría ser una división en tiempos siempre igual o una división en frecuencias acotadas[1]. Si las necesidades de la red son cambiantes en el tiempo, esta forma de controlar el acceso al medio no es conveniente[2]. No hablaremos de ellos más ya que no se utilizan en las redes de computadores.

2.1.2. Dinámicos

Los estudiaremos a fondo ya que son los que se utilizan en la práctica (aunque nos detendremos en la evolución que han sufrido en estos años).

1. Mecanimos de contienda:

   a) ALOHA: surgió en los años 70 en la Universidad de Hawaii ya que deseaban resolver el acceso al medio en un entorno de radiofrecuencia. Querían acceder de forma no coordinada al medio (es decir, sin establecer reservas de canal). Dentro de ALOHA encontramos dos tipos:

      1) ALOHA puro:
         - Cada entidad accede al medio cuando desea.
         - Las entidades sólo escuchan tras transmitir.
         - Es necesario terminar una trasmisión antes de empezar la siguiente.
         - El terminal que ha enviado la trama puede saber si ha habido destrucción por colisión (ej: dos terminales enviaron tramas al mismo tiempo y ambas se destruyeron al ser imposible intentar autoexcluirse mutuamente).

[1] Por ejemplo, la máquina A sólo envía en los minutos pares, mientras que la máquina B lo hace en los minutos impares.

[2] Se podría hablar de una similitud conceptual con los mecanimos FCFS en los algoritmos de planificación de procesos. Véase [2].

- Si la trama es destruida, espera un tiempo aleatorio (para evitar nuevas colisiones) y vuelve a transmitir.
- Modelado estadístico del medio: Supondremos infinitos usuarios y tramas de tamaño fijo. Llamaremos período t de una trama al tiempo necesario para transmitir esa trama. Asumiremos que se generan nuevas tramas según una distribución de Poisson[3] de media S (tramas por período de trama). Vamos a considerar que $0 < S < 1$. También hay que tener en cuenta las retransmisiones de tramas; en k intentos de transmisión (tramas enviadas y tramas reenviadas) también se sigue una distribución de Poisson de media G. Evidentemente se cumple siempre que $G \geq S$. Si la carga es baja (apenas se usa el medio) habrá pocas colisiones y $S \simeq 0$ a la vez que $G \simeq S$. En el caso de que el tráfico sea alto, $S \uparrow \Longrightarrow G \rightarrow S$. El nº de tramas enviadas con éxito será $\mathbf{S} = G \cdot P_0$ siendo P_0 la probabilidad de que una trama consiga transmitir con éxito. Según la distribución de Poisson, la probabilidad de que se generen k tramas por cada tiempo de trama será

$$P_r[k] = \frac{G^k \cdot e^{-G}}{k!}$$

Por tanto, la posibilidad de que no se genere ninguna trama en un tiempo t será $P_r[0] = e^{-G}$. Si G es muy grande (mucho tráfico) esta probabilidad tiende a cero, evidentemente. Llamamos **tiempo vulnerable** al tiempo de transmisión sujeto a colisiones. Como muestra la figura 2.2 , no es t sino $2t$.

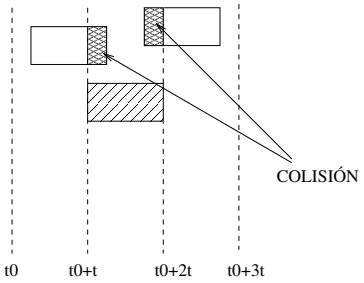


Figura 2.2: Esquema de tramas utilizando t tiempo de transmisión. Es necesario un margen mayor de tiempo para evitar la colisión.

Si asumimos que los sucesos son independientes, la probabilidad de no colisionar en ese periodo vulnerable $2t$ será el producto de probabilidades de no generación de más tramas en cada uno de los períodos t individuales.

$$P_0 = P_r[0] \cdot P_r[0] = e^{-2t}$$

[3] dados sucesos independientes que ocurren en un determinado tiempo, si el tiempo es lo suficientemente grande, la forma de la distribución tiende a una forma concreta que llamamos en términos estadísticos como *distribución de Poisson*.

luego el nº de tramas que llega satisfactoriamente seguirá la siguiente distribución de probabilidad

$$S = G - e^{-2G}$$

Veáse la figura 2.3 en donde puede apreciarse que el porcentaje de éxito es el 18 %.

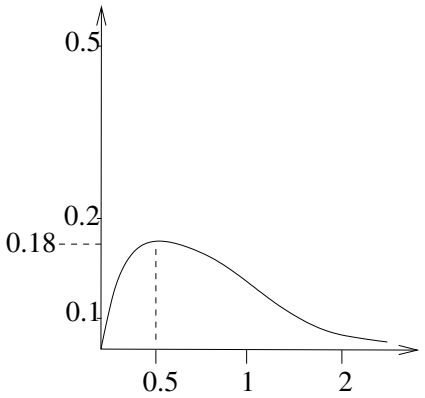


Figura 2.3: Efectividad del control ALOHA puro

2) ALOHA ranurado:
   - Existen unos periodos de transmisión definidos por una de las entidades. Es, de alguna forma, una compartición.
   - Los períodos de transmisión son intervalos discretos de tamaño

$$t = \frac{\text{trama}}{\text{tasa de bits de la red}}$$

   - El período vulnerable pasa a ser t ya que hemos fijado el tiempo de transmisión.
   - La retransmisión se produce tras un número **entero** aleatorio de cuantos de tiempo de trama. Véase la figura 2.4 para un esquema de su evolución en el tiempo.

b) ETHERNET: Toman el nombre genérico de *mecanismos de acceso múltiple con detección de portadora* (CSMA)

Extiende el concepto de ALOHA mediante la detección del uso del canal **antes** de transmitir (detección de portadora). El ETHERNET actual utiliza el control de acceso al medio denominado **CSMA/CD 1 persistente**.

1) CSMA 1 persistente[4]: Cuando una estación tiene intención de transmitir, mirará si ya está ocupado el canal. Se llama *1 persistente* porque la estación transmite con **probabilidad 1** cuando encuentra el canal vacío[5].

[4] Las empresas que definieron este estándar fueron Xerox e Intel.
[5] Transmitir con probabilidad 1 no quiere decir que no exista una colisión más tarde.

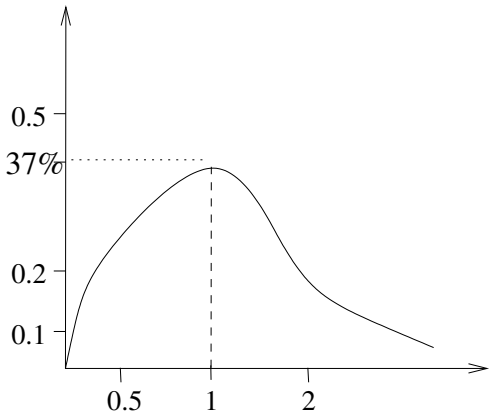


Figura 2.4: Efectividad del control ALOHA ranurado

Si hay colisión se comporta como ALOHA (espera de tiempo aleatorio y vuelta a empezar). El problema que presenta esto es fundamentalmente el siguiente: Si varias estaciones están esperando a que quede libre el medio, cuando una estación deje de enviar tramas, el resto provocará una colisión inicial prácticamente inevitable porque al ser 1 persistente todas se lanzan inmediatamente a transmitir.

2) CSMA no persistente: La entidad que desee transmitir escucha y si el canal está ocupado *espera un tiempo aleatorio* antes de volver a escuchar.

3) CSMA p-persistente: Si el canal está ocupado la entidad se mantiene a la escucha, y si está libre transmite con una probabilidad p. Tendrá una probabilidad $q = 1 - p$ de no transmitir. Se utiliza con ranuración de tiempo. Si en un intento de transmisión el canal está ocupado, esperará un tiempo aleatorio antes de volver a transmitir con p probabilidad. De esta forma, mezcla *persistencia con probabilidad* con *no persistencia* para volver a escuchar. Tras un número de veces adecuado (G) aumenta la probabilidad de lograr transmitir. Véase la figura2.5 en donde se hace una comparativa de los diferentes métodos.

4) CSMA/CD 1 persistente: Es el método que utiliza actualmente ETHERNET. La parte CD (*Collision detection*) es una detección *temprana* de la colisión (sólo envía la trama entera si el comienzo de ella no ha colisionado ya que al enviar los pulsos eléctricos correspondientes a los primeros bits uno puede saber si ha colisionado mediante un control analógico). Se puede esquematizar como: *Mientras transmito*, *escucho*. Las ventajas son aumento del ancho de banda disponible y ahorro de tiempo de transmisión. Calcularemos el tiempo que una entidad necesita para estar segura de que no ha habido una colisión. Si τ es el tiempo que tarda en llegar una trama de una entidad a la otra, y ha transcurrido $t_o + \tau - \epsilon$ cuando la estación destino decide transmitir creyendo que el canal está libre, la trama de vuelta (o la señal de colisión) deberá volver, tardando un tiempo

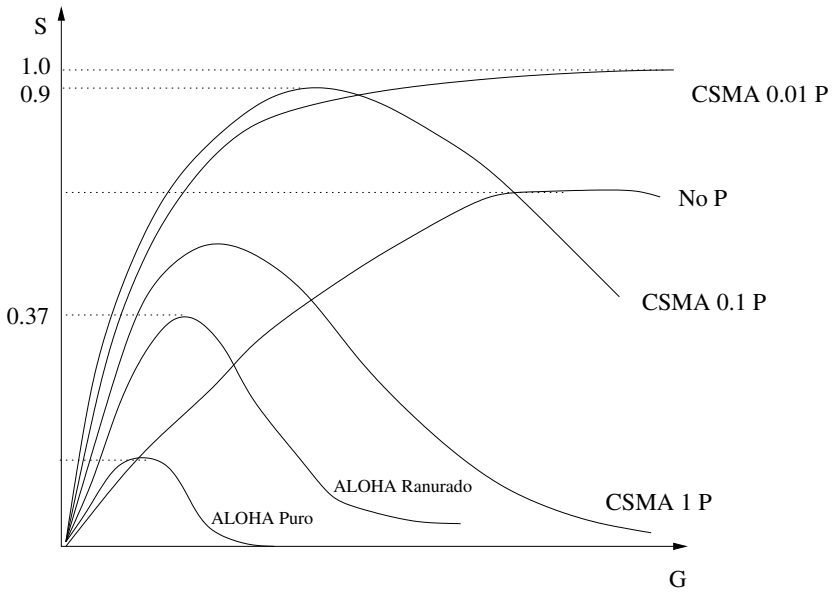


Figura 2.5: Comparativa de los diferentes métodos

total máximo de 2τ.

2. Mecanismos de reserva: Son mecanismos libres de colisiones. Antes de una transmisión habrá una ventana temporal "periodo de reserva" donde todas aquellas computadoras que deseen transmitir lo comunicarán. Veamos un ejemplo característico:

   Mapa de bits Tenemos una LAN de $N = 8$ máquinas. La ventana de reserva se divide en 8 partes. Esta partición del tiempo sí es estática. Las máquinas van colocando un 1 en su ranura para indicar si quieren transmitir durante lo que se conoce como *periodo de reserva*. Cumplido ese plazo, la primero que transmite es la primera que dijo "1" según la secuencia de bits de la ventana de reserva. El tamaño de trama para el envío no es fijo aunque existen unos máximos. Véase figura 2.6. Una vez que todas las que querían transmitir lo hicieron, la ventana de reserva vuelve a estar disponible. Existe un problema con este procedimiento y es que es muy probable que una estación desee transmitir cuando se le haya pasado el periodo de reserva. Estadísticamente, las estaciones etiquetadas con números bajos tardan 1.5 veces más que los números altos. Otro protocolo de este tipo es el Conteo Binario Descendente (ver [Tanembaum]).

3. Mecanismos de selección: Se basan en que para transmitir han de estar en posesión de una determinada trama llamada *testigo*. Veremos únicamente el protocolo Token Ring. Se refiere al estándar IEEE 802.5. El inventor, IBM, buscaba un protocolo de acceso al medio en donde las esperas máximas de acceso al medio estuviesen

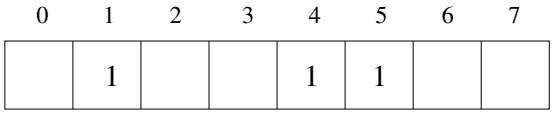


Figura 2.6: Ejemplo de Mecanismo de reserva; mapa de bits. Las máquinas 1,4 y 5 han mostrado su deseo de transmitir.

acotadas y, sobre todo, por su capacidad para crear un anillo utilizando conexión Punto a Punto en cada par de máquinas consecutivas. Tendremos una *trama testigo* que viajará por toda la red en una sola dirección (siempre se transmite en un sentido, véase la figura 2.7) y quedará atrapada por aquella máquina que desee transmitir. La trama testigo habilita a la máquina a transmitir durante un tiempo máximo *de vencimiento* de trama. Con este tipo de topología ganamos dos cosas.

- Tiempo máximo de espera acotado.
- Asentimientos muy sencillos.

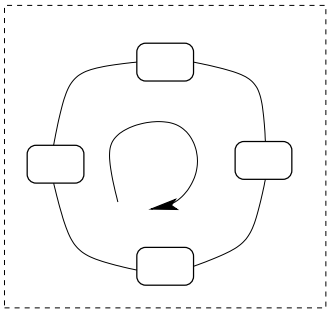


Figura 2.7: En TokenRing, la transmisión se realiza siempre en un mismo sentido

Las estaciones actúan de repetidores ya que aunque lean una trama que no desean la reenvían. Para evitar que haya tramas viajando indefinidamente por el canal, la estación origen se encarga de comprobar que todo ha ido perfectamente cuando una trama que envió vuelve a ella. La trama testigo se vuelve a generar en la estación que emitió por última vez cuando la transmisión ha sido completada o cuando el tiempo de vencimiento queda agotado. También existen problemas:

- si se cae una estación, se rompe el anillo.
- una nueva máquina a insertar necesita conocer a sus vecinos (configuración).
- ¿si el testigo se pierde quién la regenera? Por ejemplo, la máquina que lo tenía se cae. Token Ring,a pesar de su estructura de anillo, es una topología de selección **centralizada** así que una nueva máquina, del anillo, se encarga de ello.

Las máquinas que están *en pie* introducen siempre un retardo en el cable de forma que para saber si una máquina está caída o no se detectará si no existe ese retardo. De ser así, las máquinas vecinas sabrán inmediatamente que la máquina está vacía y se redefinirán como vecinos directos. Las máquinas pueden estar transmitiendo, escuchando o caídas. Véase figura 2.8.

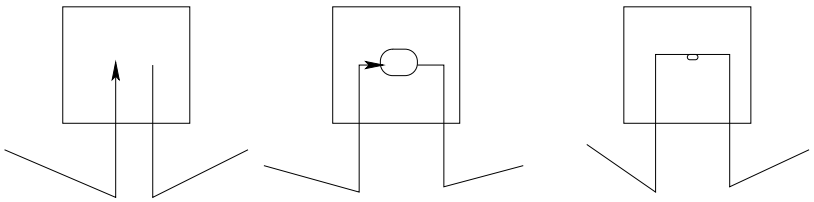


Figura 2.8: Transmitiendo. Escuchando (retardo). Apagado (sin retardo).

¿Cómo es la trama testigo? Es una trama de 3 bytes. Cuando una máquina desea transmitir, modifica la trama en un bit y esta trama queda etiquetada, no ya como *testigo* sino como "detras de mí viene una trama de datos" (la llamaremos, trama *post-testigo*).

Decíamos que era necesaria una máquina líder o supervisora que se encargase de regenerar el anillo si éste se rompe. Cuando una máquina supervisora se cae, pasa un tiempo y cualquier otra máquina pasa a ser supervisora.

Ya vimos qué mecanismo de puenteo se emplea cuando una máquina se cae ¿pero qué sucede si el cable se corta? Hay diferentes topologías que hacen frente a este problema:

- Está la topología anillo-estrella que vemos en la figura 2.9. El concentrador, si se corta un cable, conmuta.

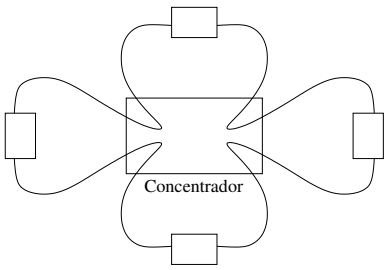


Figura 2.9: Topología anillo-estrella en TokenRing.

- Doble anillo. Conmuta y se genera otro anillo. Véase la figura 2.10.

Token Ring nos permitirá asignar prioridades al tráfico de 1 a 7, siendo 7 lo más prioritario. De forma que para transmitir es necesario capturar un testigo de prioridad igual o inferior a la prioridad de los datos que se desea transmitir. Esos testigos de prioridades se generan de la siguiente manera.

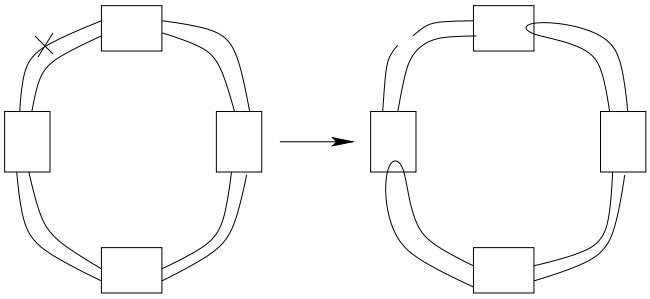


Figura 2.10: Topología de doble anillo en TokenRing. Comportamiento ante una ruptura física.

Una máquina está transmitiendo datos, la trama *que fue* testigo (post-testigo) que viaja delante de los datos va siendo modificada por las máquinas por las que pasa con una cierta prioridad (ej, 2, 4, 5). Sólo una prioridad alta sobreescribe ese campo de prioridad en la trama (una máquina que desee transmitir con prioridad 2 no tocará una trama post-testigo que tenga prioridad 3). Cuando la transmisión finaliza, la máquina origen genera un testigo con prioridad igual a la prioridad máxima que llegó con la anterior trama post-testigo. Sólo las máquinas que tengan datos con prioridad igual o mayor que la prioridad del testigo podrán capturarlo y transmitir. Aquéllas que no tengan la prioridad requerida pueden, no obstante, marcar la prioridad en el testigo. En realidad, son los servicios son lo que proporcionan la prioridad, no la máquina en sí. La responsabilidad de bajar la prioridad del testigo es de la máquina que subió la prioridad de dicho testigo (si no, las máquinas con servicios de poca prioridad nunca podrían transmitir)[6]. Cuando una estación recoge el testigo de la red, modifica el bit de Token **T** (pasa de 0 a 1) y la trama testigo se convierte en trama post-testigo. Véase la figura 2.11. **SD**: comienzo de trama. **ED**: fin de trama. **FS** (Frame state, 2 bits): no va controlado por la trama y por ello va tras **ED**, archiva los asentimientos (bits $A = 0, C = 0$ ⇒el destino no está presente o encendido. bits $A = 1, C = 0$ ⇒el destino está presente pero por alguna razón no ha aceptado la trama. bits $A = 1, C = 1$ ⇒ el destino ha aceptado la trama. A es 'recibido' y C es 'correcto') Como este campo no está controlado por el *checksum*, estos bits están replicados para asegurar la corrección. **FC**: distingue entre tramas de datos, tramas de control (de la propia capa). Checksum: mecanismo de detección de errores (más simple que el **CRC**).

[6] Intentarán bajar la prioridad al nivel en la que se la encontró modificando un bit cada vez.

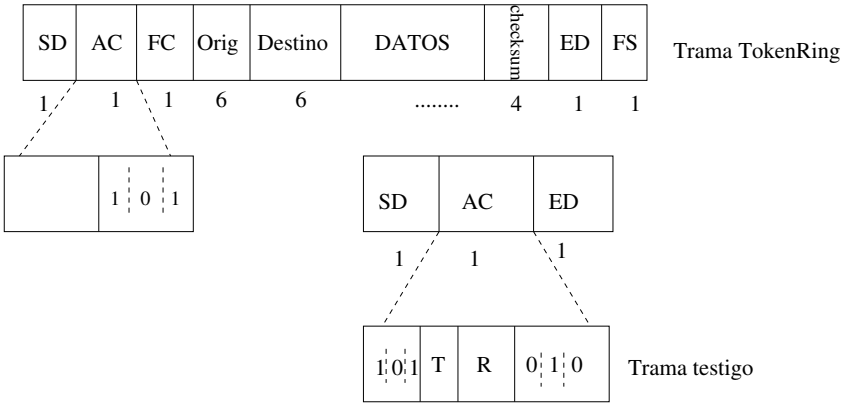


Figura 2.11: Tramas Token Ring y Testigo. Detalle de las prioridades.

2.2. Formato de trama definido por el estándar 802.3

Una PDU del protocolo definido por el estándar 802.3 tiene la forma descrita en la figura 2.12

CRC es el código de redundancia cíclica, que sirve para comprobar que la trama ha llegado correctamente.

Dirección destino	Dirección origen	Longitud	Datos	PAD	CRC
6 bytes	6 bytes	2 bytes	0–1500 bytes	0–46 bytes	4 bytes

Figura 2.12: Estructura interna de una PDU del rpotocolo ETHERNET

Debido a que existen impactos/colisiones durante el tráfico, quedarán fragmentos de tramas inservibles circulando. Para discernir entre tramas fragmentadas y no fragmentadas se fija un tamaño mínimo de trama (64 bytes, que es el total de sumar 6+6+2+46+4). el campo **PAD** sirve para meter *relleno* hasta completar un tamaño mínimo de trama. Si el campo de Datos es superior a 46 bytes, el PAD es cero. Otra razón para establecer un tamaño mínimo de trama es para evitar que una estación finalice una transmisión antes de saber si existirá colisión o no (recordemos el mecanismo CSMA/CD 1-persistente de Ethernet).

Además de los campos que ya hemos visto en la figura 2.12 existen dos añadidos delante y detrás de la trama de 7 bytes (10101010 siete veces) que son, en realidad, una señal cuadrada como veremos más adelante. Siguiendo a esos 7 bytes hay otro byte 10101010 *de comienzo de trama*. Su propósito es delimitar inicio y fin de trama y aunque van insertadas obligatoriamente para que el nivel de enlace sepa tratar tramas de forma correcta no los hemos incluido en nuestro esquema de trama puesto que no se consideran parte de la definición de trama.

2.2.1. Codificación de la señal para MAC

Ya vimos que era posible codificar los 1 y 0 mediante voltajes discretos. El problema de esta codificación binaria es que es muy sensible a ruido en el canal. Se utiliza, por tanto, otro sistema denominado codificación de Manchester. Consiste básicamente en establecer aumentos o disminución del voltaje dependiendo de si queremos transmitir un 0 ó un 1. Si queremos transmitir un 0, subiremos bruscamente el voltaje y si lo que deseamos transmitir es un 1 bajaremos bruscamente la tensión.

Un caso, quizá mejorado, de la codificación Manchester es Manchester diferencial: invierte el signo de la señal sólo si hay una transición a un determinado valor (1 ó 0, lo que decidamos). Véase la figura 2.13 para una comparativa visual de las tres codificaciones.

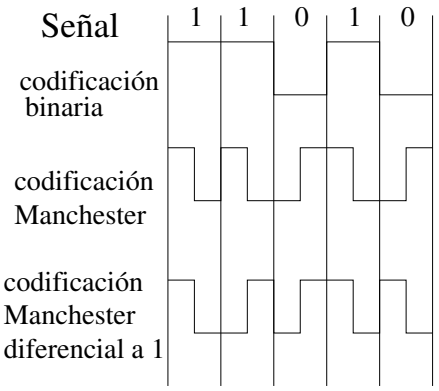


Figura 2.13: Ejemplos de codificación de señales

2.2.2. Retroceso exponencial binario

En un mecanismo que se utiliza en redes ETHERNET utilizando CSMA y ranuración de tiempos que consiste en lo siguiente: Nos dará el intervalo en el que se moverá el tiempo aleatorio de retardo que veíamos **en función del nº de colisiones**. La relación matemática es conforme a potencias de 2. Es decir, a medida que aumenta el nº de colisiones, el tiempo de retardo aleatorio tiende a números cada vez más grandes.

Tras i colisiones, el tiempo aleatorio de retardo tendrá un rango $\left[1, 2^i - 1\right]$de valores de ranura. Tras 10 colisiones se fija el número de ranuras en lugar de seguir creciendo, y será fijado a 1023 ranuras. Tras 16 colisiones se informa al protocolo superior del fracaso de la transmisión.

2.3. LLC (Logic Link Central)

Recordemos que LLC servía a la capa de red para trabajar independientemente del modo de acceso al medio (MAC) y también ofrecía el control del flujo. Además, podremos hacer corrección de errores (en el subnivel MAC sólo podíamos detectarlos). Como se

encuentra inmediatamente debajo de la capa de red, ofrece servicios a ésta que pueden ser CO y fiables, de datagramas con asentimiento (CL y fiable), de datagramas sin asentimiento (CL no fiable).

Los dos primeros necesitan números de secuencia y de asentimiento.

números de secuencia sirven para descartar tramas duplicadas innecesariamente (cuando se ha recibido una trama correctamente pero el asentimiento de vuelta no llega por alguna razón y provoca un reenvío en origen innecesario).

números de asentimiento Hemos de saber a qué trama corresponde un asentimiento concreto.

LLC se basa en una especificación OSI llamada HDLC *High-level data link central*.

La funcionalidad que se encuentra en LLC es el Control de errores de transmisión; En general, para un control de errores existen dos estrategias básicas:

- FEC: *Forward error corrector*. se corrigen los errores al llegar al final (el destino es el que corrige errores). Son muy complejos porque necesitan información redundante. Se utilizará en enlaces donde la retransmisión sea imposible o no tenga sentido.
- ARQ: Detección + reenvío. se detecta el error y se solicita un reenvío. Se utilizarán en la mayoría de los casos debido a su simplicidad y su eficiencia.
- Híbridas: combina las dos anteriores según se muestra en la figura 2.14.

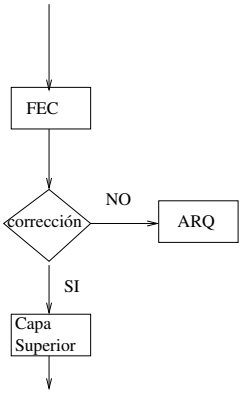


Figura 2.14: Esquema de una estrategia híbrida de control de errores.

En general hablaremos de tramas de m bits y r bits para detectar errores, a la suma de m y r lo denominaremos *palabra código*.

Distancia Hamming entre dos *palabras código* es el número de bits en que difieren. Se hace haciendo un XOR y contando el número de 1's.

ejemplo $1010 - 1001 = 0011$ Distancia Hamming $= 2$

Distancia Hamming de todo un código será la mínima distancia Hamming de dadas dos palabras cualesquiera del código.

Si dos palabras código tienen distancia hamming $DH = d$ son necesarios d errores sencillos (en un bit) para pasar de una palabra a otra. Un código que tenga distancia hamming $HD = d + 1$ nos va a permitir **detectar** d errores sencillos.

Para **corregir** d errores el código ha de tener una $DH = 2d + 1$

ejemplo Supongamos el siguiente código:

0000000000 0000011111 1111100000 1111111111

Vemos que la Distancia Hamming es $DH = 5$. Por tanto, seremos capaces de detectar 4 errores y corregir hasta 2. Si llega una trama **0100001000**, comprobamos rápidamente que no es una trama del sistema. Calculamos entonces la DH con respecto a todas las tramas posibles del código y la convertiremos en aquélla con menor DH relativa a nuestra trama errónea.

Intuitivamente se ve que para aumentar DH con muchas palabras diferentes, deberemos aumentar el tamaño de palabra. Ese aumento se va hacia los códigos de redunancia.

2.3.1. Métodos reales de detección de errores

Bit de paridad

1. Par: Dada una trama, cuenta el número de 1s y añade un 0 si es par y 1 si es impar.

2. Impar: Dada una trama, cuenta el número de 1s y añade un 0 si es impar y 1 si es par.

Es un código de detección muy sencillo y, por tanto, poco potente ya que será imposible saber qué bit cambio y habrá que solicitar un reenvío.

En ocasiones se suele redundar/repetir los bits para complementar la información básica con otra que ayude a informar sobre el bit que cambió.

Este código puede tener sentido en capas inferiores que pueden descartar un porcentaje apreciable de casos. Los casos *que se cuelan* se dejan para mecanismos de corrección de más alto nivel, como FEC.

Checksum

Como su propio nombre indica, es una suma considerada como *check*. Coge la trama de bits, suma *en base dos* todos los bits y el número resultante lo coloca en el campo de *checksum*. Como el bit de paridad, es capaz de detectar que hubo errores pero no cuáles fueron.

Código de redundancia cíclica CRC

Habitualmente, los campos en las tramas que albergan la información del CRC se llaman FCS (Frame check sequence)[7]. Trata las tramas de bits como polinomios binarios.

Ejemplo La trama 101101se transforma en $x^5 + x^3 + x^2 + 1 = M(x)$

El emisor y el receptor se ponen de acuerdo en el tipo de polinomio generador $G(x)$que van a utilizar. Lo que se va a enviar será $T(x) = M(x) \cdot x^n + F(x)$. El destino dividirá $T(X)$entre $G(X)$ y sólo un resultado del **resto** igual a 0 asegurará que no ha habido errores.

Ejemplo Tenemos una trama 1101 y con un código generador $G(x) = 11$. Los polinomios asociados son, respectivamente, $x^3 + x^2 + 1$ y $x + 1$. n, el exponente de x será el grado del polinomio generador, 1 en nuestro caso.

$$M(x) \cdot x^n = \left(x^3 + x^2 + 1\right) x = x^4 + x^3 + x$$

$F(x)$ se definirá como

$$F(x) = Resto\left(\frac{M(x)\, x^n}{G(x)}\right)$$

que en nuestro caso, será $\frac{11010}{11} = 10$. De esta forma ya tenemos $T(x) = 11010 + 10 = 11000$ (esta suma se hace sin acarrero, o lo que es lo mismo, estamos realizando un XOR). La estación destino procederá a dividir $T(x)$ entre $G(x)$ y si el resto da 0, todo habrá ido bien.

Los tipos de polinomios generadores más utilizados hoy en día son[8]:

- CRC-12: $x^{12} + x^{11} + x^3 + x^2 + 1$
- CRC-32 (definido el estándar IEEE 802.2): $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

2.3.2. Métodos de corrección de errores

Veremos solamente un caso sencillo de corrección que podríamos englobar en el mismo nivel de simplicidad que el bit de paridad en la detección de errores.

Este método sólo corregirá errores sencillos (recordemos que un error sencillo era aquél que sólo sea veía afectado en un bit). Queremos transmitir m bits pero insertamos en ella r bits de la forma siguiente: r_i se meterá en cada posición potencia de 2 fragmentando m. Ejemplo con 1011.

1. Se enumeran todos los bits de la palabra código empezando en 1 y por la izquierda.

[7] Importante a la hora de consultar la bibliografía.
[8] en términos de menor información redundante y eficiencia en la detección de errores.

x	x	1	x	0	1	1
1	2	3	4	5	6	7

2. Los bits que sean potencias de dos serán considerados bits de chequeo, los demás serán bits de datos.

x	x	1	x	0	1	1
1	2	3	4	5	6	7
c	c	d	c	d	d	d

3. Cada bit de chequeo fuerza la paridad con una colección determinada de bits (que son los bits de datos y los descompongo en *monomios* potencias de dos, pero sólo aquéllos que tengan al bit de chequeo como parte de esa descomposición).

4. El bit incorrecto se halla sumando la posición de los bits de secuencia erróneos.

2.3.3. Mecanismos de control de flujo

Tenemos que preguntarnos cuándo un emisor puede transmitir una trama y cuándo no, ¡quizá el receptor no sea capaz en ese momento de procesar los datos!. Debemos definir también estrategias de retransmisión (fiabilidad en el servicio). Además, hemos de decidir el tamaño que debe tener la trama porque éste afectará a las transmisiones. Por una parte las tramas pequeñas aseguran que la retransmisión será pequeña pero, por otro lado, habrá que meter más bits de control (overhead). A su vez, una trama grande evita el malgasto del canal en información de control pero es más costosa de transmitir[9]. Vemos claramente que habrá que establecer un compromiso.

De todas formas, vamos a olvidarnos de ese detalle por ahora y nos centraremos exclusivamente en las dos estrategias de retransmisión más importantes, que son:

1. **Parada y espera**: El emisor emite una trama y espera a la respuesta del receptor para enviar otra trama. Cuando el emisor envía una trama, éste mantiene una copia de esa trama por si fuera necesario repetir el envío. Los números son números de secuencia y comienzan en el 0. El código de asentimiento funciona con el siguiente convenio: *El receptor dice que ha recibido bien una trama solicitando la siguiente*. Ej: trama $0 \rightarrow ACK1$. En un primer análisis podríamos pensar que este método funciona bien porque asegura una correcta transmisión, sin embargo generar asentimientos por cada trama ocupa, innecesariamente a veces, ancho del canal.

2. **Envío continuo**: Es una evolución del anterior método. Enviamos las tramas seguidas y no esperamos a que nos informen de si una de ellas ha llegado incorrectamente. El problema es que hasta que no nos confirmen las recepciones de forma correcta, no podemos desechar las copias de tramas *enviadas pero no confirmadas*

[9] entre otros motivos, por la mayor facilidad para colisionar.

y eso requiere un gasto de memoria grande[10]. Las *ventanas* son sólo de capacidad unidad, tanto en origen como en destino. Sólo se es capaz de guardar una trama recibida a la vez. Si la capa de enlace no puede pasar la trama a la siguiente capa, la descartará inmediatamente.

Ventana es el número de tramas que pueden estar pendientes de confirmar por el receptor.

ejemplo se envía la 0 y llega a destino, se envía la 1 pero ésta no llega (el emisor no ha recibido aún confirmación ni de 0 -está en camino- ni de 1 -no la ha recibido-). el receptor recibe la trama 2 sin haber recibido **antes** la trama 1. Para decir "rechazo" a la trama 2 hemos de responder con ACK1 diciendo 'espera, no recibí la trama 1 correctamente, reenviala'. Como una trama de asentimiento puede perderse (es la vida), la manera que tiene el emisor de confirmar que se envío una trama es recibir un ACK mayor de lo requerido pues eso ya implica una correcta transmisión hasta la fecha. Es decir, *un ACK de número n confirma todos los n-1 anteriores.*

Dos formas de rechazar la transmisión:

- Rechazo simple: cuando hay un error se notifica y se retransmite *desde el último asentimiento positivo* ya que el receptor ha ido rechazando todas las tramas que se mandaron después de la última trama correcta (recordamos, tamaño de ventana receptora 1).
- Rechazo selectivo: complicamos el receptor de forma que sólo tenga necesidad de pedir las tramas incorrectas, ninguna más. Para ello incrementamos la memoria disponible para almacenar más tramas, aumentamos las ventana de gestión (incorporamos un *buffer*).

La técnica más interesante para resolver el problema del flujo (adaptar velocidades) es lo que se conoce como *ventanas deslizantes*. Combinado con la modalidad de Rechazo selectivo en Envío continuo (e incluso rechazo simple) tendremos el mecanismo más eficiente, como veremos a continuación.

En cualquier instante, usando los protocolos que utilizan ventanas deslizantes, el emisor va a mantener un número n finito de números de secuencia que va a corresponder al número de tramas que le está permitido transmitir de forma continua. Se dice que esas tramas que puede transmitir *caen* dentro de la **ventana de transmisión**.

Por otro lado, el receptor va a trabajar con una ventana que maneja unos números de secuencia que nos indica las tramas que le está permitido recibir. Es decir, tendremos dos ventanas, una de transmisión y otra de recepción **no necesariamente iguales** en tamaño.

Si se agota la ventana del emisor habrá que esperar asentimientos positivos que le permitan liberar espacio en la ventana e incluir tramas en espera.

[10] también aparecen problemas con el uso de los número de secuencia, que son finitos, pero eso lo veremos más adelante.

Cuando el número de secuencia llega a un valor máximo se reinicia, recordemos que usamos un número de bits finito que permite una cantidad limitada de números de secuencia. Veremos más adelante una relación matemática entre el número de secuencia y el tamaño de ventana.

Antes de seguir con este análisis de las ventanas es conveniente introducir un concepto.

Un **ACK** es una trama que al fin y al cabo ocupa el canal y no lleva datos. Como estamos siempre buscando la máxima eficiencia en el uso de los canales de transmisión, existen método para que el ACK vaya incluido en otra trama de datos con destino el Emisor. Este método se conoce como *PiggyBacking* y evidentemente las tramas portadoras van identificadas como 'soy una trama de PiggyBacking'.

Llamaremos V_{ei} al borde inferior de la ventana emisora. V_{es} al borde superior de la ventana emisora. V_{ri} borde inferior de la ventana receptora y V_{rs} al borde superior de la ventana receptora.

La ventana máxima de emisión $V_E \geq V_{es} - V_{ei}$ y por lo tanto varía en función de las tramas que se van enviando.

El tamaño máximo de la ventana receptora es $V_R = V_{rs} - V_{ri} = \text{cte}$.

Sobre la Ventana de emisión diremos que el borde inferior, cuando le llega un ACK, pasará siempre a +1 salvo que se acaben los números de secuencia. El borde superior se le suma 1 cuando se envía una trama.

Veremos ahora la relación existente entre el número de secuencia y el tamaño de Ventana. Supongamos Ventanas deslizantes con rechazo simple con los siguientes parámetros. $V_E = 3$, $V_R = 1$. Vamos a meter algo que provocará un error eligiendo un número máximo de secuencia erróneo, $MaxSeq = 2$. Efectivamente, podríamos volver a emitir una trama con número de secuencia todavía vigente y provocaríamos confusión. Véase la figura 2.15 para un análisis detallado de una de estos escenarios.

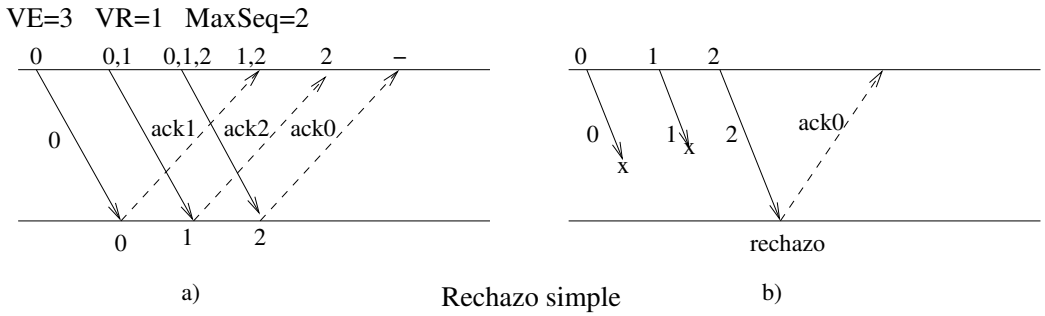


Figura 2.15: Rechazo simple. a) las tramas llegan todas bien. b) Sólo llega bien la trama 2 y se devuelve un ACK0 indicando que se produjo un error en la trama 0.

Se ve claramente que $V_E \leq MaxSeq$ ya que de otro modo se puede perder información sobre qué tramas llegaron incorrectamente al tener un espacio de respuestas limitado.

Un ejemplo parecido al anterior pero utilizando Rechazo selectivo:

Supondremos que el valor máximo de secuencia es $MaxSeq = 2$ y que $V_E = 2$ y $V_R = 2$. Véase la figura 2.16.

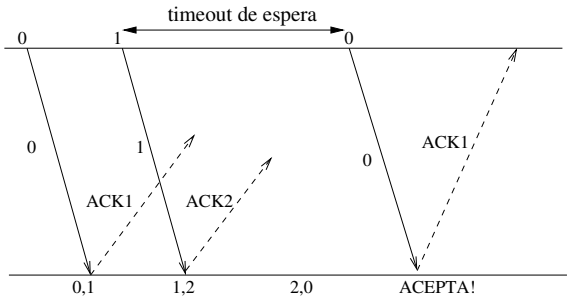


Figura 2.16: Rechazo selectivo. La máquina destino, al tener una $V_E > MaxSeq$ llega a aceptar una trama inicial tomándola como una nueva.

Vemos que a la capa de red le llegan datos repetidos cuando no era eso lo que se pretendía por lo que la comunicación falla[11]. Para evitar estos problemas decimos que el tamaño de ventana se calcula como

$$V_E = V_R \leq \frac{MaxSeq - 1}{2}$$

2.3.4. HDLC

HDLC *High-level data link control* fue ideado en los años 70 a partir de varias versiones pensadas por fabricantes que acabaron siendo estandarizadas por la ISO e IBM. En este protocolo se basa LLC. Existen varios modos que realmente se utilizan, entre ellos destaca ABM *asynchronous balanced mode* que todavía se utiliza hoy en día. Los otros modos estaba ideados para entornos centralizados que cada vez se utilizan menos.

La trama de HDLC se observa en la figura 2.17.

Tramas de información

$N\,(R)$ es el número de secuencia. Luego el número máximo de secuencia es 7. $N\,(S)$ es la siguiente trama que estamos solicitando (usamos la técnica **Piggybacking**).

S/F es el sondeo final que *se utilizaba* para dar y ceder permisos para poder transmitir en entornos no simétricos (centralizados).

Tramas de supervisión

El campo S indica el tipo de trama de supervisión. En la tabla siguiente se muestran los diferentes valores y sus significados.

Valor	Significado

[11] El error se manifestará en capas superiores.

00	Preparado para recibir
01	Rechazo a partir de la trama con $N(S)$
10	No estoy preparado para recibir, pero he recibido bien hasta lo que me indica en el $N(S)$
11	Rechazo, y deseo que me envíes la trama que te indico en $N(S)$

Si no podemos utilizar la técnica piggybacking, podemos enviar una trama de supervisión con la información que pedimos. En la bibliografía encontraremos que

00	RR
01	REJ
10	RND
11	SREJ

Tramas no numeradas

M son órdenes de elección de modo de operación. $2^5 = 64$ órdenes distintas (pero no se utilizan todas). ej: desconexión, reset, asentimiento no numerado para datagramas.

2.4. Nivel de enlace de la arquitectura TCP/IP

Básicamente podemos *montar* cualquier cosa sobre TCP/IP. Los servicios que monta la capa enlace debe permitirme llevar de un sitio a otro datagramas, y también datagramas ARP, RARP e IP[12].

La ortodoxia nos dice que montaremos un nivel de enlace siguiendo los estándares internacionales.

> El campo Tipo: Si vale 0800, el campo datos guarda un datagrama IP. Si vale 0806, el campo datos guarda un datagrama ARP (peticiones/respuesta ARP) (sólo 28 bytes y metemos un relleno PAD de 10 para llegar al tamaño mínimo de trama, 64). Si vale 8035, el campo datos guarda un datagrama RARP (sólo 28 bytes y metemos un relleno PAD de 10 para llegar al tamaño mínimo de trama, 64).

Otro protocolo que no es estándar, el SLIP, también puede montarse sobre el nivel de enlace en TCP/IP. Suele usarse para conectar dos equipos a través de una línea serie. La encapsulación es mínima. Llega un datagrama IP, le añade un par de deilimitadores de un byte y de valor C0. Para evitar confusiones, se sustiuye el código de C0 dentro del datagrama IP por **dbdc**. Pero como también puede aparecer **dbdc** en los datos, *escapamos* con **db ->dbdd**.

[12] que veremos en capítulos posteriores.

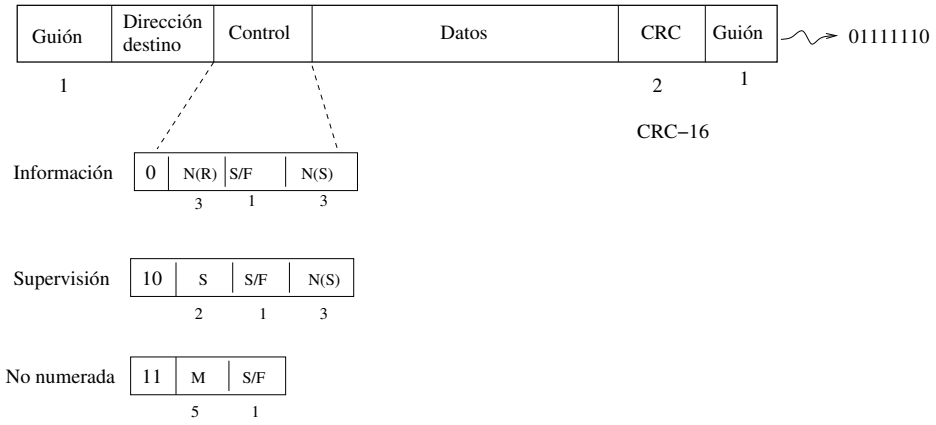


Figura 2.17: Esquema de una trama HDLC

Otro protocolo montado sobre TCP/IP más complejo que SLIP es el Point-to-Point protocolo PPP. Sirve para encapsular datagramas IP en una línea serie. Véase figura 2.18.

Figura 2.18: Estructura de un paquete PPP.

3 Nivel de Red

En el capítulo anterior, *Nivel de enlace,* hemos resuelto el envío de información entre dos equipos juntos/adyacentes. Conseguimos sincronización, control de flujo, detección de errores, etc.

Pero ahora nos planteamos el siguiente problema; **conectar una máquina con otra máquina esté donde esté**. El nivel de red le va a ofrecer a su capa usuaria (la de transporte) la posibilidad de conectarse con su entidad par situada en cualquier lugar del mundo. Es la primera capa en ofrecer un servicio extremo a extremo.

Las funciones que ha de cumplir son:

- Enrutar paquetes (así llamaremos a la PDU del nivel de red): Ésta es claramente su función básica.
- Mecanismos de control de congestión: para discernir qué nodos están congestionados, o no, para enrutar por un lado u otro.
- Contabilidad: contar la información que pasa por determinado sitio para poder tarifarla.

Para abordar el diseño de la capa de red debemos pensar en dos aspectos principalmente.

1. Qué tipo de servicios vamos a proporcionar al nivel superior (capa de transporte).
2. La organización interna de la propia capa.

3.1. Tipos de servicios proporcionados

En la actualidad existen dos tendencias claramente diferenciadas en el mercado acerca de lo que se debe ofrecer a la capa de transporte. La primera facción propugna ofrecer servicios orientados a conexión, CO, a la capa de transporte (las compañías telefónicas se alinean aquí). La segunda facción defiende que la capa de red debe ofrecer servicios no orientados a conexión, CL, a la capa de transporte. Si ninguna se ha llevado el gato al agua es porque se descubren ventajas y desventajas en ambos modelos. Los servicios CO son buenos para aplicaciones multimedia o, en general, cualquier servicio en donde primero hay que negociar una calidad de servicio (cota inferior de transferencia, retardo máximo aceptado, etc). Este modelo, por tanto, está asumiendo que hay mucha *inteligencia*[1] en la red.

[1] *hace* muchas cosas, es compleja y estable.

El otro modelo, por contra, defiende que la red es inherentemente inestable, no es controlable por nadie en su conjunto, y por ello consideran que los servicios CL son los más adecuados. Dicho de otro modo, *ya se encargarán* las capas superiores de arreglar los desaguisados producidos más abajo.

3.2. Organización interna de la red

Trata el movimiento de la información de un sitio a otro. Básicamente, habrá dos tendencias.

1. Utilizar conmutación de circuitos virtuales (no hay ninguna reserva física del canal pero a efectos prácticos lo simula). Servicio CO.
2. La organización en forma de datagramas. Servicio CL.

3.2.1. Circuitos virtuales

En el contexto de la organización interna de la capa de red, llamaremos circuito virtual a una determinada conexión. Pensando en un router que conecta dos dominios de emisión, podemos imaginarnos que ese router, en vez de analizar cada paquete que le llega (es su tarea habitual) para tomar una decisión de por dónde va, puede ser forzado[2] a realizar una misma operación para una serie de paquetes identificados (tipo de conexión). De esta forma, obtenemos una especie de circuito virtual que funciona de igual manera que su análogo lógico pero *sin reservar una capacidad física fija*. Estos paquetes **siempre** irán por el **mismo sitio** y **llegarán en orden**. De esta forma, el tiempo de procesamiento disminuye mucho y favorece a los servicios que necesitan retardos mínimos. Dado que los paquetes siguen la misma ruta deben recordar por qué interfaz[3] deben enviar cada paquete de una determinada comunicación.

En un nodo, ese *recuerdo* es, en realidad, lo que se denomina Tabla de Circuitos Virtuales. E indica qué deben guardar cada una de las entradas de esa tabla. Para cada uno de los circuitos virtuales guarda cuál es la interfaz de entrada al nodo y cuál la de salida. Cada paquete que viaja por un circuito virtual debe reflejar el nº de circuito virtual al que pertenece (lo veremos más adelante) y cuando sale, el nodo por el que pasó modifica su identificador de circuito. Véase la figura 3.1.

Es importante destacar que cada proceso indica cuándo ha dejado de usar un determinado circuito virtual. Es la forma en que se liberan los recursos con este sistema. Esa tarea la realizará un paquete de **cierre final** que va indicando a los nodos por los que pasa que el circuito virtual ya no será usado más y que se puede borrar la entrada correspondiente a la tabla.

[2] Se suele implementar por medio de unas tablas de consulta.
[3] Una interfaz es un dispositivo físico de entrada y salida de datos.

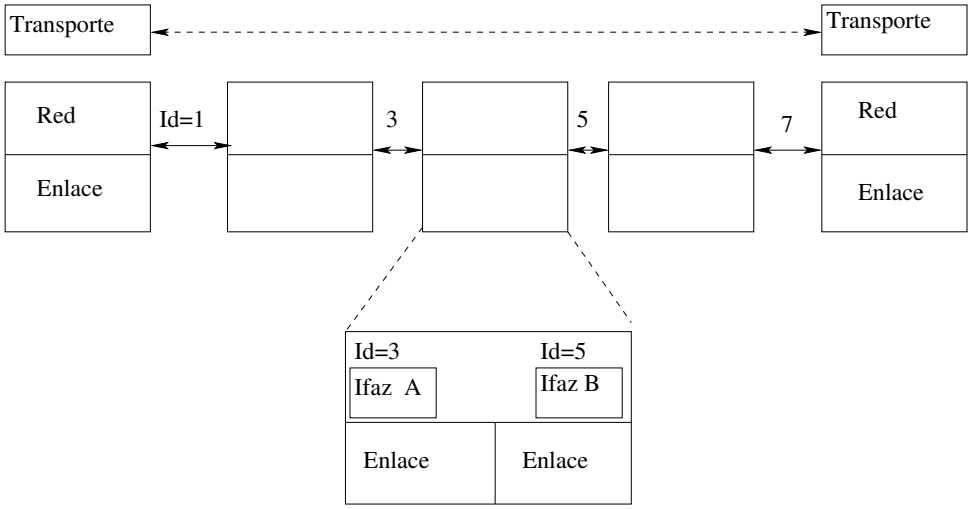


Figura 3.1: Esquema de circuitos virtuales.

3.2.2. Datagramas CL

Cada paquete entre A y B es tratado de forma independiente y cada nodo intermedio calcula la ruta para cada paquete individual. La tabla de circuitos virtuales, en este caso, no existe y se ve sustituida por la Tabla de Enrutamiento. Evidentemente, los nodos que sepan manejar ambos tipos de comunicación, albergarán los dos tipos de tablas.

Lo habitual, como vimos en el primer tema, era asociar datagramas a servicios CL y circuitos a servicios CO, pero esto no es así necesariamente. La combinación servicios CL + circuitos virtuales, por ejemplo, no tiene mucho sentido pero es factible.

Una comparativa entre Circuitos virtuales y Datagramas.

- Un circuito virtual implica un cierto tiempo de establecimiento de canal mientras que los datagramas no.
- El direccionamiento en los circuitos virtuales se establece mediantes los Ids, y en datagramas el direccionamiento es global (se basan en la dirección IP destino que se coloca en la cabecera). Los saltos que se dan en los nodos intermedios se realizan siempre en función de una dirección final.
- La información de estado. En los CV, se lleva mediante la tabla de CV mientras que en el caso de datagramas no se guarda información de estado.
- Enrutamiento: CV, una ruta fija para todos los paquetes. Datagramas, no existe una ruta fija.
- Efecto de un Nodo caído: en CV se pierden todos los CV que pasen por ahí. En el caso de los datagramas, se perderán sólo los datagramas que se encuentren en ese momento en el buffer del nodo afectado (equivalente a otra pérdida de transmisión).

- Control de congestión: en CV es relativamente sencillo porque al establecer el canal se está reservando una reserva de capacidades (y se puede establecer un máximo de éstos). En Datagramas el asunto es más complejo porque la información de la situación de la comunicación está más dispersa.

3.3. Protocolo IP

Este protocolo es tan fundamental que en ocasiones, se le llama nivel IP al nivel de red. La especificación formal del protocolo IP viene definida en **RFC 791**.[3]

Proporciona a la capa de transporte un servicio de entrega de datagramas que es **No Fiable** y **CL**. Cuando queramos utilizar IP para otro tipo de servicios tendremos que *decorar* el protocolo con extras. IP no mantiene información de estado de los datagramas.

IP envía las tramas de izquierda a derecha empezando por el bit de mayor orden (big endian). Las arquitecturas SUN trabajan nativamente de esta forma y apenas tienen que retocar nada para enviar. Todo lo contrario que los Intel, que usan Little endian. Véase la figura 3.2.

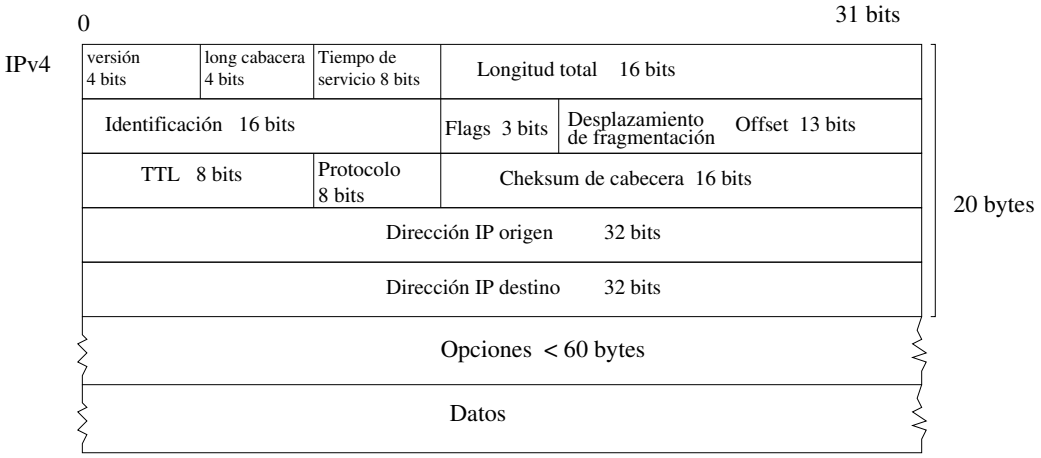


Figura 3.2: Esquema de un datagrama IP

Pasamos a comentar los campos del Datagrama IP.

- Versión: IPv4
- Longitud de cabecera: es el nº de bytes de cabecera medidos en bloques de cuatro bytes (incluye las opciones si las hubiere).
- Tiempo de servicio: 8 bits. los tres primeros son los bits de precedencia y no se usan para nada. los siguientes 4 bits son de información, y 1 bit que no se utiliza tampoco. Sobre los 4 bits de información diremos que sólo 1 *puede* estar a valor distinto de cero. Dependiendo de la posición del bit 1...

- Primer bit indica retardo mínimo.
- Segundo bit indica rendimiento máximo.
- Tercer bit indica fiabilidad.
- Cuarto bit indica coste

Según el servicio que se quiera usar, activaremos uno de estos bits. En una comunicación normal, están todos a cero. Sin embargo, en un servicio tipo **telnet** queremos que el retardo sea mínimo, luego marcaremos el primer bit con un 1. El rendimiento máximo significa que con poca información de control se consiga máxima transferencia. Para la gestión de una red (SNMP) necesito que el servicio sea fiable. El servicio de News, es un ejemplo de abaratamiento de coste máximo.

- Longitud total: Es un campo de 16 bits que nos da el tamaño total de la longitud del datagrama, siendo el tamaño máximo teórico $2^{16} = 65535$ bytes. Este dato se utiliza en el nivel de enlace para calcular el tamaño del relleno necesario. En cualquier caso, Las aplicaciones suelen restringir el tamaño máximo de datos a 8192 bytes.

- Protocolo, 8 bits: El tipo de protocolo que tengo encapsulado en el nivel de Datos. A veces se encapsula IP sobre IP (para crear un túnel, por ejemplo). Cuando encapsula ICMP (mensajes de control, que veremos en un capítulo posterior) toma valor 1. Cuando encapsula IGMP, toma el valor 2. Y, habitualmente, TCP (protocolo de transporte) toma el identificador 6, y UDP (también protocolo de transporte) toma 17.

  Ahora nos enfrentamos a un problema. El nivel físico puede imponer ciertas restricciones sobre el tamaño de trama al nivel de enlace. Para manejar estas situaciones existen tres campos especiales (segunda fila en nuestro diagrama de la figura 3.2). Aparece la denominada *fragmentación* que consiste en trocear los datagramas identificados claramente como 'fragmentos'. El nivel de enlace destino se da cuenta de esto y espera a tener todos los fragmentos para concatenar y *subir arriba*.

- Identificación, 16 bits: identifica a un datagrama IP de forma unívoca. Cuando se produce una segmentación, la identificación de los hijos es la misma que la del padre. Normalmente, el identificador funciona como un contador entero. Con 16 bits tenemos suficientes números de secuencia disponibles para evitar solapamiento efectivo.

- Flags, 3 bits: El primero no se usa. Todos los fragmentos llevan el segundo bit a 1 (indicando que existen *More Fragments*) y el último de la lista de fragmentos llevará 0 (al igual que los no fragmentados). El tercer bit está a 1 si no permite fragmentarse (*Denied Fragment*). Ahora bien, ¿Qué sucede si un datagrama no permite fragmentación y una capa más baja es incapaz de manejarlo? Se produce un error tipificado ICMP.

- Desplazamiento de fragmentación, 13 bits: Indica la posición relativa del fragmento en la trama troceada. Establece un límite al número de fragmentos para una misma trama. Se suele fragmentar las tramas en múltiplos de 8 bytes.

Cuando realizamos una fragmentación, necesitamos *tocar* el campo Longitud para calcular las longitudes de los *hijos*.

- TTL (Time To Live), 8 bits: tiempo de vida máximo para el paquete. Previene bucles de enrutamiento. Un router que recibe un datagrama con TTL=0, lo descarta y devuelve un datagrama ICMP específico al origen. En un inicio, se refería al número de segundos permitido de vida pero el cálculo se vuelve complicado, así que se decidió que se redefiniría como número de saltos permitido al paquete. El valor 255 suele ser algo teórico, en realidad se suelen establecer maximos de 32 o 64 saltos.
- Checksum de cabecera, 16 bits: Se calcula la suma *complemento a 1* de bloques de 16 bits. Al resultado de esa suma se le vuelve a hacer el *complemento a 1*, y eso se coloca en el campo *checksum* de cabecera. La recepción realiza la suma *complemento a 1* de la cabecera respetando el *checksum* original, si da distinto de 0 es que algo ha salido mal. Como cada router en cada salto modifica el campo **TTL** es necesario recalcular en cada salto el *checksum*.
- Direcciones IP

  Cada una de las interfaces de red de Internet tiene asignada una dirección IP. Estas direcciones están agrupadas en cuatro grupos de 4 bytes. Existen cinco clases de direcciones IP que mostramos en la figura 3.3.

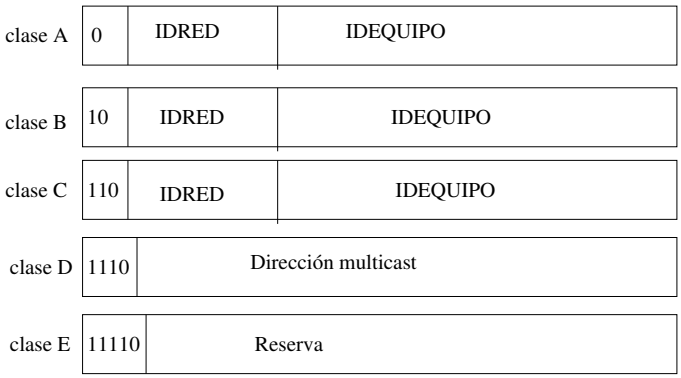


Figura 3.3: Rangos de IPs

- Clase A: Siempre empieza por el bit 0, tiene una parte de red que la identifica y otra parte que identifica al host. Su rango va de 0.0.0.0 a 127.255.255.255
- Clase B: Siempre empieza por el 10 . Su rango va de 128.0.0.0 a 191.255.255.255

- Clase C: Siempre empieza por 110. Su rango va de 192.0.0.0 a 233.255.255.255
- Clase D: Siempre empieza por 1110: Su rango va de 234.0.0.0 a 239.255.255.255
- Clase E: Siempre empieza por 11110. Su rango va de 240.0.0.0 a 247.255.255.255

Existen unos rangos de IPs privadas que los routers nunca redireccionan, sino que devuelven. Ejemplos: 192.168.x.x, 10.x.x.x, 172.26.x.x. Una dirección con todo a 0s es como decir YO. La dirección loopback es 127.0.0.1

Algoritmo 1 Salida del comando netstat -rn

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
213.149.240.76  0.0.0.0         255.255.255.252 U     0      0        0 eth0
213.149.242.220 0.0.0.0         255.255.255.252 U     0      0        0 eth0
213.149.240.64  0.0.0.0         255.255.255.192 U     0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         213.149.240.65  0.0.0.0         UG    0      0        0 eth0
```

- Opciones: El apartado de opciones, que suele ser ignorado por la mayoría de las aplicaciones, suele utilizarse con
  - Seguridad y gestión de restricciones (motivos militares)
  - Registro de ruta: el paquete, a medida que pasa por distintos routers, se le añaden identificativos de por dónde ha pasado. Esto tiene una restricción para el número máximo de direcciones IP que pueden ser almacenadas, que es 9. Por ello, registrar una ruta con muchos saltos no es viable.
  - Registro de fechas de paso, *TimeStamp*. Es equivalente al caso anterior pero en vez de almacenar IPs, lo hacemos con las fechas.
  - Enrutamiento en origen: Se establece la ruta por la que pasará el datagrama. Este enrutamiento puede ser:
    - Estricto: **Tiene** que pasar por los puntos específicados.
    - Vago: Tiene que pasar por los puntos especificados, pero **puede** intercalarlos con otros puntos.

El apartado de opciones tiene siempre una longitud múltiplo de 4 bytes, utilizando relleno cuando se necesita.

3.4. Enrutamiento IP

Más adelante tendremos un tema dedicado exclusivamente a este tema.

En general, en enrutamiento IP, si el destino está conectado a la máquina que lo envía, lo que se hace es enviarlo directamente. Si no es así, lo que se hace es enviarse al router adyacente o gateway. Es decir, si estamos en el mismo dominio de emisión, la

máquina origen obtiene la dirección MAC de la tarjeta destino, y se envía el datagrama (IP destino, MAC destino). En caso contrario, no puede obtener la MAC del destino y coge en su defecto la MAC del gateway (IP destino, MAC del gateway).

El nivel IP de un determinado equipo se puede configurar de dos formas:

- Modo host
- Modo gateway

La consecuencia evidente de esto es que cualquier algoritmo de enrutamiento debe ser capaz de funcionar en ambas configuraciones.

Las tablas de enrutamiento (que se cargan en memoria) son las que deciden hacia dónde va cada paquete.

La máscara de red es un conjunto de 0's y 1's en donde los 1's coinciden con el identificador propio de red y 0's en donde está el rango. Para hacer subredes o *subnetting* variaremos la máscara, acotando más el rango a 255.255.255.0 por ejemplo. Para identificar una red con una sola IP, se utiliza el 32 en el último byte y la máscara correspondiente es 224 en el último byte.

Por último, es importante comentar que en el campo de Flags de la tabla de enrutamiento, el símbolo **U** indica que está activada, la **G** que se trata de un gateway, y la **H** que es un Host (no una red).

4 ARP y RARP

4.1. Introducción

Imaginemos la red de la figura 4.1 y supondremos que la máquina H le envía un datagrama de red a F. H encuentra en su tabla de enrutamiento que para la 150.244.13.0, la salida por defecto es él mismo. La dirección destino IP pasará a ser la dirección MAC asociada pero antes es necesario preguntarla utilizando un protocolo especial llamado ARP[1]. Envía una trama ARP (de broadcast) que se dedica a preguntar quién tiene la IP destino. A partir de esa información que se devuelve, el nivel de red puede bajar un nivel.

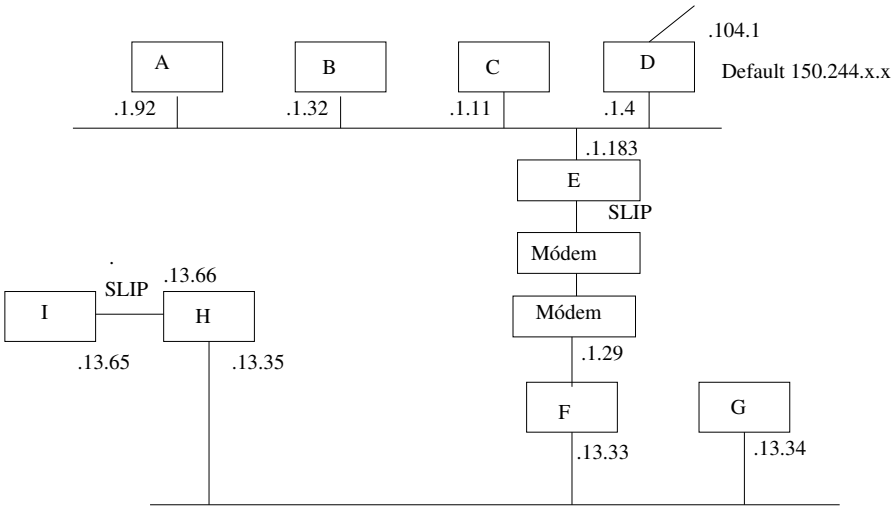


Figura 4.1: Una red de ejemplo

Supongamos ahora que la máquina H desea enviar un paquete a C. Buscará en sus tablas de enrutamiento y encontrará que su salida DEAFULT es F. Y volverá a querer saber la dirección física de F. Envía un ARP preguntando cuál es la dirección física del DEFAULT. Construimos un paquete que lleva la IP destino de C, pero dirección MAC, la de F. En general sucede que el nodo origen que está buscando la IP hace un ARP request y, en cada salto, la direccion MAC destino pasa a ser la dirección MAC origen, manteniéndose en todo momento inmutable las IPs origen y destino.

RARP obtiene una dirección IP de una dirección MAC. Para ello se utilizan ARP

[1] Address resolution protocol.

request y ARP reply (que responde con la IP si está en la red o la MAC del gateway). Generalmente, se utiliza en máquinas de red sin disco duro que necesitan *pedir* su IP a un servidor a través del dato de la MAC.

Existe algo llamado ARP caché para no estar siempre haciendo ARP request. El tiempo de expiración de una entrada en este caché puede decidirlo el administrador, pero gira entorno a los 20 minutos desde su entrada en él.

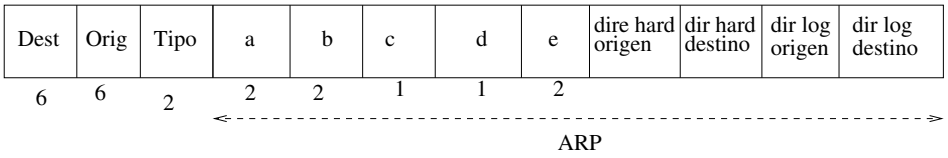


Figura 4.2: Esquema de una trama ARP

Los campos de la Trama ARP, que podemos ver en la figura 4.2 son:

- Tipo: ARP será 0x0806 y RARP será 0x8035.
- c: Tamaño de la dirección hardware
- d: Tamaño de la dirección lógica
- a: Tipo de hardware (ej: Ethernet=1)
- b: Tipo de protocolo
- e:Opciones (1=ARP request, 2=ARP Reply, 3=RARP Request, 4=RARP reply)

5 ICMP, PING y TRACEROUTE

5.1. ICMP: Internet Control Messaging Protocol

La función es enviar mensajes de control así como realizar tareas de búsqueda e información.

ICMP pertenece al nivel de red que está encapsulado sobre otro protocolo del mismo nivel, el protocolo IP. Ver figura 5.1.

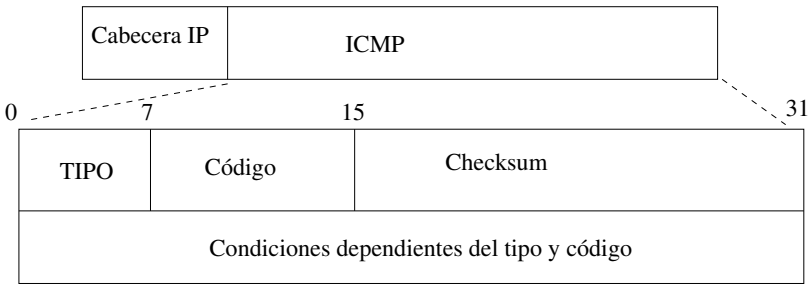


Figura 5.1: Esquema de una trama ICMP

En la siguiente tabla se detallan las combinaciones más usuales de Tipo y Código.

Tipo	Código	Descripción
0	0	Respuesta de eco (ping)
3	0	Destino inalcanzable debido a que la red a la que pertenece es inalcanzable
3	1	Destino inalcanzable debido a que la máquina es inalcanzable
3	2	Destino inalcanzable debido a que el protocolo utilizado no es válido
3	3	Destino inalcanzable debido a que el puerto utilizado no es válido

Vamos a ver un ejemplo concreto de este tipo de mensajes.

Ejemplo ¿Cuál es la **máscara de red** que se asigna en la red donde estoy yo? Se utiliza un ICMP *address mask request* para obtener esta información. Suele emplearse en máquinas sin disco duro[1] al arrancar. Un sistema alternativo para obtener la

[1]Es un mecanismo análogo al RARP empleado por máquinas sin disco duro para obtener la IP.

máscara de subred es mediante el protocolo BOOTP, que veremos en capítulos posteriores. Véase la figura 5.2 para un esquema del ICMP address mask request. Tanto el Identificador como el nº de secuencia pueden ser definidos por el remitente, sabiendo que estos mismos datos se devolverán intactos en la respuesta, muy útil para emparejar preguntas con respuestas.

Tipo:17 ó 18	Código 0	Checksum
Identificador		nº de secuencia
Máscara de red		

Figura 5.2: ICMP para averiguar la máscara de red propia

5.2. Ping y Traceroute

Estas dos herramientas resultan fundamentales en la gestión de redes ya que proporcionan información de primera mano sobre el estado de la red, equipos activos, rutas que siguen los paquetes, etc.

Ping Packet Internet Groper. Para implementar este comando se utilizan mensajes ICMP. Este comando se utiliza mucho como primera herramienta de información de red. Lo que hace el cliente es enviar un mensaje ICMP del tipo 0 (Echo Request). Lo que hace el servidor es coger ese mensaje y responder con un Echo Reply.

Traceroute El Ping sólo devuelve información del destino, no del camino intermedio. Si hacemos un traceroute nos da la información salto a salto. Sirve para encontrar un fallo en la red. Nos da más información que un Ping. Se apoya en el protocolo de transporte UDP y el campo TTL.

6 Enrutamiento IP

El enrutamiento es una de las funciones más importantes de IP. En este tema trataremos de responder a una pregunta clave: ¿Cómo enrutar paquetes IP?

Veamos un ejemplo detallado de lo que podría ser una red de redes bastante compleja en la figura 6.1.

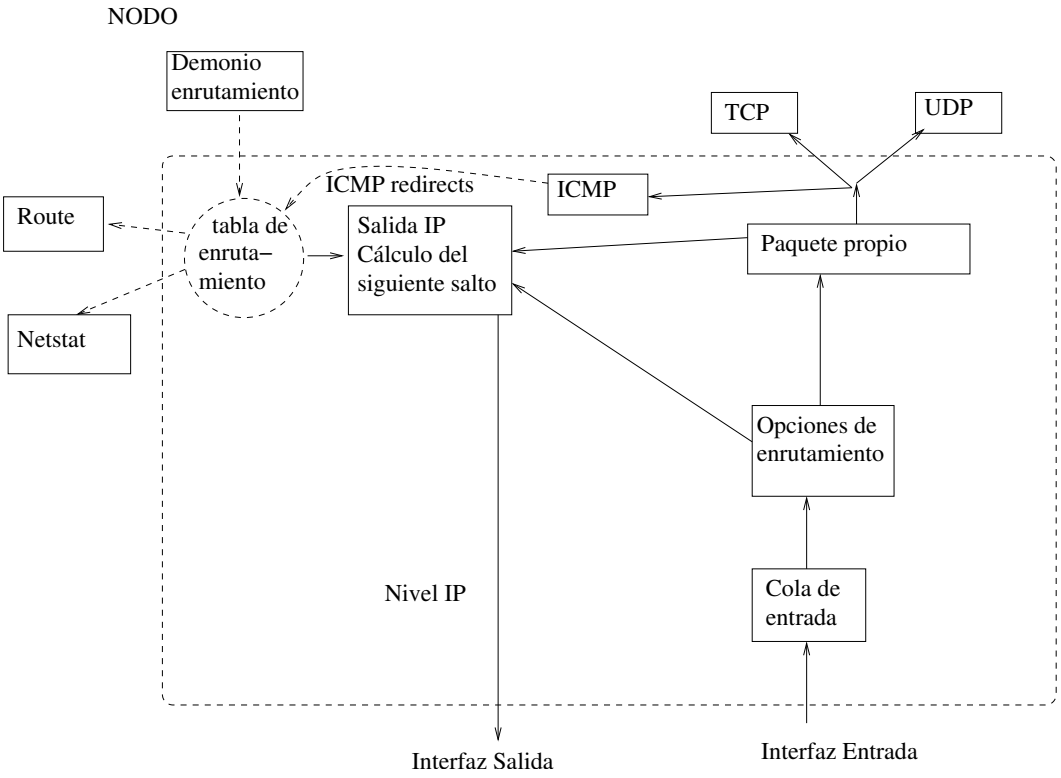


Figura 6.1: Esquema general de lo visto hasta ahora

Existe una forma totalmente estática. Se cogen las tablas de enrutamiento de los nodos y se editan las entradas a mano. Si la red es pequeña, es posible hacerlo de esta forma.

Pero otra forma más dinámica es hacer preguntas que descubran el estado de la red y determinen el mejor camino para cada vez.

Básicamente, el *mecanismo de enrutamiento* sigue una sencilla receta de tres pasos.

1. Se busca en la tabla de enrutamiento la dirección Host destino. Si no se encuentra...

2. Se busca en la tabla de enrutamiento la dirección de red destino. Si no se encuentra...

3. Se busca la entrada DEFAULT.

Los ICMP redirects los envía un router a un remitente de un datagrama IP cuando debería haber enviado un datagrama IP a otro router en vez de a él mismo. Los routers TOPLEVEL Router Domains no tienen entrada default en su trabla de enrutamiento y son los que responderá con un código 3 (redirect por tipo de servicio y Host). Adicionalmente, el router incorrecto envía el datagrama IP al router correcto.

El formato de un ICMP redirect se ve en la figura 6.2.

TIPO (5)	Código (0–3)	Checksum
IP router a ser usado		
cabecera IP + 8 primeros bytes del datagrama IP enviado		

Figura 6.2: Esquema de una trama ICMP redirect.

Hay dos maneras de configurar las tablas de enrutamiento para descubrir la red.

1. En base a **solicitudes**: una máquina aparece en la red. Envía un mensaje de solicitud de tipo broadcast a nivel de red a su dominio de emisión. Los routers responden y ya tenemos la información necesaria. El formato ICMP de solicitud es el de la figura 6.3.

ICMP solicitud

Tipo (0)	Código (0)	Checksum
Sin uso		

Figura 6.3: Esquema de una trama ICMP solicitud

2. En base a **anuncios**: no son las máquinas las que preguntan sino que son los routers que *se anuncian* y las máquinas capturan la información. Véase la figura 44. Los anuncios del router se hacen cada 7-10 minutos. El tiempo de validez de un anuncio suele ser de 30 minutos. En cuanto a nivel de preferencia, es decisión del administrador la asignación de éstos a los routers[1]. El formato ICMP de anuncio es el de la figura 6.4.

[1] más tarde pueden variar, pero incialmente es así.

ICMP anuncio

Tipo (9)	Código (0)	Checksum
nº dirección	tamaño direccion	Tiempo de validez
Dirección Router (1)		
Nivel de preferencia (1)		
Direccion Router (2)		
....		

Figura 6.4: Esquema de una trama ICMP anuncio

Bibliografía

[1] Bertrand Meyer. *Object oriented programming.* ACM Press, 1993. 9

[2] Pablo Ruiz Múzquiz. *Sistemas Operativos.* Alqua, 2004. 2

[3] W. Richard Stevens. *TCP/IP Illustrated. Volume I: The Protocols.* Addison Wesley, 2000. 3.3

Índice alfabético

Historia

0.6.0 - 15 de abril de 2004

- Primera versión pública a partir de los materiales y las clases preparados por Manel Regueiro.

Las siguientes tareas merecen atención, a juicio de los editores y autores:

- Reescribir algunos pasajes
- Mejorar algunas figuras y hacer algunas más
- Añadir capítulos sobre Nivel de Transporte
- Añadir capítulos sobre protocolos de enrutamiento

Creative Commons Deed

Attribution-NonCommercial-ShareAlike 1.0: Key License Terms

Attribution. The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original author credit.

Noncommercial. The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.

Share Alike. The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.

Whoever has associated this Commons Deed with their copyrighted work licenses his or her work to you on the terms of the Creative Commons License found here: Legal Code (the full license)

This is not a license. It is simply a handy reference for understanding the Legal Code (the full license) - it is a human-readable expression of some of its key terms. Think of it as the user-friendly interface to the Legal Code beneath. This Deed itself has no legal value, and its contents do not appear in the actual license.

Creative Commons is not a law firm and does not provide legal services. Distributing of, displaying of, or linking to this Commons Deed does not create an attorney-client relationship.

Learn how to distribute your work using this license

Manifiesto de Alqua

Origen y metas del proyecto

En 1999 fundamos el proyecto Alqua con el objetivo de promover la creación de un fondo de documentos libres de carácter científico que permita a cualquiera aprender con libertad.

Al constatar la duplicación de esfuerzos en la preparación de materiales didácticos para la física y con el deseo de compartir nuestros conocimientos, nos inspiramos en los principios de libertad que rigen el movimiento del software libre para establecer aquéllos de Alqua. Primero pensamos que lo que escribiésemos debería poder disfrutarse sin merma de libertad por las personas interesadas, y más tarde decidimos organizar nuestros esfuerzos para ayudar a otras personas que compartían nuestra visión a difundir sus saberes mediante un esfuerzo cooperativo.

Para hacer efectivos dichos principios decidimos que los documentos publicados deben ser libres en un sentido amplio: pueden reproducirse y distribuirse (gratuitamente o no, es irrelevante) pero también pueden modificarse y usarse como base para otros trabajos. A fin de evitar que estas libertades del lector-autor se restrinjan posteriormente, los documentos contienen una licencia que explica los derechos que posee y estipula que nadie que distribuya el documento, modificado o no, puede hacerlo de modo no libre.

Las ventajas de los documentos libres

Actualmente es ilegal compartir o modificar la mayoría del conocimiento científico en fuentes impresas, que suelen ser inaccesibles para la mayoría de los estudiantes y bibliotecas del mundo en virtud de su precio y se actualizan con poca frecuencia debido a su sistema de distribución tradicional.

En este contexto los documentos libres presentan ciertas ventajas.

Por una parte, en algunas disciplinas los documentos libres permiten facilitar el establecimiento de un sistema de mérito reduciendo las barreras de precio y disponibilidad. El modelo de desarrollo libre para la ciencia se apoya sobre las libertades de distribución y modificación. Éstas se ven favorecidas por el medio digital, así como por la concepción del conocimiento como un patrimonio comunitario. Todo lo anterior permite reducir el coste del documento a una cantidad marginal y anima a que lo mejor se combine con lo mejor para producir un resultado excelente a la vez que actualizado.

Por otra parte, en casos donde la evaluación del mérito es más subjetiva, los documentos libres pueden aportar una base sobre la que elaborar con un menor esfuerzo diferentes perspectivas doctrinales o estéticas, mutaciones, iteraciones y apuestas que incentivan la

creación como un aspecto más del disfrute de la obra.

En suma, los documentos libres fomentan un acceso a la cultura más justo y completo. Para algunos dominios del conocimiento científico el proceso de desarrollo libre facilita la recombinación, lo que permite la producción de obras muy sofisticadas y completas mientras que en otros ámbitos facilita la difusión de perspectivas plurales y la experimentación creativa.

Una nueva dinámica de creación y aprendizaje

Algunas personas que hemos conocido están interesadas por este modelo de colaboración, pero se preguntan qué clase de control tienen sobre sus documentos libres. La respuesta es sencilla: la licencia está diseñada de modo que a cada cual se le atribuya aquello de lo que es responsable y nada más. Para ello, se incluye en el documento una sección en la que se explica quién hizo qué y cuándo lo hizo.

Uno de los efectos más interesantes de introducir los documentos libres en el aula es que difuminan la frontera entre quien aprende y quien enseña. Los documentos libres son un puente para establecer contacto con una comunidad de interés mucho más vasta que la del centro educativo, permitiendo el aprendizaje continuo y fomentando una experiencia plural y transformadora: el criterio para participar en un documento es, solamente, hacerlo bien.

Un autor puede pensar que distribuir su documento bajo un copyright que restringe la libertad de copia es *más rentable* que otorgar mayores libertades. Esto no es necesariamente así, por varias razones.

En primer lugar, libre no quiere decir gratuito. Una editorial puede publicar un documento libre obteniendo beneficio de ello. De hecho, es una buena idea hacerlo dado lo agradable que resulta manejar un libro bien encuadernado. También los autores pueden aceptar una compensación de los lectores por su trabajo en un determinado documento.

En segundo lugar, la mayor parte de los autores son primeramente lectores. Cabe esperar, pues, que para la mayoría el enorme ahorro derivado del acceso a *muchos* documentos libres supere holgadamente el beneficio económico obtenido de *unos pocos* documentos no libres. La experiencia del software libre lo avala.

Finalmente, no se puede poner precio al beneficio social derivado de la existencia de documentos libres. Gracias a los derechos que uno posee sobre un documento libre puede adaptarlo para un curso académico eliminando lo que no es pertinente o es demasiado avanzado y complementando el tema con nuevas aportaciones, desde ejercicios o diagramas hasta apartados enteros.

Pensamos que las universidades u otras instituciones educativas podrían cumplir mejor su función social poniendo a disposición de la sociedad que las financia, en condiciones de libertad, su patrimonio más importante: el conocimiento.

El modelo de cooperación que proponemos (que anima al trabajo en equipo aunque no lo impone) permite abrir todas estas perspectivas y algunas más. Alqua intenta ofrecer los medios para esta tarea y relacionar, a través de los documentos libres, a los que tienen saberes que comunicar y a los que sienten curiosidad por dichos saberes.

Conclusión

Alqua tiene una tarea muy ilusionante y tan ambiciosa que sólo es factible en comunidad. Por ello, pedimos a las personas que forman parte de instituciones o empresas que colaboren con Alqua para que éstas apoyen económicamente el proyecto o patrocinen ediciones impresas y donaciones a las bibliotecas públicas. Ciertamente, los medios materiales son necesarios, pero inútiles si, a nivel particular, no contamos con tu participación como individuo, aprendiendo y enseñando, para que los documentos libres en marcha y otros nuevos alcancen los altos niveles de calidad a los que aspiramos.

Te invitamos a construir un patrimonio científico que nos pertenezca a todos.

Versión 2.0, marzo de 2003

El proyecto libros abiertos de Alqua

El texto que sigue es una explicación de qué es y cómo se utiliza un libro abierto y contiene algunas recomendaciones sobre cómo crear un libro abierto a partir de un documento de Alqua. Si estás leyendo estas páginas como anexo a otro documento, éste es casi con seguridad un *documento libre* de Alqua; libre en el sentido descrito en el manifiesto de Alqua y las directrices para documentos libres de Alqua . Si has obtenido dicho documento en un centro público, como una biblioteca, entonces es además un *libro abierto* de Alqua.

Qué son los libros abiertos

Los libros abiertos son ediciones impresas de los documentos libres de Alqua que se pueden obtener en las bibliotecas u otros centros públicos. La particularidad de los libros abiertos no reside en *qué contienen* (el contenido es el mismo que el de los libros descargados de la red) sino en *cómo pueden utilizarse*.

Al igual que los usuarios de Alqua a través de la red forman una comunidad de interés que aprende colectivamente leyendo los documentos, discutiendo sobre ellos y modificándolos para adaptarlos a propósitos muy variados, los lectores de una biblioteca constituyen también una comunidad. El ciclo de vida de un documento libre es de constante realimentación: las nuevas versiones son leídas, corregidas o quizá bifurcadas, lo que conduce a la publicación de nuevas versiones listas a su vez para un nuevo ciclo del proceso. ¿Por qué no abrir esa dinámica a la participación de comunidades que no se articulan en torno a la red?. No todos disponen del tiempo o los medios para participar efectivamente en el proceso de mejora de los documentos a través de la red, que es la aportación diferencial más importante de los libros libres respecto a los no libres. Por ello queremos poner a disposición de las bibliotecas *libros abiertos* que faciliten lo siguiente:

- El acceso de personas sin recursos informáticos al conocimiento que su estudio proporciona.
- La posibilidad de contribuir a la mejora de dichos documentos por parte de la amplísima comunidad de lectores de las bibliotecas, sin otro medio que un lápiz o una pluma.
- La formación de grupos de interés locales: compartir a través de un documento libre puede compartir su proceso de aprendizaje con personas interesadas por temas afines.

- La constitución, hasta en los centros que cuentan con una financiación más débil, de un fondo de documentos libres que cubra áreas del conocimiento que su presupuesto no permite afrontar.

¿Cómo puedo contribuir a los libros abiertos?

Sólo tienes que utilizarlos como si fuesen tuyos, pero recordando que compartes tu experiencia de aprendizaje con otras personas.

Por ejemplo, contrariamente a lo que harías con cualquier otro libro de la biblioteca puedes escribir en los márgenes de los libros abiertos tus propios comentarios: correcciones, aclaraciones, bibliografía relacionada... Intenta hacerlo ordenadamente, de modo que no interrumpa la lectura.

Si quieres compartir algún razonamiento más largo, puedes utilizar tus propias hojas e incorporarlas al final del documento, poniendo una nota donde corresponda. En este caso, no olvides firmar tu contribución con un nombre o seudónimo y, opcionalmente, una dirección de correo electrónico u otra forma de contacto.

Cualquiera que pueda participar a través de la red puede incorporar tus contribuciones a la versión que se distribuye en línea, con la ayuda de la comunidad de Alqua. De esta manera abrimos el mecanismo de colaboración a los lectores que no están acostumbrados al ordenador o prefieren no usarlo. La firma permite atribuir la autoría en el caso de que los cambios se incorporen y establecer contacto al respecto. Damos por hecho que al escribir tus aportaciones en un libro abierto estás de acuerdo con que sean libremente utilizadas (en el sentido descrito en las directrices para documentos libres ya mencionadas) y por lo tanto incorporadas a las sucesivas versiones digitales.

Los libros abiertos pueden ser editados de modo que se puedan separar sus hojas porque no hay inconveniente en que éstas sean fotocopiadas: no tenemos que usar la encuadernación como un modo de evitar la reproducción, puesto que no sólo no la prohibimos sino que animamos a ella. Por tanto, una vez que obtengas un ejemplar en préstamo puedes llevar contigo sólo la parte que estés utilizando.

Como lector, tu ayuda es necesaria no sólo para mejorar los documentos, sino para que existan: hace falta imprimir, encuadernar y donar a una biblioteca un documento libre de Alqua para que se convierta en un *libro abierto*.

Quienes tengan acceso a una impresora pueden ayudar a que los *libros abiertos* perduren en la biblioteca sustituyendo las partes deterioradas por el uso y actualizando periódicamente el documento impreso. Para facilitar la tarea a continuación proponemos un sistema de encuadernación modular.

¿Cómo puedo publicar un libro abierto?

Los pasos para publicar un libro abierto son los siguientes:

1. Imprimir la versión más actualizada del documento tal cual se distribuye en la página web de Alqua, http://alqua.org

2. Conseguir una encuadernación modular – sugerimos un archivador de anillas con una ventana o de portada transparente. Ello permite llevar consigo sólo la parte del libro que se está usando y añadir hojas con nuevas contribuciones.

3. Encuadernar el libro y situar el título, el autor y la clasificación decimal universal en su lomo y tapas.

4. Si puedes, adjuntar al archivador una copia del CD-ROM de documentos libres de Alqua .

5. Donarlo a la biblioteca y comunicar a Alqua la edición, escribiendo a librosabiertos@alqua.org .

Se trata de un proceso sencillo al alcance tanto de particulares como de bibliotecas y otras instituciones, con un coste marginal que no se verá significativamente incrementado por la conservación y actualización puesto que se puede mantener la encuadernación y sustituir solamente las páginas impresas.

En conclusión

El proyecto *libros abiertos*, consecuencia de los principios establecidos en el manifiesto de Alqua , persigue dotar a las bibliotecas de un fondo amplio y asequible de documentos libres y a la vez facilitar la participación de los usuarios en el proceso creativo del que son fruto.

Tu ayuda es esencial para que el proyecto alcance estos objetivos.

Redes y sistemas de comunicación

Pablo Ruiz Múzquiz

descripción

En este documento se dan a conocer las técnicas y diseño subyacentes a las redes de computadores utilizadas en la actualidad. Sirve como texto de apoyo para un primer curso de redes en donde se traten los protocolos de red, enrutamiento y subneting.

requisitos

- Manejo elemental de un ordenador conectado a la red
- Conocimientos básicos de electrónica

http://alqua.org/libredoc/RSC

Aprende en comunidad - http://alqua.org ◁

otros documentos libres

Variedades, tensores y física - Óptica electromagnética - Ecuaciones diferenciales ordinarias - Introducción a la física cuántica, segunda parte - Redes y sistemas - Sistemas Operativos - Geometría simpléctica - Física del láser - Análisis funcional - Geografía general de España (en preparación).

http://alqua.org/libredoc/

alqua, madeincommunity