Universidad Antonio de Nebrija

IT3518 Teleinformática y Redes

Prof: Javier Meijomence Taboada

Introducción

Para un estudio adecuado de la disciplina que nos ocupa, comenzaremos estableciendo brevemente cuál es su ámbito de acción, así como los factores que motivaron su aparición y contribuyeron a su desarrollo.

En esencia, la Telemática se ocupa del estudio de los sistemas de comunicación entre ordenadores. Se trata por lo tanto de una **disciplina integradora de las Telecomunicaciones y la informática**. Su aparición viene motivada por la progresiva convergencia de las Telecomunicaciones y el procesado electrónico de datos.

Los primeros ordenadores disponibles comercialmente resultaban demasiado primitivos para permitir las comunicaciones. La evolución de la tecnología y las mejoras del software permitieron aumentar las prestaciones de los equipos. Se mejoraron los dispositivos de almacenamiento y se desarrollaron sistemas operativos capaces de soportar varias tareas en tiempo compartido. De esta forma, podían mantenerse varias tareas del sistema activas mientras se atendía a otros usuarios de forma interactiva.

La década de los sesenta resultó de capital importancia para el desarrollo de la Telemática. Los trabajos llevados a cabo por la Agencia de Investigación de Proyectos Avanzados (**ARPA**) configuraron un primer modelo de sistema teleinformático, que heredaba de los sistemas telefónicos la idea de una red de comunicaciones, proveedora de servicios a una serie de centros usuarios.

Desde 1970 asistimos a un espectacular crecimiento de las redes. Aparecieron arquitecturas concebidas para sistemas distribuidos: **SNA** de IBM (1974), **DNA** de Digital (1976) etc. Este gran desarrollo trajo como consecuencia la necesidad del establecimiento de estándares que hicieran posible la comunicación entre sistemas producidos por distintos fabricantes, lo que conduce a los denominados sistemas **abiertos**.

En 1977 el comité técnico número 97 (TC-97) de la Oficina Internacional de Estándares **ISO** creó un nuevo subcomité (SC-16) encargado de resolver esta situación de carencia de una filosofía y estructura de comunicaciones común. El objetivo de este subcomité fue el establecimiento de una arquitectura que proporcionara el marco de definición. El desarrollo y la validación de estándares en la nueva generación de sistemas de información distribuida.

A través de este texto, utilizaremos el concepto de **redes de ordenadores** para dar a entender una colección *interconectada* de ordenadores *autónomos*. Se dice que dos ordenadores están interconectados, si estos son capaces de intercambiar información. La conexión no necesita hacerse a través de un hilo de cobre también puede hacerse mediante el uso de láser, microondas y satélites de comunicaciones. Al indicar que los ordenadores son autónomos, queremos excluir de nuestra definición a los sistemas en donde existe una clara relación maestro/esclavo. Si un ordenador puede forzosamente arrancar, parar o controlar a otro, éstos no se consideran autónomos. Un sistema constituido por una unidad de control y muchos esclavos no es una red, ni tampoco lo es un ordenador grande con lectoras de tarjetas de control remoto, impresoras y terminales.

Existe en la literatura una notable confusión entre una red de ordenadores y un **sistema distribuido.** La clave de la diferencia es que en un sistema distribuido la existencia de múltiples ordenadores autónomos es transparente al usuario (es decir, no le es visible). Él puede teclear un

comando para correr un programa, y observar que corre. El hecho de seleccionar el mejor procesador, encontrar y transportar todos los archivos de entrada al procesador y poner los resultados en el lugar apropiado, depende del sistema operativo.

En otras palabras, el usuario de un sistema distribuido no tiene conocimiento de que hay múltiples procesadores, más bien se ve al sistema como un monoprocesador virtual. La asignación de trabajos al procesador y archivos a discos, el movimiento de archivos entre donde se almacenan y donde son necesarios, y todas las demás operaciones del sistema, deben ser automáticas.

Con una red, el usuario debe *explícitamente* entrar en una máquina, *explícitamente* enviar trabajos remotos, *explícitamente* mover archivos y, por lo general, gestionar de manera personal toda la administración de la red. Con un sistema distribuido nada se tiene que hacer de forma explícita, todo lo hace de manera automática el sistema sin que el usuario tengo conocimiento de ello.

Un sistema distribuido es efectivamente un caso especial de una red, aquél cuyo software da un alto grado de cohesión y transparencia. Por lo tanto, la diferencia entre una red y un sistema distribuido está más bien en el software (en especial el sistema operativo) que en el hardware.

Puesto en una forma más general, el tema aquí consiste en **compartir recursos**, y el objetivo es hacer que todos los programas, datos y equipos estén disponibles para cualquiera de la red que así lo solicite, sin importar la localización física del recurso y del usuario. En otras palabras, el hecho de que el usuario se encuentre a 1000km de distancia de los datos, no debe evitar que éste los pueda utilizar como si fueran originados localmente. Otro aspecto de compartir recursos es el relacionado con la compartición de la carga.

Un segundo objetivo consiste en proporcionar una **alta fiabilidad**, al contar con fuentes alternativas de suministro. Por ejemplo, todos los archivos podrían duplicarse en dos o tres máquinas, de tal manera que si una de ellas no se encuentra disponible (como consecuencia de un fallo de hardware), podría utilizarse alguna de los otras copias. Además, la presencia de múltiples CPU significa que si una de ellas deja de funcionar, las otras pueden ser capaces de encargarse de su trabajo, aunque se tenga un rendimiento global menor.

Otro objetivo es el **ahorro económico**. Los ordenadores pequeños tienen una mejor relación costo/rendimiento, comparada con la ofrecida por las máquinas grandes. Estas son, a grandes rasgos, diez veces más rápidas que el más rápido de los microprocesadores, pero su costo es miles de veces mayor. Esto conduce al concepto de redes con varios ordenadores localizados en el mismo edificio. A este tipo de red se le denomina **LAN** (Local Area Network) o **Red de Area Local**, en contraste con lo extenso de una **WAN** (Wide Area Network) o **Red de Area Extendida**.

Otro objetivo del establecimiento de una red de ordenadores no tiene nada que ver con la tecnología. Una red de ordenadores puede proporcionar un poderoso **medio de comunicación** entre personas que se encuentran muy alejadas entre si.

Distancia entre procesadores	**Procesadores ubicados en el mismo...**	**Ejemplo**
0,1m	La tarjeta del circuito	Máquina de flujo de datos
1m	El sistema	Multiprocesador
10m	El cuarto	Red Local
100m	El edificio	“”
1Km	La ciudad	Red de gran alcance
100Km	El país	“”
1000Km	El continente	Interconexión de redes de gran alcance
10,000Km	El planeta	“”

En la tabla anterior, se muestra la clasificación de sistemas multiprocesadores distribuidos de

acuerdo con su tamaño físico. En la parte superior se encuentran las **máquinas de flujo de datos**, que son ordenadores con un alto nivel de paralelismo y muchas unidades funcionales trabajando en el mismo programa. Después vienen los **multiprocesadores**, que son sistemas que se comunican a través de memoria compartida. Enseguida de los multiprocesadores se muestran las verdaderas redes, que son ordenadores que se comunican por medio del intercambio de mensajes. Finalmente, a la conexión de dos más redes se le denomina **interconexión de redes**.

Una de las primeras y más importantes redes abiertas es la **Arpanet** (USA). Su nombre viene de *Advanced Research Projects Agency*, que pertenece al **DOD** o *Department of Defense*. A finales de los años 60 esta red conectaba los departamentos de ciencias de varias universidades y algunas empresas privadas. Actualmente cubre medio globo terrestre, desde Hawaii hasta Noruega. Mucho del presente conocimiento sobre redes es consecuencia directa del proyecto *Arpanet*.

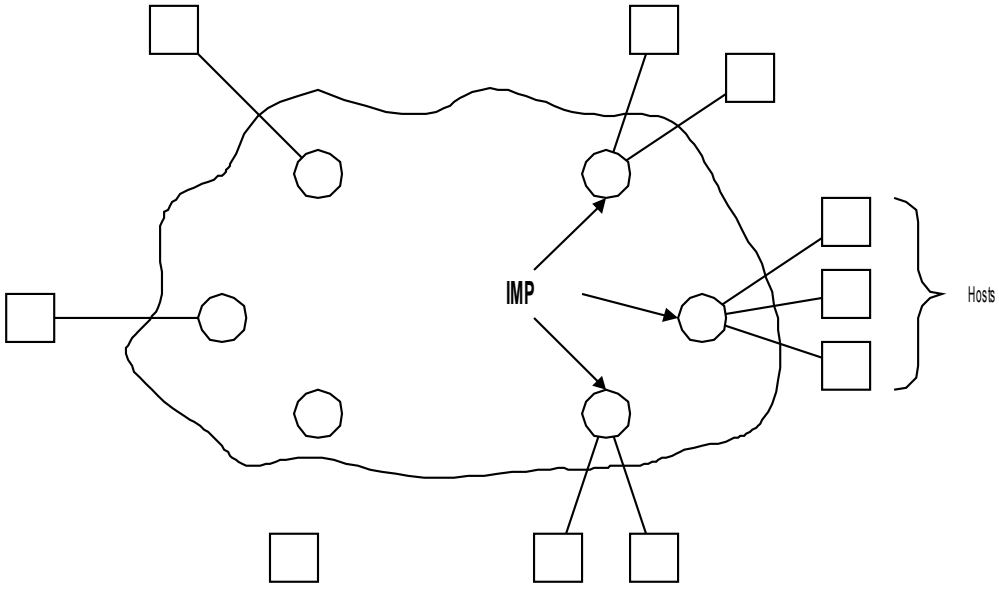
Arpanet diferencia en una red los siguientes elementos:

- **Host**: Máquinas que ejecutan procesos de usuario (aplicaciones). En esta definición se incluyen los mecanismos de acceso a la sub-red.

- **Sub-Red**: Mecanismos que permiten el paso de información de un host a otro. En la mayor parte de las redes de área extendida, una sub-red consiste de dos componentes diferentes: las líneas de transmisión y los IMP:

1. **Líneas de transmisión**; también se denominan circuitos o canales. Es el medio físico a través del cual se realiza la transmisión de los datos.

2. **I.M.P. (Interface Message processor)**: también llamados nodos, conmutadores de paquetes, ordenadores de comunicaciones, intercambiadores de datos, sistemas intermedios, etc. Son ordenadores especializados que sólo ejecutan programas de comunicaciones. Su misión es habilitar una conexión entre en dos o más líneas de transmisión. Cuando los datos llegan por una línea de entrada, el elemento de conmutación deberá seleccionar una línea de salida para reexpedirlos.



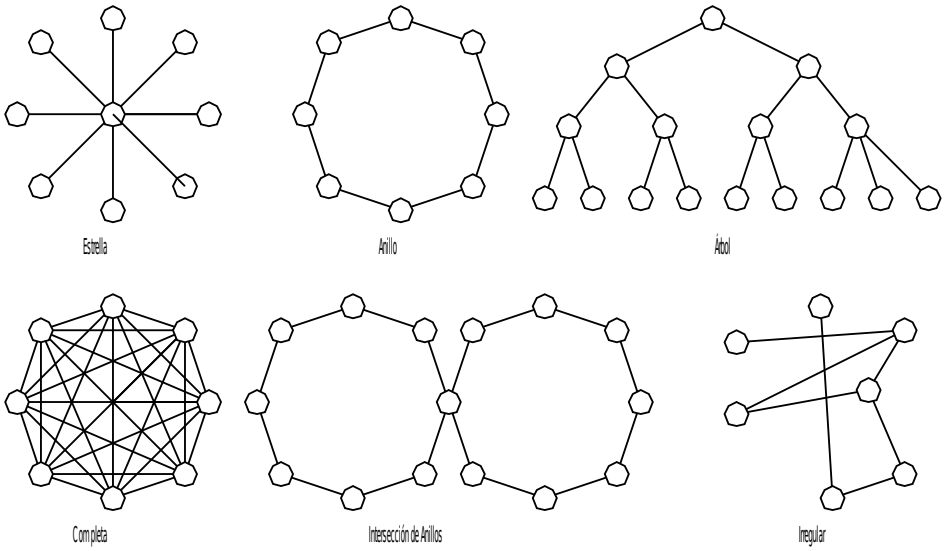
Relación entre hosts e IMPs

En términos generales, puede decirse que hay dos tipos de diseños para la sub-red de comunicación:

- Canales **punto a punto** (point to point).
- Canales de difusión o **multipunto** (broadcast).

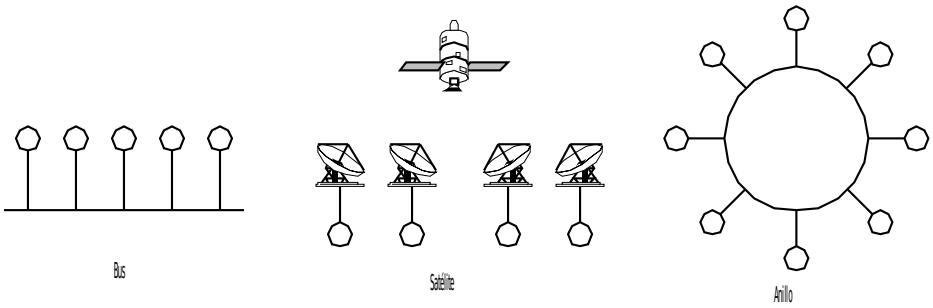
En el primero de ellos, la red contiene varios cables o líneas telefónicas alquiladas, conectando cada una de ellas un par de IMP. Si dos IMP desean comunicarse y no comparten un cable común, deberán hacerlo indirectamente a través de otros IMP. Cuando un mensaje (que en el contexto de sub-red normalmente se denomina **paquete** o **packet**) se envía de un IMP a otro, a través de uno o más IMP intermediarios, el paquete se recibe íntegramente en cada uno de estos IMP intermediarios. Se almacenará ahí y no continuará su camino hasta que la línea de salida necesaria para reexpedirlo esté libre. La sub-red que utiliza este principio se denomina sub-red **punto a punto**, **de almacenamiento y reenvío** o de **conmutación de paquetes**. Casi todas las redes de área extendida tienen sub-redes del tipo de almacenamiento y reenvío.

Un aspecto importante de diseño, cuando se utiliza una sub-red punto a punto, consiste en considerar cómo deberá ser la topología de interconexión de los IMP. En La redes locales que se diseñaron como tales, tienen por lo general una topología simétrica. A diferencia de éstas, las redes de área extendida tienen típicamente topologías irregulares.



Algunas topologías punto-punto

La estructura multipunto se emplea como un segundo tipo de arquitectura de comunicación y la utilizan la mayoría de las redes de área local y un número muy reducido de redes de área extendida. En una red de área local, el IMP se reduce a un solo chip, el cual se incluye en el interior del host, de tal manera que siempre habrá un host para cada IMP, mientras que en una red de área extendida podrá tener varios hosts por IMP.



Comunicación de subredes de difusión

Los sistemas multipunto (o de difusión) tienen un solo canal de comunicación que, a su vez, es

compartido por todas las máquinas que constituyen la red. Los paquetes que una máquina cualquiera envía, son recibidos por todas las demás. El campo de dirección, localizado en el interior de un paquete, especifica a quién va dirigido. En el momento en que se recibe un paquete, se verifica el campo de dirección y, si el paquete está destinado a otra maquina, éste simplemente se ignora. En cualquier instante, una máquina conectada a una sub-red multipunto, tiene la función de maestra y está capacitada para transmitir. El resto de las máquinas no pueden enviar. Se necesita un mecanismo de arbitraje para resolver los conflictos en el momento en que dos o más máquinas quieren transmitir a la vez. Este mecanismo de arbitraje puede estar centralizado o distribuido. Resumiendo, podemos dividir las topologías multipunto en:

- **Estáticas**: Cada IMP posee el canal para transmitir durante un tiempo predeterminado (*Quantum*), que se desperdicia en caso de que el IMP no tenga nada que transmitir.

- **Dinámicas**: Dentro de las cuales existen dos tipos:

  - *Centralizadas:* Un IMP que desea transmitir lo solicita a un elemento común que arbitra.

  - *Descentralizadas:* Los IMP deciden por sí mismos si pueden o no tomar el canal para transmitir.

Modelo de Referencia OSI

2.1.1. Justificación del modelo

Otra visión del tema vendría dada por la enumeración de las tareas básicas que deben llevarse a cabo en un sistema de comunicación de datos. A saber:

- **Utilización del sistema de transmisión:** Este primer ítem hace referencia a la necesidad de hacer un uso eficiente de los servicio de transmisión que suelen compartirse entre varios dispositivos de comunicación. Existen variedad de técnicas (conocidas como *multiplexación*) para repartir la capacidad del medio de transmisión entre varios usuarios. Asimismo, pueden requerirse técnicas de control de la congestión para asegurar que el sistema no se vea desbordado por la demanda excesiva de servicios de transmisión.

- **Generación de señales:** Todas las formas de comunicación que mencionaremos dependen en última instancia de la transmisión de señales electromagnéticas en el seno de un medio. Una vez establecida la interfaz, se requiere la generación de señales para la comunicación. Las propiedades de la señal, tales como forma de onda e intensidad, deben hacer que ésta resulte adecuada para propagarse por el medio de transmisión e interpretable como datos por el receptor.

- **Sincronización:** Debe hacer alguna forma de sincronización entre transmisor y receptor. El receptor debe poder determinar cuando una señal empieza a llegar y cuando termina. Debe conocer también la duración de cada elemento de la señal.

- **Gestión de intercambio:** Incluye aspectos como decidir si ambos usuarios pueden transmitir simultáneamente o por turno, la cantidad de datos que pueden incluirse en un envío, el formato de los datos y las medidas a tomar en caso de error.

- **Detección y corrección de errores:** Necesario en circunstancias en las que no pueden tolerarse fallos. Por ejemplo: Transferencia de ficheros.

- **Control de flujo:** Necesario para evitar que el emisor desborde al receptor.

- **Direccionamiento y encaminamiento:** Cuando un elemento de comunicación es compartido por más de dos dispositivos, el sistema emisor debe identificar el destino deseado. El sistema de transmisión debe garantizar que únicamente el sistema receptor recibe los datos. El sistema de transmisión puede ser una red que permita varias rutas posibles entre fuente y destino, debiéndose elegir un camino entre los posibles.

- **Recuperación:** Concepto distinto a la corrección de errores. Las técnicas de recuperación son necesarias en aquellos casos en los que el intercambio de información, por ejemplo, acceso a bases de datos o transferencia de ficheros, queda interrumpido debido a fallos en el sistema. El objetivo es reanudar el intercambio en el punto de interrupción o al menos restaurar el estado de los sistemas involucrados.

- **Formato de mensajes:** Ambas partes deben estar de acuerdo con el formato de los datos que se transmiten. Por ejemplo, deben utilizar el mismo código binario para los caracteres.

2.1.2. Jerarquías de protocolos

Cuando los dispositivos que van a intervenir en la comunicación son ordenadores hace falta un alto grado de cooperación entre ellos. Por ejemplo. En una transmisión de ficheros, además de las tareas usuales relacionadas con la comunicación de datos, es necesario asegurarse que el sistema destino está listo para recibir datos, para aceptar y almacenar el fichero, si el formato de ficheros que usan lo dos sistemas es incompatible uno u otro debe realizar una transformación, etc.

Al hablar de redes y de comunicaciones entre ordenadores resultan fundamentales dos conceptos:

- Protocolos.
- Arquitectura de comunicación.

Los protocolos se utilizan para la comunicación entre **entidades** de diferentes sistemas. Los términos *Entidad* y *Sistema* se utilizan aquí en un sentido muy general. Ejemplos de entidades son programas de aplicación de usuario, paquetes de transferencia de ficheros, sistemas de manejo de bases de datos y terminales. Ejemplos de sistemas son ordenadores, terminales y sensores remotos. En general, una entidad es *algo* capaz de enviar o de recibir información, y un sistema es un objeto que contiene una o más entidades. Para que dos entidades puedan comunicarse deben *hablar el mismo idioma*. Qué se comunica, cómo se comunica y cuando se comunica debe cumplir ciertas convenciones entre las entidades involucradas. Este conjunto de convenciones constituye un **protocolo,** que puede definirse como *el conjunto de reglas que gobiernan el intercambio de datos entre dos entidades.*

La tarea de la comunicación entre dos entidades de diferentes sistemas es demasiado complicada para ser manejada por un simple proceso o módulo. En lugar de manejar un único protocolo, implementaremos las funciones de comunicación mediante un conjunto de protocolos estructurados. La organización de estos protocolos se realiza mediante una serie de **capas** o **niveles**, con objeto de reducir la complejidad de sus diseño. Cada una de ellas se construye sobre su predecesora. El número de capas, el nombre, contenido y función de cada una varían de una red a otra. Sin embargo, en cualquier red, el propósito de capa es ofrecer ciertos servicios a las capas superiores, liberándolas del conocimiento detallado sobre cómo se realizan dichos servicios.

La capa *n* en una máquina conversa con la capa *n* de otra máquina. Las reglas y convenciones utilizadas en esta conversación se conocen conjuntamente con **protocolo de la capa *n***. A las entidades que forman las capas correspondientes en máquinas diferentes se les denomina **procesos pares** (**igual a igual**). En otras palabras, son los procesos pares los que se comunican mediante el uso del protocolo.

En realidad no existe una transferencia directa de datos desde una capa *n* de una máquina a la capa *n* de otra; sino, más bien, cada capa pasa la información de datos y control a la capa inmediatamente inferior, y así sucesivamente hasta que se alcanza la capa localizada en la parte más baja de la estructura. Debajo de la capa 1 está el **medio físico**, a través del cual se realiza la comunicación real.

Entre cada par de capas adyacentes hay un **interfaz**, la cual define los servicios y operaciones primitivas que la capa inferior ofrece a la superior. El diseño claro y limpio de una interfaz, además de minimizar la cantidad de información que debe pasarse entre capas, hace más simple la sustitución de la realización de una capa por otra completamente diferente (por ejemplo, todas las línea telefónicas se reemplazan por canales satélite).

Al conjunto de capas (con sus interfaces) y protocolos se le denomina **arquitectura de red**.

2.1.3. Estándares

En la industria se aceptó hace ya bastante tiempo, la necesidad de estándares que gobernaran las acciones y las características físicas y eléctricas de los equipos de comunicación. Este punto de vista, sin embargo ha tardado en imponerse en la industria de los ordenadores.

Entre las organizaciones más importantes que han colaborado en el desarrollo de estándares en nuestra área tenemos:

- **ISO *(International Organization for Standarization):*** Agrupa a 89 países, se trata de una organización voluntaria, no gubernamental, cuyos miembros han desarrollado estándares para las naciones participantes. Uno de sus comités se ocupa de los sistemas de información. Han desarrollado el modelo de referencia **OSI** (Open Systems Interconnection) y protocolos estándar para varios niveles del modelo.

- **CCITT *(Comité Consultatif International de Télégraphique et Téléphonique):*** Organización de la Naciones Unidas constituida, en principio, por las autoridades de Correos, Telégrafos y Teléfonos (PTT) de los países miembros. Estados Unidos está representado por el departamento de Estado. Se encarga de realizar recomendaciones técnicas sobre teléfono, telégrafo e interfaces de comunicación de datos, que a menudo se reconocen como estándares. Trabaja en colaboración con **ISO** (que en la actualidad es miembro de CCITT).

- **EIA *(Electronic Industries Association):*** Asociación vinculada al ámbito de la electrónica. Es miembro de **ANSI**. Sus estándares se encuadran dentro del nivel 1 del modelo de referencia **OSI**.

- **ANSI *(American National Standard Institute):*** Asociación con fines no lucrativos, formada por fabricantes, usuarios, compañías que ofrecen servicios públicos de comunicaciones y otras organizaciones interesadas en temas de comunicación. Es el representante estadounidense en **ISO**. Que adopta con frecuencia los estándares **ANSI** como estándares internacionales.

La aceptación mayoritaria de los diferentes estándares ha supuesto un crecimiento de la oferta de equipos compatibles de diversos fabricantes, proporcionando a los usuarios una mayor libertad de elección, favoreciendo la competencia entre fabricantes e incrementando la demanda de equipos compatibles.

Sin embargo los estándares llevan también aparejados ciertos inconvenientes, como puede ser la introducción de retraso tecnológico, que ralentiza nuevos desarrollos y la multiplicidad de estándares no compatibles.

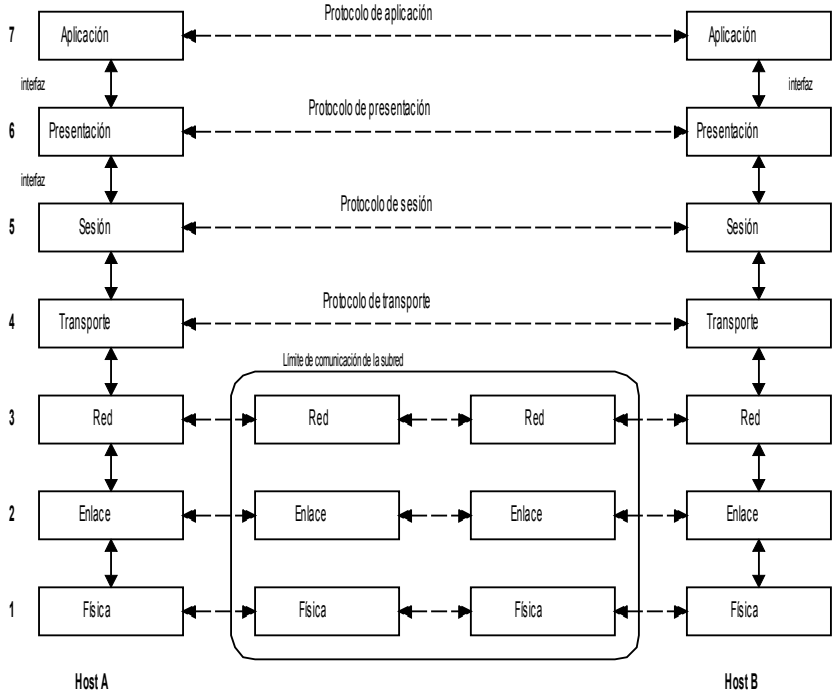
En 1977 la Organización INTERNACIONAL DE Estandarización **ISO** estableció un subcomité encargado de diseñar una arquitectura de comunicación. El resultado fue el *Modelo de referencia para la Interconexión de Sistemas Abiertos **OSI***, adoptado en 1983, que establece unas bases que permiten conectar sistemas *abiertos* para procesamiento de aplicaciones distribuidas. Se trata de un marco de referencia para definir estándares que permitan comunicar ordenadores heterogéneos.

Dicho modelo define una arquitectura de comunicación estructurada en siete niveles verticales. Cada nivel ejecuta un subconjunto de las funciones que se requieren para comunicar con el otro sistema. Para ello se apoya en los servicios que le ofrece el nivel inmediato inferior y ofrece sus servicios al nivel que está por encima de él. Idealmente, los cambios que se realicen en un nivel no deberían afectar a su nivel vecino mientras ni se modifiquen los servicios que le ofrece.

La tarea del subcomité **ISO** fue definir el conjunto de niveles y los servicios proporcionados por cada nivel. Los principios aplicados para establecer un nivel fueron los siguientes:

- Diferentes niveles deben corresponder a diferentes niveles de abstracción en el manejo de los datos (por ejemplo diferencias en la morfología, la sintaxis, la semántica).
- Cada nivel debe ejecutar una función bien definida.
- Aprovechar la experiencia de protocolos anteriores. Las fronteras de niveles deben situarse donde la experiencia ha demostrado que son convenientes.
- Establecer las divisiones de los niveles de forma que se minimice el flujo de información entre ellos.
- El número de niveles debe ser suficiente para que no agrupen funciones distintas, pero no tan grande que haga la arquitectura inmanejable.
- Permitir que las modificaciones de funciones o protocolos que se realicen en un nivel no afecten a los niveles contiguos.
- Cada nivel debe interaccionar únicamente con los niveles contiguos a él (superior e inferiormente).

2.2.1. Niveles OSI



Arquitectura de red en el modelo OSI

Los siete niveles que configuran el modelo **OSI** suelen agruparse en dos bloques. Los tres niveles inferiores (nivel físico, nivel de enlace y nivel de red) constituyen el bloque de transmisión. Son niveles dependientes de la red de conmutación utilizada para la comunicación entre los dos sistemas. Por el contrario, los tres niveles superiores (nivel de sesión, de presentación y de aplicación) son niveles orientados a la aplicación y realizan funciones directamente vinculadas con los procesos de aplicación que desean comunicarse. El nivel intermedio (nivel de transporte) enmascara a los niveles orientados a la aplicación, el funcionamiento detallado de los niveles dependientes de la red.

Pasemos a discutir brevemente cada nivel:

- **Nivel físico:** El nivel físico se ocupa de la transmisión de bits a través de un canal de comunicación. Regula aspectos de la comunicación como el tipo de señal, el esquema de codificación, el modo de comunicación (dúplex, semi-dúplex o símplex) y, en general, todas las cuestiones eléctricas, mecánicas y de procedimiento en la interfaz física entre los dispositivos que se comunican.

- **Nivel de enlace de datos:** Mientras el nivel físico proporciona únicamente un servicio de transmisión de bits a través de una canal, el nivel de enlace tiene el objetivo de hacer la comunicación fiable y proporcionar los medios para activar, mantener y desconectar el enlace. El principal servicio proporcionado por este nivel es el de detección y control de errores. Con un protocolo suficientemente elaborado, el nivel de red puede asumir una transmisión libre de errores a través del enlace. Pero, hay que tener en cuenta, que si los sistemas que se comunican no están directamente conectados, sino a través de varios enlaces, cada uno funcionará de forma independiente y los niveles superiores no estarán eximidos de la responsabilidad de controlar los errores.

- **Nivel de Red:** El servicio básico de este nivel es proporcionar transferencia de datos transparente entre entidades de transporte. Es decir, libera al nivel de transporte de la necesidad de conocer el funcionamiento interno de la subred. Entre sus principales funciones se encuentran el encaminamiento y el control de la congestión.

- **Nivel de Transporte:** Es el primer nivel que lleva a cabo comunicación *extremo - extremo*, condición que se mantiene en los niveles superiores a él. Su objetivo es proporcionar mecanismos que garanticen que el intercambio de datos entre procesos de distintos sistemas se lleve a cabo de forma fiable. El nivel de transporte debe asegurar que los paquetes de datos se entregan libres de error, ordenadas y sin pérdidas ni duplicados. Puede también optimizar el uso de los servicios de red (por ejemplo mediante multiplexación) y proporcionar la calidad de servicio solicitada por los servicios de sesión.

El tamaño y la complejidad de un protocolo de transporte dependen del tipo de servicio proporcionado por el nivel de red. Con un servicio de red fiable, orientado a la conexión, un protocolo de transporte mínimo resultará suficiente. Por el contrario, si el nivel 3 proporciona un servicio no fiable y/o basado en datagramas el protocolo de transporte debe incluir detección y recuperación de errores. De acuerdo con esto, **ISO** ha definido cinco clases de protocolo de transporte orientados a distintos tipos de servicios de red. Otro estándar de protocolo de transporte ampliamente extendido – aunque fuera del entorno **ISO** – es el protocolo **TCP**.

- **Nivel de sesión:** Este nivel proporciona los mecanismos para controlar el diálogo entre aplicaciones. Como mínimo el nivel de sesión proporciona un medio para que dos procesos de aplicación puedan establecer y utilizar una conexión, llamada **sesión**. Además de esto, puede proporcionar una serie de servicios de mejora sobre el nivel de transporte, como son:

    1. Gestión del diálogo, mediante la utilización de testigos.

    2. Mecanismos de recuperación (*checkpointing).*

- **Nivel de Presentación:** A diferencia de los niveles anteriores, interesados en la fiabilidad de los datos que se transmiten, el nivel de presentación se ocupa de aspectos sintácticos y semánticos de la información transmitida.

Un ejemplo típico de un servicio de presentación es la codificación de datos de una forma estándar. Cada ordenador tiene su propia forma de representar strings de caracteres (ASCII, EBCDIC, ...), enteros (complemento a uno, dos, ...) números en coma flotante y estructuras compuestas. Para hacer posible la comunicación entre computadoras con distintos esquemas de representación. Las estructuras de datos pueden definirse durante la comunicación. El trabajo de gestionar estas estructuras de datos y convertirlas de la representación usada en el ordenador a la representación estándar y viceversa, es realizado por el nivel de presentación. Además de lo anterior, este nivel

maneja otros aspectos de representación de la información como compresión de datos y cifrado.

- **Nivel de aplicación:** El nivel de aplicación proporciona un medio a los procesos de aplicación para acceder al entorno **OSI**. Contiene funciones de gestión y mecanismos útiles para soportar aplicaciones distribuidas. Ejemplos de protocolos a este nivel son los de transferencia de ficheros y correo electrónico.

2.2.2. Servicios

Las entidades en un nivel *N* ofrecen servicios que son utilizados por las entidades del nivel *N+1*. El nivel *N* es, entonces, el **proveedor del servicio** y el nivel *N+1* el **usuario del servicio**. A su vez, el nivel *N* para proporcionar sus servicios puede utilizar los servicios que le ofrece el nivel *N-1*.

Los servicios se hacen disponibles en los **SAP** (Puntos de acceso al servicio). Los **SAPs** del Nivel *N* son los puntos donde el nivel *N+1* puede acceder a los servicios ofrecidos.

Un servicio es invocado por el usuario, o es indicado por el proveedor del servicio mediante el intercambio de un conjunto de **primitivas de servicio** a través de la interfaz entre los niveles implicados. En el modelo OSI, estas primitivas se dividen en cuatro clases:

- ***Request:*** Una entidad solicita el servicio.
- ***Indication:*** Una entidad es informada de algún evento.
- ***Response:*** Una entidad quiere responder a un evento.
- ***Confirm:*** Una entidad es informada sobre su solicitud.

Los servicios pueden ser confirmados o no confirmados. Un servicio confirmado utiliza las cuatro primitivas request, indication, response y confirm. Un servicio no confirmado sólo requiere primitivas request e indication. El establecimiento de la conexión siempre es un servicio confirmado, mientras que la transferencia de datos puede ser confirmada o no, dependiendo de que el emisor necesite o no un reconocimiento.

Los niveles pueden ofrecer dos tipos básicos de servicio: sin conexión y orientados a conexión. La *conexión* permitirá establecer unos parámetros generales para toda la comunicación, agrupando los diferentes mensajes en un marco común.

En los servicios sin conexión la información recibida por cada nivel es procesada de forma autónoma, independientemente de la que haya recibido anteriormente. Es un servicio similar al ofrecido por correos, en el cual cada carta viaja de forma independiente de las anteriores.

Se pueden distinguir dos modelos de servicios sin conexión:

- **Datagrama:** Consiste en enviar la información y despreocuparse de ella. Por ello se le suele denominar *Send & Pray (Sueña y Reza)*. Este servicio sería equivalente al correo ordinario, en el cual enviamos una carta y no obtenemos confirmación de su llegada.

- **Con acuse de recibo (Asentimiento):** El receptor tiene que enviar un reconocimiento de que ha recibido la información.

Los servicios orientados a conexión corresponden al modelo del sistema telefónico. Cada mensaje enviado es interpretado en un contexto formado por los mensajes anteriores y posteriores, de forma que forman una unidad. Para ellos es necesario que se cumplan tres fases:

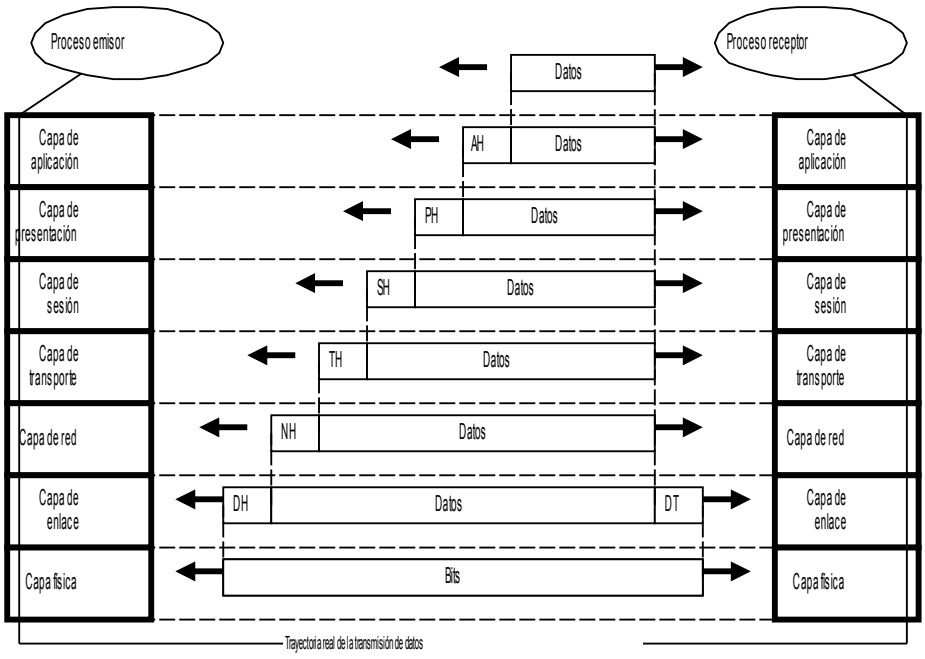
- Establecimiento de la conexión.

- Utilización.
- Desconexión, o cierre de la conexión.

2.2.3. Transmisión de datos en el modelo OSI

En el esquema OSI se pretende implementar la comunicación de aplicaciones de usuario mediante la utilización de servicios proporcionados por los niveles inferiores. Ambas aplicaciones tendrán una unidad de información básica a intercambiar, cumpliendo su protocolo establecido de nivel de aplicación. Debemos conseguir que esta información transmitida llegue tal y como fue enviada al nivel de aplicación del host receptor.

Sin embargo, y para asegurar el cumplimiento de sus funciones, en cada nivel es necesario utilizar cierta información de control que sólo será interpretada por el nivel equivalente de la máquina receptora. Por ejemplo, para que el nivel de red de los distintos IMPs por los que pasará la información puedan enviar correctamente la misma, es necesario conocer las direcciones en la red de las máquinas origen y destino de los datos, pero esta información no tiene por qué ser conocida por el nivel de transporte ni por el de enlace de datos. De hecho, y para proteger la independencia de niveles, resulta aconsejable que cada información de control sea exclusiva del nivel que la requiera. Cada nivel, pues, tratará la información procedente del nivel superior como si fueran datos en su integridad, y añadirá su propia información de control (cabecera) antes de pasarlo al nivel inferior.



Transmisión de datos en el modelo OSI

Como puede verse, cada nivel añade información a transmitir para cumplir su protocolo, tratando la información de control añadida por el nivel anterior como datos. Los términos AH, PH, etc, denotan las cabeceras añadidas por cada uno de los niveles (Aplication Header, Presentation Header, etc).

A pesar de que la comunicación se realiza verticalmente (entre niveles) hay que tener en cuenta la existencia de los protocolos a cada nivel.

Nivel Físico

El nivel físico, o primer nivel del modelo ISO/OSI, se preocupa de todos los problemas relacionados con la transmisión de la señal que lleva la información, Este puede ser enviada a través de los medios de transmisión modificando alguna propiedad física del mismo. Por ejemplo, sobre una línea eléctrica podemos enviar datos modificando el voltaje o la intensidad que circula sobre la misma. Representando el valor de este voltaje o intensidad como una función del tiempo *f(t)*, podemos modelar las características de la señal y aplicarle un análisis matemático. En este tema nos preocuparemos de este análisis. En concreto estudiaremos la naturaleza de la señal, así como sus propiedades y comportamientos sobre el medio de transmisión.

Podemos abordar el estudio de una *señal* desde dos puntos de vista distintos: desde el dominio del tiempo y desde el dominio de la frecuencia:

3.1.1. Conceptos en el dominio del tiempo

Desde este punto de vista podemos plantearnos si la función *f(t)* es:

- **Continua:** Cuando $\forall a \; Lim_{t \to a} f(t) = f(a)$
- **Discontinua:** Cuando existen discontinuidades o saltos en la función, es decir, no se verifica la expresión anterior.
- **Discreta:** La función toma un conjunto finito de valores. Un ejemplo de esto es una señal digital.
- **Analógica:** La función puede tomar un conjunto infinito de valores.

Asimismo, podemos plantearnos si la señal es **periódica**, es decir si la función toma el mismo valor cada un cierto tiempo *T*, al que denominaremos *periodo*. Podemos decir que una señal periódica es aquella que cumple que:

$$f(a) = f(a + T) \quad \forall a$$

Un ejemplo de función periódica será la siguiente:

$$f(a) = A \times cos(\omega t + \phi)$$

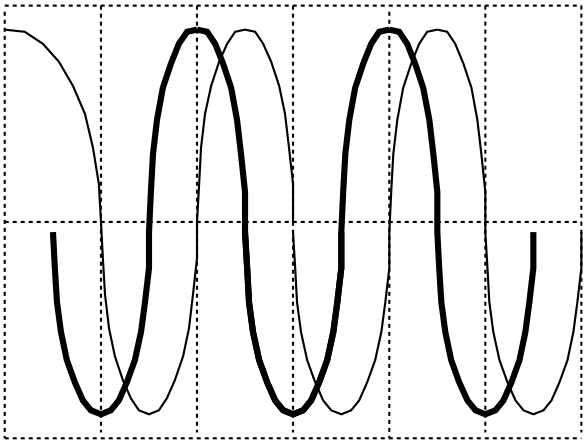
De una función periódica podemos distinguir los siguientes parámetros:

- **Amplitud:** Máximo valor que puede adoptar la señal periódica. En el ejemplo, coincide con *A*.

■ **Frecuencia:** Número de ciclos por segundo o hertzios. Se calcula como la inversa del periodo. Se representa por *f*.

■ **Pulsación:** Variable derivada de la frecuencia. Se calcula como $\omega = 2\pi f$ y se mide en radianes por segundo.

■ **Fase:** Diferencia en el valor de paso por cero de la función. Sirve para distinguir señales que aunque tienen la misma frecuencia y amplitud no son iguales. Esta diferencia se refleja en la siguiente gráfica.

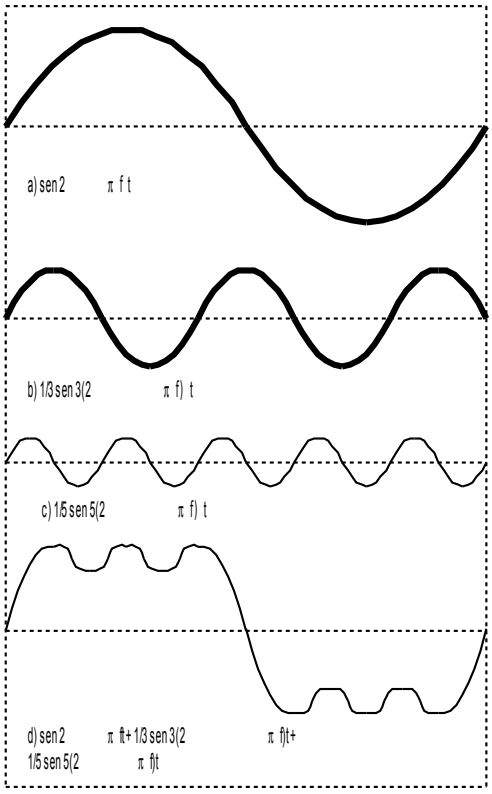


3.1.2. Conceptos en el dominio de la frecuencia

La señal que se transmite suele representarse como una función del tiempo, pero también puede expresarse en función de la frecuencia. Generalmente está constituida por varias componentes frecuenciales, lo que hace su análisis menos intuitivo. A efectos de transmisión de datos suele resultar más útil el análisis frecuencial de la señal que el temporal. Por ejemplo, la señal *s(t)* .

$$s(t) = \sin 2\pi ft + 1/2 \sin 2(2\pi ft) + 1/3 \sin 3(2\pi ft)$$

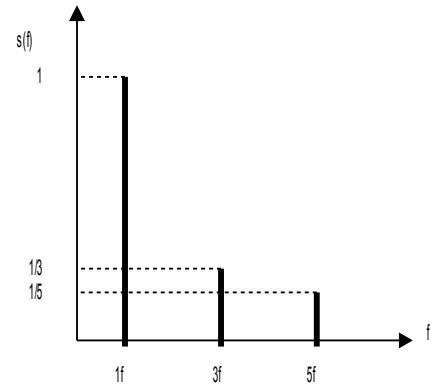
presenta tres componentes sinusoidales de frecuencias *f*, *2f*, *3f*, que pueden verse en la siguiente figura.



Puede demostrarse (por medio del análisis de Fourier), que cualquier señal periódica puede descomponerse en una o más componentes, siendo cada componente una sinusoide.

Podemos afirmar que, para cada señal existe una función *s(t)* en el dominio del tiempo que especifica la amplitud de la señal en cada instante, y de forma análoga existe una función *S(f)* en el dominio de la frecuencia que especifica las frecuencias que constituyen la señal. La siguiente figura muestra la representación en el dominio de la frecuencia de la señal de ejemplo. Nótese que , en este caso *S(f)* es discreta, situación que se dará siempre que representemos señales periódicas.

El **espectro** de una señal es el rango de frecuencias que contiene. En la figura, el espectro se extiende desde *f* hasta *5f*. El **ancho de banda absoluto** es la anchura del espectro, que el caso anterior es de *4f*. Muchas señales poseen un ancho de bando absoluto infinito, lo que en principio dificultaría su transmisión, ya que los medios de transmisión de comportan como filtros, dejando pasar únicamente una banda de frecuencias y eliminando las restantes lo que da lugar a que se modifique la forma de onda de la señal. Sin embargo, la mayor parte de la energía de la señal suele concentrarse en una pocas frecuencias que se conocen cono **ancho de banda efectivo** de la señal, o simplemente como **ancho de banda**. Esto se traduce en que eliminar las componentes frecuenciales que quedan fuera del ancho de banda efectivo no tengo demasiada importancia y la información que contiene la señal pueda todavía ser correctamente interpretada en el receptor.



Representación en el domino de la frecuencia

3.1.3. Señales analógicas y señales digitales

Una **señal analógica** representa un onda electromagnética que varía de forma continua. Dependiendo de su espectro, las señales analógicas pueden transmitirse por una amplia variedad de medios, por ejemplo, cables como el coaxial, la fibra óptica y medios de propagación espacial o atmosférica.

Una **señal digital** es una secuencia de pulsos de voltaje que pueden transmitirse por medio de un cable; por ejemplo, un nivel de voltaje positivo constante puede representar el uno binario y un nivel de voltaje negativo puede representar el cero binario.

3.1.4. Análisis de Fourier

Hemos visto como podemos definir una señal, tanto en el dominio del tiempo como en el de la frecuencias, y ahora con el *análisis de Fourier* vamos a tratar de estudiarlo matemáticamente.

Cualquier función periódica *s(t)* de periodo *T*, puede expresarse como una suma (posiblemente infinita) de senos y cosenos:

$$s(t) = a_0 + \sum_{n=1}^{\infty} a_n \operatorname{sen}(2\pi f n t) + \sum_{n=1}^{\infty} b_n \cos(2\pi f n t)$$

Esta herramienta matemática recibe el nombre de **Serire de Fourier**. $f = 1/T$ es la **frecuencia fundamental** y los términos a_n y b_n las amplitudes de los *n*-ésimos **armónicos**. La componente con $n=1$ es la **componente fundamental**. Obsérvese que el valor a_0 representa la componente continua de la señal.

La transmisión de las señales provoca una disminución de su potencia. Si todas las componentes de la señal se vieran afectadas de igual forma, la amplitud de la señal disminuiría pero su forma no se vería distorsionada. Desafortunadamente, esto no sucede. Usualmente, las amplitudes apenas sufren variación desde la de frecuencia cero hasta una cierta frecuencia de corte f_c, a partir de la que se atenúan fuertemente. En algunos casos, esto se debe al propio medio de transmisión y en otros se logra intencionadamente por medio de filtros que limitan el ancho de banda asignado a cada usuario.

Es importante tener en cuenta que al aumentar la velocidad de transmisión de los datos, se aumenta la frecuencia fundamental asociada a la señal, y por tanto, el número de armónicos que consiguen pasar disminuye. Esto establece un límite superior en la velocidad de transmisión que puede utilizarse.

Las series de Fourier constituyen una importante herramienta en la comunicación porque permiten calcular el ancho de banda de las señales periódicas. Simplifican el análisis de complicadas formas de onda mediante su descomposición en señales más elementales (sinusoides). Además, en la mayoría de las señales manejadas usualmente los coeficientes de Fourier (a_n y b_n) decaen rápidamente al aumentar n, lo que hace posible obtener una buena aproximación de la señal manejando únicamente unas pocos términos.

La transmisión de una señal supone el paso de la misma a través de una determinado medio, por ejemplo: un cable, el aire, etc. Debido a diferentes fenómenos físicos, la señal que llega al receptor difiere de la emitida por el transmisor. Vamos a estudiar a continuación una serie de efectos que contribuyen a modificar la señal que se transmite.

S la suma de todos los efectos no produce una gran diferencia entre ambas señales, conseguiremos una transmisión libre de errores. Por el contrario, cuando la señal recibida difiera en exceso de la señal transmitida el receptor puede interpretar incorrectamente la información y decimos entonces que se produce un **error de transmisión**. Evidentemente no todas las señales sufren los mismos efectos al atravesar los distintos medios de transmisión, luego cuando sea posible, escogeremos el tipo de señales y medios que conduzcan a las mejores condiciones de transmisión.

Veamos ahora algunos de estos problemas de la transmisión.

La atenuación

Consiste en el debilitamiento o pérdida de amplitud de la señal recibida frente a la transmitida. Por ejemplo, sabemos que cualquier sonido se percibe con menor intensidad cuando más alejados nos encontramos de la fuente que lo origina. Efectivamente, la atenuación tiene un efecto proporcional a la distancia. A partir de una determinada distancia, la señal recibida es tan débil que no se puede reconocer mensaje alguno.

Para paliar el efecto de la atenuación se pueden incorporar en el camino de la señal unos dispositivos activos, cuya función es amplificar la señal en la misma medida en que acaba de ser atenuada por el medio, de esta forma se consigue recuperar la señal para que pueda alcanzar más distancia.

Según el tipo de señal, analógica o digital, estos dispositivos tienen un comportamiento distinto y también diferente nombre. Para el caso de **señales digitales** hablamos de dispositivos **repetidores**, que son capaces de restaurar la misma señal original. Para las **señales analógicas** se denominan **amplificadores** y estos elementos no permiten recuperar la señal original, debido al efecto del ruido que no se puede aislar de las señales analógicas pero sí de las digitales.

Debido a la imposibilidad de supresión del ruido en el caso de las señales analógicas aparece la limitación del número máximo de amplificadores que pueden ser conectados en una línea de transmisión y con ello se limita la distancia máxima de este tipo de transmisiones.

Distorsión por atenuación

Hasta ahora hemos supuesto que la atenuación afecta por igual a todas las señales.. sin embargo, la atenuación es función, además de la distancia, de la frecuencia de las señales que se propagan. Las de mayores frecuencias sufren una mayor atenuación.

Este fenómeno produce, en las señales con diferentes componentes frecuenciales, una atenuación distinta para cada componente de frecuencia, lo que origina que la señal recibida

tenga una forma diferente de la transmitida, amen de una menor amplitud. Como la señal recibida se ha *deformado* con respecto a la transmitida decimos que se ha *distorsionado*.

Para compensar esta diferente atenuación a distintas frecuencias, los amplificadores pueden incorporar una etapa denominada *ecualizador*.

El retardo de grupo

Otro de los problemas de la transmisión es el **retardo**. Sabemos que todas las señales se propagan a una cierta velocidad, que depende del medio y de la naturaleza de la señal. Por ejemplo: el sonido se propaga en el aire aproximadamente a 340 m/s, la luz a 3000.000 km/s, etc.

Luego todas las señales van a tardar un cierto tiempo en recorrer la distancia que separa al emisor del receptor. Además, si en el camino la señal atraviesa determinados circuitos electrónicos, ópticos, o de cualquier otra naturaleza, estos pueden añadir un retardo adicional. Por ejemplo: una puerta lógica introduce un retardo del orden de 15ns entre su entrada y su salida.

De igual forma que sucedía con la atenuación, el retardo tampoco es una función constante con la frecuencia y las diferentes componentes de una señal sufren distintos retardos. Por ejemplo: para una señal limitada en ancho de banda la velocidad tiende a ser más alta en la frecuencia central y decrece en los límites de la banda de frecuencias. Esto trae como consecuencia que en un instante dado las componentes frecuenciales que llegan al receptor no son las mismas que unos instantes antes envió el emisor, por lo tanto, la señal recibida tendrá una forma distinta de la emitida, de nuevo hablamos de distorsión. A la distorsión producida por el retardo, se la denomina **distorsión por retardo**.

Este fenómeno carece de trascendencia en las transmisiones de voz, ya que el oído humano no es sensible a las diferencias de retardo. Sin embargo, tiene efectos importantes en la transmisión de datos digitales, especialmente a alta velocidad.

La diafonía

La **diafonía** (*crosstalk)* Es un fenómeno que todos hemos experimentado en las comunicaciones telefónicas. Consiste en la interferencia de un canal (o cable) próximo con el nuestro, esto produce una señal que es la suma de la señal transmitida y otra señal externa atenuada que aparece *de fondo*. En una conversación telefónica esto se observa como una segunda conversación que se oye de fondo mezclada con la nuestra.

El motivo de este fenómeno es la influencia mutua entre dos canales de transmisión próximos en frecuencia o que comparten el mismo tendido de cables.

El ruido impulsivo

Otra fuente de problemas en la transmisión es el denominado **ruido impulsivo**. Consiste en pulsos irregulares de corta duración y relativamente gran amplitud, que son provocados por inducciones, como consecuencia de conmutaciones electromagnéticas. Este tipo de ruido es debido a causas variadas externas al medio de transmisión. Podemos asociarlo a las interferencias en un receptor de radio cuando se aproxima una motocicleta, o también al encendido de determinados aparatos en un domicilio (por ejemplo: una lavadora o nevera).

Existen infinidad de dispositivos cuyo encendido o apagado genera un impulso de radio frecuencia capaz de influir a canales de comunicación próximos. El ruido impulsivo es típicamente aleatorio, es decir, se produce de manera inesperada y no suele ser repetitivo.

El ruido térmico

Está presente en todos los dispositivos electrónicos y medios de transmisión y es debido a

la agitación de los electrones en un conductor. Es proporcional a la temperatura y se encuentra distribuido uniformemente en todo el espectro de frecuencias. Habitualmente el efecto del ruido térmico es despreciable, excepto en aquellos casos en los que se trabaja con señales muy débiles.

Como ya hemos visto, diferentes circunstancias producen ruido en la transmiisón de las señales. Si la amplitud del ruido es mucho menor que la de la señal transmitida el receptor puede interpretar la información sin errores, pero si el nivel de ruido aumenta la señal recibida resultará ininteligible, o al menos se producirá un número importante de errores.

Para poder determinar cuantitativamente esta importancia del nivel de ruido en un medio transmisión se podría manejar el cociente entre el nivel medio de las señales y el ruido. Sin embargo, esta medida tendría un rango de variación muy elevado debido a las grandes diferencias que podemos encontrar entre unos medios y otros. Resultará más conveniente emplear unidades que no supongan el empleo de grandes magnitudes.

Además, la potencia de la señal que se transmite disminuye de forma logarítmica, lo que hace que las pérdidas puedan ser expresadas fácilmente en términos de una unidad logarítmica. Por estas razones, la unidad empleada para expresar relaciones de potencia entre dos señales es el **decibelio**, que se calcula según la siguiente expresión:

$$(S/N)_{db} = 10log_{10}(potencia\_señal/potencia\_ruido)$$

Así, por ejemplo, una relación señal ruido de 30db, es una relación 100:1. Es decir, la potencia de la señal es mil veces superior a la del ruido.

3.4.1. Capacidad de un canal ideal

En lo sucesivo, mientras no se indique lo contrario, se asumen sistemas de comunicación digitales.

Hasta ahora hemos visto una serie de efectos físicos que producen problemas en la transmisión. Aparte de ellos, existen unos límites máximos que determinan la velocidad de transmisión máxima de un canal, dependiendo del ancho de banda del canal y el tipo de señal digital empleada.

Por canal ideal entenderemos un canal exento de ruidos y de distorsiones. En este medio ideal, le velocidad máxima de transmisión de datos viene limitada por la fórmula:

$$C = 2Wlog_2 M \quad bps$$

Siendo *M* el número de niveles posibles de la señal y W el ancho de banda expresado en hercios. El valor de *M* para una señal digital binaria es dos, pues dos son los valores que toma la señal.

El interés de esta expresión radica en que fija una cota superior con la que compara el comportamiento de los sistemas que se diseñan.

3.4.2. Capacidad de un canal con ruido

Ya que los canales ideales no existen, sería interesante conocer la capacidad de un canal real, en el que va a aparecer una cierta cantidad de ruido. La siguiente expresión – conocida como fórmula de Shannon- nos proporciona la capacidad máxima de un canal con ruido:

$$C = 2W\log_2 (1 + S/N) \quad bps$$

Vemos pues que la capacidad de los canales con poco ruido será mayor que la de aquéllos con mucho ruido.

Esta capacidad máxima es inalcanzable, ya que la fórmula de Shannon supone unas condiciones que en la práctica no se dan. No tiene en cuenta el ruido impulsivo, ni la atenuación ni la distorsión. Representa el límite teórico máximo alcanzable.

El medio de transmisión constituye el soporte físico a través del cual emisor y receptor pueden comunicarse en un sistema de transmisión de datos. Distinguimos dos tipos de medios: **guiados** y **no guiados**. En ambos casos la transmisión se realiza por medio de ondas electromagnéticas. Los medios guiados conducen (guían) las ondas a través de un camino físico, ejemplos de estos medios son el cable coaxial, la fibra óptica y el par trenzado. Los medios no guiados proporcionan un soporte para que las ondas se transmitan, pero no las dirigen; como ejemplo de ellos tenemos el aire y el vacío.

La naturaleza del medio junto con la de la señal que se transmite a través de él constituyen los factores determinantes de las características y la calidad de la transmisión. En el caso de medios guiados es el propio medio el que determina el que determina principalmente las limitaciones de la transmisión: velocidad de transmisión de los datos, ancho de banda que puede soportar y espaciado entre repetidores. Sin embargo, al utilizar medios no guiados resulta más determinante en la transmisión el espectro de frecuencia de la señal producida por la antena que el propio medio de transmisión. En la siguiente tabla se muestran las características más típicas de algunos medios guiados.

Medio	V. de transmisión	Ancho de banda	Distancia entre repetidores
Par trenzado	4 Mbps	250 Khz	2 – 10 km
Cable coaxial	550 Mbps	350 Mhz	10 –100 km
Fibra óptica	2 Gbps	2 Ghz	

3.5.1. Medios guiados más usuales

Medio magnético

Una de las formas más comunes para el transporte de datos, de un ordenador a otro, consiste en escribir dicha información sobre una cinta magnética o en discos, y transportar físicamente la cinta o los discos hasta la máquina destino, para que después ésta pueda leer la información. Este método, aunque no tan sofisticado como aquellos en los que se utilizan satélites de comunicación geosíncronos, es bastante efectivo en coste, en especial en los casos en los que se necesitan anchos de banda grandes o en donde el coste por bit transportado representa un factor clave.

Líneas de hilo desnudo

Es el medio más simple de transmisión. Prácticamente ha caído ya en desuso. La información se transmite por medio de cables conductores sin recubrimiento aislante que deben por tanto, ir separados. La señal, que es típicamente un voltaje o nivel relativo de corriente respecto a una referencia, se aplica a uno de los hilos mientras que el otro se conecta a tierra.

No constituye un buen medio porque tiene una elevada atenuación y además es muy sensible a los ruidos eléctricos (diafonía) causados por el acoplamiento capacitivo entre dos hilos.

Pares trenzados

Este consiste en dos alambres de cobre aislados, en general de 1mm de espesor. Los alambres se entrelazan en forma helicoidal, como en una molécula de DNA. La forma trenzada del cable se utiliza para reducir la interferencia eléctrica con respecto a los pares cercanos que se encuentran a su alrededor, (Dos cables paralelos constituyen una antena simple, en tanto que un par trenzado no.)

La aplicación más común del par trenzado es el sistema telefónico, casi todos los teléfonos están conectados a la oficina de la compañía telefónica a través de un par trenzado. La distancia que se puede recorrer con estos cables es de varios kilómetros, sin necesidad de amplificar las señales, pero si es necesario incluir repetidores en distancias más largas. Cuando hay muchos pares trenzados colocados paralelamente que recorren distancias considerables, como podría ser el caso de los cables de un edificio de departamentos que se dirigen a la oficina de teléfonos, éstos se agrupan y se cubren con una malla protectora. Los pares dentro de estos agrupamientos podrían sufrir interferencias mutuas si no estuviera trenzados. En algunos lugares del mundo en donde las líneas telefónicas se instalan en la parte alta de los postes, se observan frecuentemente dichos agrupamientos, como cables con diámetros de varios centímetros.

Los pares trenzados se pueden utilizar tanto para transmisión analógica como digital, y su ancho de banda depende del calibre del alambre y de la distancia que recorre; en muchos casos pueden obtenerse transmisiones de varios megabits/s, en distancias de pocos kilómetros. Debido a su adecuado comportamiento y bajo costo, los pares trenzados se utilizan ampliamente y es probable que se presencia permanezca por muchos años.

El principal factor limitador de las línea de par trenzado es causado por un fenómeno conocido como *efecto piel*. Al aumentar la velocidad de transmisión y por tanto la frecuencia de la señal transmitida, la corriente tiende a fluir únicamente por la superficie del cable. Esto tiene el efecto de aumentar la resistencia de los hilos para las señales de alta frecuencia, lo que provoca una mayor atenuación de las señales transmitidas. A altas frecuencias una cantidad creciente de la potencia de la señal se pierde debido a los efectos de la radiación. Por eso, tradicionalmente para aquellas aplicaciones que requieren una velocidad superior a 1Mbps solían emplearse otros medios de transmisión. Una línea de transmisión que minimiza ambos efectos es el cable coaxial.

Cable coaxial de banda base

El cable coaxial es otro medio típico de transmisión. Hay dos tipos de cable coaxial que se utilizan con frecuencia, uno de ellos es el cable de 50 ohms, que se utiliza en la transmisión digital y es precisamente el tema de esta sección; en tanto que el otro tipo, el cable de 75ohms, que se emplea en la transmisión analógica, será el tema de la sección siguiente.

El cable coaxial consta de un alambre de cobre duro en su parte central, es decir, que constituye el núcleo, el cual se encuentra rodeado por un material aislante. Este material aislante está rodeado por un conductor cilíndrico que frecuentemente se presenta como una malla de tejido trenzado. El conductor externo está cubierto por una capa de plástico

protector.

La construcción del cable coaxial produce una buena combinación e un gran ancho de banda y una excelente inmunidad al ruido. El ancho de banda que se puede obtener depende de la longitud del cable; para cables de 1km, por ejemplo, es factible obtener velocidades de datos de hasta 10Mbps, y en cables de longitudes menores, es posible obtener velocidades superiores. Se pueden utilizar cables con mayor longitud, pero se obtienen velocidades muy bajas. Los cables coaxiales se emplean ampliamente en redes de área local y para transmisiones de largas distancia del sistema telefónico.

Cable coaxial de banda ancha

El sistema que considera el otro tipo de cable coaxial emplea la transmisión analógica en el cableado que se utiliza comúnmente para el envío de la señal de televisión por cable, y se le denomina de **banda ancha**. Aunque el término *banda ancha* proviene del medio telefónico, en el cual se refiere a frecuencias superiores a los 4kHz, el significado de este término en el medio de redes de ordenadores se asocia a las redes de cables utilizadas para la transmisión analógica. Dado que las redes de banda ancha utilizan la tecnología patrón para envío de señales de televisión por cable, los cables pueden emplearse para aplicaciones que necesiten hasta los 300 Mhz (y en algunos casos hasta los 450 Mhz), y extenderse a longitudes que alcanzan hasta los 100 Km, gracias a la naturaleza analógica de la señal, que es menos crítica que la del tipo digital. Para transmitir señales digitales en una red analógica, cada interfaz debe tener un dispositivo electrónico que convierta en señal analógica el flujo de bits de envío, y otro para convertir la señal analógica que llega en un flujo de bits. Dependiendo del tipo (y precio) de estos dispositivos electrónicos, 1 bps puede llegar a ocupar un ancho de banda que va desde 1 a 4 Hz. Un cable típico de 300Mhz, por lo general puede mantener velocidades de transmisión de datos de hasta 150 Mbps.

Fibras ópticas

Una fibra óptica es un cilindro de pequeña sección (diámetro del orden de 2 a 125 μm), de un medio flexible –cristal, plástico- capaz de conducir un rayo óptico. Las mejores son las de silicio puro, y que tienen menos pérdidas, aunque su precio es más elevado..

Un cable de fibra óptica consta de tres secciones concéntricas. La más interna, el núcleo, consiste en una o más hebras o fibras hechas de cristal o plástico. Cada una de ellas lleva un revestimiento de cristal o plástico con propiedades ópticas distintas a las del núcleo. La capa más exterior, que recubre una o más fibras, debe ser de un material opaco y resistente.

Un sistema de transmisión por fibra óptica está formado por una fuente luminosa muy monocromática (generalmente un láser), la fibra encargada de transmitir la señal luminosa y un fotodiodo que reconstruye la señal eléctrica.

La luz se propaga en zig-zag debido a los fenómenos de reflexión total que tienen lugar en el interior de la fibra. Por este motivo las pérdidas son muy escasas. Además, las fibras son inmunes a las interferencias electromagnéticas y a su vez no interfieren en otros sistemas. Por lo tanto, resultan extremadamente útiles para la transmisión de señales en medios muy ruidosos. Entre otras de sus ventajas podemos citar su elevado ancho de banda (permite alcanzar velocidades del orden de Gbps sobre decenas de Km) y sus reducidos peso y tamaño.

Los sistemas de fibra óptica suelen resultar más caros que los de cable coaxial y mecánicamente más delicados, por lo que presentan más dificultades en su instalación. Se utilizan en telecomunicaciones a larga distancia, aplicaciones militares, redes locales, distribución de señales de audio/vídeo.

3.5.2. Medios no guiados

En muchas ocasiones resulta problemática la instalación de un tendido. Los enlaces vía

radio emplean la propagación de las ondas electromagnéticas en el espacio y por lo tanto no precisan de ningún tipo de cableado entre emisor y receptor. Dentro de los enlaces vía radio existen diferentes tipos según la banda empleada, exhibiendo diferentes propiedades. Los métodos más usuales son:

Radio enlaces de onda corta

La OC es una banda de radio, comprendida entre 2 y 15 Mhz aproximadamente, (aparece con las siglas SW en los receptores de radio). Poseen un alcance de miles de kilómetros, ya que se reflejan en la ionosfera y además son omnidireccionales, aunque sólo permite reducidas velocidades de transmisión, menores de 1200 bps. Aunque antaño fueron el medio más común, su uso actualmente se encuentra restringido a circunstancias especiales, debido a su limitada capacidad. Se emplea, por ejemplo, para la difusión de noticias de las agencias de teletipos de todo el mundo.

Radio enlaces de VHF y UHF

Estas bandas cubren aproximadamente desde 55 a 550 Mhz. Son también omnidireccionales, pero a diferencia de las anteriores la ionosfa es transparente a ellas. Su alcance máximo es de un centenar de kilómetros, y las velocidades que permite del orden de los 9600 bps. Su aplicación suele estar relacionada con los radioaficionados y con equipos de comunicación militares, también la televisión y los aviones.

Microondas

Además de su aplicación en hornos, las microondas nos permiten transmisiones tanto terrestres como con satélites. Dad su frecuencias, del orden de 1 a 10 Ghz, las microondas son muy direccionales y sólo se pueden emplear en situaciones en que existe una línea visual que une emisor y receptor. Los enlaces de microondas permiten grandes velocidades de transmisión, del orden de 10 Mbps.

Como hemos visto, existen diferentes tipos de enlaces vía radio, y cada uno nos permite cubrir un rango de distancias a diferentes velocidades de transmisión. No se olvide que en el caso de los enlaces de microondas las distancias, *en tierra*, de un enlace suelen ser de unos 30 a 50 Km. máximo. Sin embargo, en el caso de la comunicación con un satélite, si bien las distancias pueden ser de hasta 36.000 Km., sólo durante una pequeña parte del recorrido la señal se atenúa por el efecto de la atmósfera y el resto del trayecto es prácticamente en el vacío, que no atenúa la señal.

Para transmitir datos es preciso transformarlos en una señal temporal que los represente y que pueda atravesar un determinado medio de transmisión para llegar al receptor en las mismas condiciones posibles.

En las redes de ordenadores, los datos a intercambiar siempre están disponibles en forma de señal digital. No obstante, para su transmisión podemos optar por la utilización de señales digitales o analógicas. La elección no será, casi nunca, una decisión del usuario, sino que vendrá determinada por el medio de transmisión a emplear.

No todos los medios de transmisión permiten señales analógicas ni todos permiten señales digitales. Como la naturaleza de nuestros datos será siempre digital, es necesario un proceso previo que adecue estos datos a la señal a transmitir.

3.6.1. Información digital, señal digital

Si el medio de transmisión permite el empleo de señales digitales y los datos a enviar son de

este tipo, resulta muy conveniente el empleo de señales digitales. Para obtener la secuencia que compone la señal digital a partir de los datos digitales se efectúa un proceso denominado *codificación.*

Existen multitud de métodos de codificación, mencionaremos seguidamente los más usuales. En todos ellos se menciona la **celda de bit**, que es la duración de un bit, o sea, la inversa de la velocidad de transmisión.

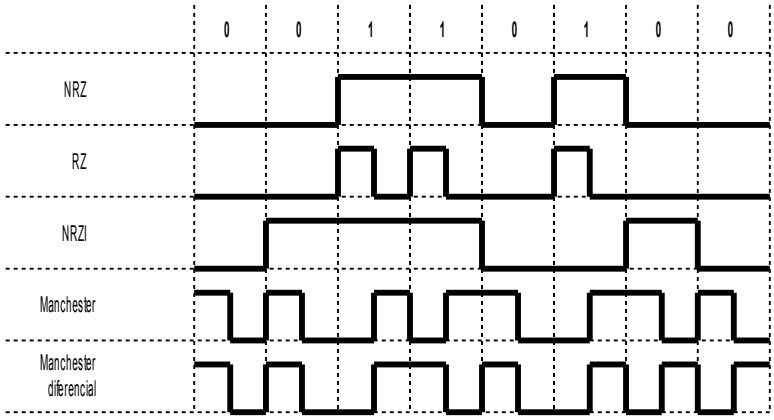
■ **NRZ (No Return to Zero)**: Es el método que empleamos para representar la evolución de una señal digital en un cronograma. Cada nivel lógico 0 y 1 toma un valor distinto de tensión.

■ **NRZI (No Return to Zero Inverted)**: La señal no cambia si se transmite un uno, y se invierte si se transmite un cero.

■ **RZ (Return to Zero)**: Si el bit es uno, la primera mitad de la celda estará a uno. La señal vale cero en cualquier otro caso.

■ **Manchester**: Los valores lógicos no se representan como niveles de la señal, sino como transiciones en mitad de la celda de bit. Un flanco de bajada representa un cero y un flanco de subida un uno.

■ **Manchester diferencial**: Manteniendo las transiciones realizadas en el método Manchester, en este método introduce la codificación diferencial. Al comienzo del intervalo de bit, la señal se invierte si se transmite un cero, y no cambia si se transmite un uno.



Diferentessistemas de codificación

Ya que existen muchos sistemas de codificación conviene destacar que no todos los métodos generarán señales con las mismas propiedades. A la hora de escoger entre unos y otros conviene considerar las siguientes características:

■ Espectro resultante: A cada esquema de codificación le corresponde un espectro diferente. Algunos esquemas no tienen componente continua en su espectro (Manchester) lo que es muy útil si deseamos acoplar inductivamente la señal.

■ Capacidad autoreloj: Algunas de las señales resultantes garantizan transiciones de nivel, con independencia del valor de los datos, que pueden servir para determinar el reloj de transmisión y la posición de la celda de bit. En otro caso, es necesario transmitir la señal de reloj por separado.

■ Capacidad de detección de errores: Algunos códigos permiten detectar condiciones de error. Por ejemplo: una transmisión que se debiera producir y no aparece.

3.6.2. Información digital, señal Analógica

Cuando las características del medio de transmisión no permiten señales digitales, se hace necesario convertir los datos digitales en una señal analógica susceptible de ser transmitida correctamente. El ejemplo más popular de esto, es la transmisión de datos digitales a través de las redes telefónicas. Estas redes fueron diseñadas para transmitir señales analógicas en el rango del espectro vocal (300-3400) Hz., lo que no resulta adecuado para la transmisión de señales digitales. Sin embargo, puede ligarse un dispositivo digital a la red telefónica por medio de un **modem**, que convierte los datos digitales en señales analógicas y viceversa.

Al proceso por el cual obtenemos una señal analógica a partir de unos datos digitales se le denomina **modulación**. Esta señal la transmitimos y el receptor debe realizar el proceso contrario, denominado **demodulación** para recuperar la información.

El proceso de modulación precisa de dos señales, **moduladora** y **portadora**, y genera como resultado una tercera señal, denominada **señal modulada**. Se define el término modulación como el proceso mediante el cual una señal que contiene información (moduladora) se combina con otra señal (portadora) para dar como resultado una nueva señal (modulada) que contiene la misma información que la primera pero que es el resultado de modificar alguno de los parámetros característicos (amplitud, frecuencia o fase) de la segunda.

Veamos ahora algunos esquemas simples de modulación:

- **FSK** (Modulación por desplazamiento de la frecuencia): Se modifica la frecuencia de la portadora según el valor de bit a transmitir, durante la celdas de bit con valor uno emplearemos la frecuencia f_1 y la f para los bits con valor cero. La señal así resultante será una sucesión de tramas de las dos frecuencias indicadas, y será una señal analógica.

Es el método más utilizado en modems de baja velocidad (300 a 1200 baudios) diseñados para operar con conexiones a través de la red telefónica conmutada. Presenta la ventaja de requerir circuitería simple para la demodulación y requisitos de ancho de banda bajos.

- **ASK** (modulación por desplazamiento de la amplitud): En esta técnica no se modifica la frecuencia de la portadora sino su amplitud. Los dos valores binarios se representan mediante diferentes niveles de amplitud de esta señal.

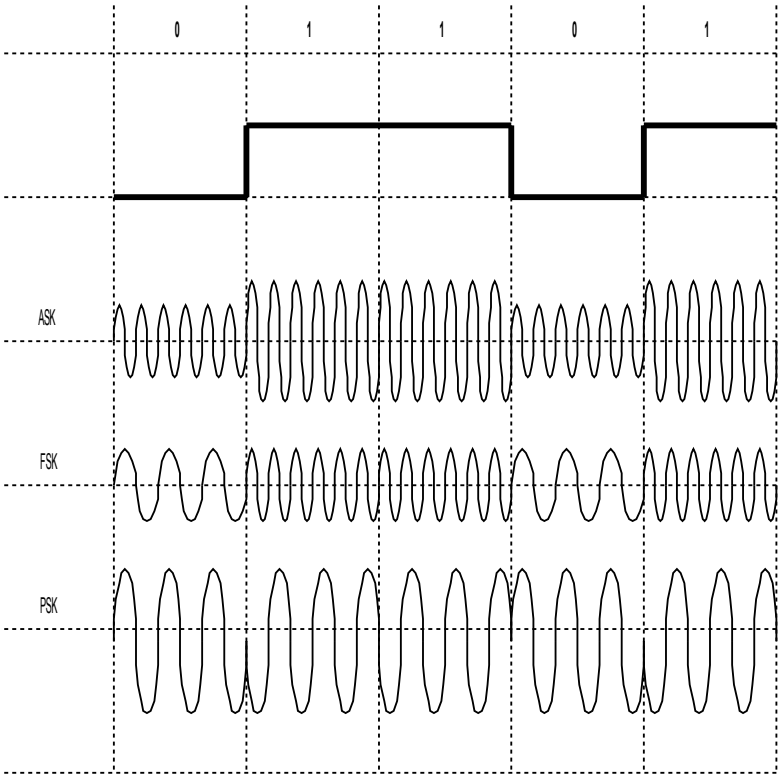
Generalmente una de las amplitudes es cero, o sea el uno binario se representa por la presencia de portadora y el cero por la ausencia de la misma. Es una técnica bastante ineficiente de modulación debido a su mayor sensibilidad al ruido (que modifica al amplitud), y sobre líneas de voz suele utilizarse sólo hasta 1200 bps. Sin embargo, resulta adecuada para la transmisión de datos digitales sobre fibra óptica. En este caso, los bits de valor uno se representan mediante pulsos de luz y los bits a cero por la ausencia de luz.

- **PSK** (Modulación por desplazamiento de fase): La frecuencia y la amplitud se mantiene constantes y se varía la fase de la portadora para representar los niveles uno y cero con distintos ángulos de fase.

Con este tipo de modulación el receptor debe mantener una señal portadora de referencia con la que comparar la fase de la señal recibida. Esto implica circuitos de demodulación complejos. Por este motivo, suele emplearse una forma alternativa de modulación en fase, que emplea desplazamientos en la fase relativos a la señal anterior transmitido. Por ejemplo, un desplazamiento de 90º relativos a la señal actual pueden servir para indicar un cero binario y un cambio den la fase de 270º un uno binario. De esta forma el circuito de demodulación solo necesita determinar los desplazamientos en la fase y no el valor absoluto (**PSK diferencial**).

En la forma más general de modulación PSK, la portadora se desplaza de forma sistemática 45º,

135º, 225º o 335º, en intervalos espaciados de manera uniforme, y por cada uno de estos desplazamientos de fase se transmiten dos bits de datos. En modems más sofisticados suelen emplearse varias combinaciones de amplitud y fase, con ello puede conseguirse, por ejemplo, que un elemento de señal represente cuatro bits, con lo que una transmisión a 2400 baudios permite un velocidad de transmisión de 9600 bps.



Diferentes métodos de modulación

En el apartado 3.2 vimos que existen diversos problemas en la transmisión de señales que pueden ocasionar, errores en la transmisión, especialmente cuando la distancia que separa emisor y receptor es grande. Pueden tomarse medidas para reducirlos, pero no para eliminarlos completamente. Por ello, al realizar una transmisión deben utilizarse técnicas que permitan detectar y corregir los errores que se hayan producido. Estas técnicas se basan siempre en la idea de añadir cierta información redundante a la información que desee enviarse. A partir de ella el receptor puede determinar, de forma bastante fiable, si los bits recibidos corresponden realmente a los enviados.

El problema puede abordarse de dos formas distintas:

- Mediante **códigos autocorrectores** (control de errores *forward*). En este caso, cada carácter o trama transmitida contiene suficiente información adicional, no sólo para que el receptor pueda detectar que ha ocurrido un error, sino para que en caso de que ello haya ocurrido pueda inferir la información correcta a partir de los datos recibidos. Este esquema se usa raramente en transmisión de datos, puesto que la segunda aproximación al problema resulta mucho más eficiente, ya que el número de bits adicionales necesarios en este tipo de control se incrementa muy rápidamente al aumentar el tamaño de los datos a enviar. Se usan únicamente

en casos donde la estrategia de retransmisión es impracticable. Situaciones de difusión y canales símplex (en un solo sentido de transmisión).

■ Utilizando **estrategias basadas en retransmisión** (control de errores *backward*). Cada carácter o trama transmitida contiene suficiente información adicional únicamente para permitir que el receptor detecte si ha ocurrido algún error, en cuyo caso el transmisor debe enviar una copia del carácter o trama dañado. En nuestro estudio se abordará únicamente este tipo de control de errores.

El control de errores **backward** puede dividirse en dos partes: la primera de ellas comprende las técnicas que pueden utilizarse para la detección de errores de forma fiable; la segunda engloba los algoritmos de control que se requieren para llevar a cabo las retransmisiones necesarias. El estudio de la primera parte se llevará a cabo en este apartado, siendo la segunda parte objeto de estudio en temas posteriores.

3.7.1. Paridad

Uno de los métodos más comúnmente empleados para detectar errores, cuando el número de bits de información a transmitir es pequeño y la probabilidad de que ocurra un error es baja, es el uso de un **bit** adicional **de paridad** por elemento transmitido.

En el caso más sencillo se añadirá un bit a cada carácter a transmitir cuyo valor será dependiente de su peso (número de unos que contiene el carácter). Si se utiliza paridad **par**, se añadirá un uno si el carácter original tiene por peso un número impar y un cero en caso contrario. Para paridad **impar**, el proceso es el inverso. Con este método puede detectarse la ocurrencia de un número impar de errores pero los errores pares no serán detectados. Resulta adecuado, por ejemplo, cuando se utiliza transmisión serie asíncrona, cada carácter transmitido puede constar de siete y ocho bits de datos más uno de paridad.

Puede conseguirse una importante mejora añadiendo un segundo grupo de bits de paridad, como puede verse en la siguiente tabla. Para ello deben agruparse los datos en bloques y aplicar el control de paridad a dos dimensiones (filas y columnas). Para cada carácter se añade un bit de paridad, como en el caso anterior. Además, se genera un bit de paridad para cada posición de bit a través de todos los caracteres. Es decir, se genera un carácter adicional en que el i-ésimo bit del carácter es un bit de paridad para el i-ésimo bit de todos los caracteres en el bloque.

	Bit 1	**Bit 2**		**Bit n**	**Bit de paridad**
Carácter 1	b_{11}	b_{21}		b_{n1}	P_1
Carácter 2	b_{12}	b_{22}		b_{n2}	P_2
Carácter m	b_{1m}	b_{2m}		b_{nm}	P_m
Carácter e paridad	C_1	C_2		C_n	C_{n+1}

El bit C_{n+1} se puede considerar como vertical, horizontal o diagonal. Los bits redundantes horizontales se conocen con las siglas **LRC** (*Longitudinal Redundancy Check*) y los verticales por **VRC** (*Vertical Redundancy Check*). Con este método pueden corregirse errores simples y detectarse dobles, triples y cuádruples si éstos no forman un rectángulo en la matriz de dígitos.

3.7.2. Códigos de redundancia cíclica (CRC)

Los códigos de **redundancia cíclica**, también conocidos como **códigos polinomiales** constituyen el método de detección de errores más empleado en comunicaciones. Se utiliza con esquemas de transmisión orientados a tramas (o bloques). Permiten sustanciales mejoras en fiabilidad respecto a los métodos anteriores, siendo a la vez una técnica de fácil implementación.

El método se basa en el uso de aritmética polinomial módulo 2 (No hay acarreos en la sustracción ni en la adición y las operaciones suma, resta y OR–exclusivo coinciden). La trama a transmitir –de *n* bits – representa un polinomio de coeficientes binarios. La idea consiste en añadir una secuencia de *k* bits, al final de la trama, de manera que la secuencia de *k+n* bits resultante constituya los coeficientes de un polinomio divisible de forma exacta por un polinomio *G(x)* determinado previamente por emisor y receptor. Cuando el receptor recibe la trama realiza la división entre *G(x)*, si el resto es distinto de cero ha ocurrido un error de transmisión.

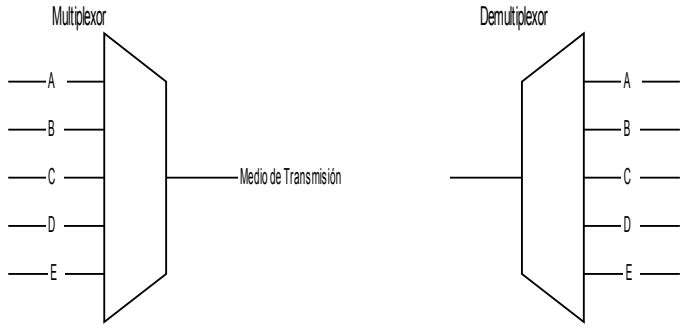
Imponiendo condiciones bastante simples sobre los polinomios divisores es posible detectar un gran número de errores. Existen tres polinomios *G(x)* que se han convertido en estándares internacionales:

CRC-12	$X^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-16	$X^{16} + x^{15} + x^2 + 1$
CRC-CCITT	$X^{16} + x^{12} + x^5 + 1$

Con secuencias de control de 16 bits, utilizando los polinomios CRC-16 y CRC-CCITT es posible detectar todos los errores simples y los dobles, todos los que afectan a un número impar de bits, todos los errores tipo ráfaga de 16 bits o menores, el 99,997% de errores ráfaga de 17 bits y el 99.998% de los de 18 bits y mayores.

Las herramientas utilizadas en la transmisión llevan aparejado un coste. El coste asociado al medio de comunicación es –casi siempre - despreciable para distancias cortas mientras que para largas distancias se incrementa notablemente. Sin embargo, a menudo las estaciones que se comunican no utilizan completamente la capacidad del medio de comunicación. Frecuentemente la comunicación entre ordenadores tiene un carácter no permanente, produciéndose ráfagas de tráfico esporádicamente, separadas por intervalos de silencio. Muchas de las aplicaciones cuentan con terminales remotos que acceden a un ordenador central a través de líneas de larga distancia. El tráfico que genera un terminal es típicamente bajo debido a que su operador humano no es capaz de teclear por encima de una cierta velocidad.

Por todo lo anterior, sería deseable que en aquellas situaciones donde se emplean líneas de transmisión de larga distancia (o de gran costa aunque no sean de gran longitud) conseguir el máximo aprovechamiento de la capacidad del medio de transmisión. Esto puede conseguirse mediante una técnica denominada **multiplexación**. Básicamente, consiste en compartir un mismo medio de transmisión entre varias comunicaciones, con lo que se divide el coste asociado a cada comunicación individual. El esquema general del proceso se muestra en la siguiente figura:



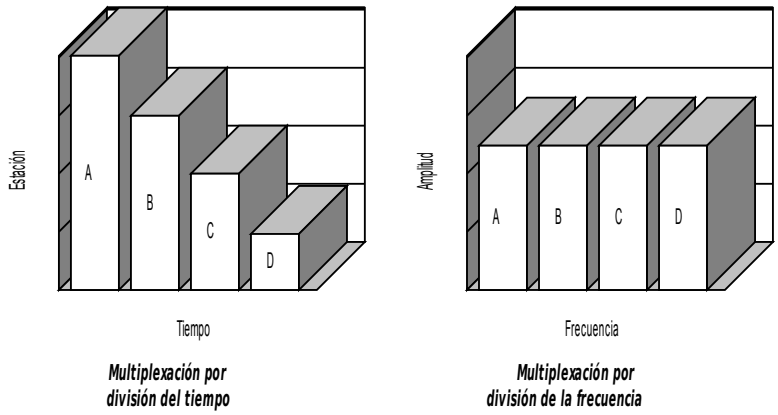
Multiplexación

El proceso de multiplexación es reversible y por tanto permite la transmisión simultánea de varias estaciones (A, B, C, D) por el mismo medio de transmisión.

Los dos métodos básicos que nos permiten realizar la multiplexación, son la **multiplexación por división del tiempo (MDT)** y la **multiplexación por división de la frecuencia (MDF)**.

En la multiplexación por división de la frecuencia se divide el ancho de banda del canal entre las señales a multiplexar. Cada estación puede transmitir simultáneamente a las demás. En este tipo de multiplexación, el ancho de banda del canal debe exceder a la suma de los anchos de banda de las señales que se transmiten, puesto que es necesario dejar bandas de seguridad entre las frecuencias asignadas a los diversos canales. Las señales transmitidas a través del medio deben ser señales analógicas.

En el segundo método, se asigna a las diferentes estaciones un turno de transmisión rotativo, durante un quantum de tiempo transmite una estación, luego la siguiente, ... etc. Luego realmente cada estación utiliza el canal alternativamente, no de forma simultánea. Seguidamente se ilustran las dos técnicas:



Técnicas de multiplexación

Se puede ver que mientras en la multiplexación por división el tiempo cada estación alterna su turno de transmisión con períodos de silencio, en la multiplexación por división de la frecuencia todas las estaciones transmiten durante todo el tiempo (o al menos pueden hacerlo), pero al disponer cada una de menor ancho de banda que el total del canal tendrán que transmitir más lentamente.

En algunas situaciones, como por ejemplo el empleo de terminales, indicado anteriormente, la transmisión se produce de manera esporádica y también existen períodos de silencio (sin transmisión). Si la técnica de multiplexación empleada es MTD, se producirá una situación poco conveniente. Se le asigna el turno para transmitir a una estación que no transmite durante el mismo.

Para solventar este problema se han desarrollado una serie de multiplexores MTD que reciben diversos nombres: **multiplexores estadísticos**, **multiplexores asíncronos** o **inteligentes**.

El propósito de estos dispositivos es aprovechar los tiempos muertos en las transmisiones de algunas estaciones para mejorar las prestaciones de la multiplexación al realizar el máximo aprovechamiento de la capacidad del medio utilizado. Para conseguirlo, estos dispositivos no asignan el turno de transmisión de una manera secuencial, sino que elaboran la información estadística obtenida del uso de cada entrada y asignan el turno mediante un algoritmo que intenta maximizar la utilización del canal.

Por ejemplo si después de un tiempo de funcionamiento síncrono (es decir, el comportamiento clásico de los multiplexores MDT, también denominado **MDTS**), se ha

obtenido la siguiente relación de utilizaciones:

Estación	Utilización (%)
A	20
B	15
C	40
D	100

Se entiende que las utilizaciones observadas están referidas al tiempo asignado a cada estación. Un 100% indica que la estación está transmitiendo permanentemente mientras dispone del turno, un valor inferior indica la presencia de espacios de silencio, tanto mayores cuanto menor es la utilización. SI a partir de los datos anteriores calculamos la utilización global del canal de transmisión:

$$U(\%) = \frac{1}{4}(20 + 15 + 40 + 100) = 43{,}75\%$$

Lógicamente observaremos que se está desaprovechando más de la mitad de la capacidad del canal, es decir, se está transmitiendo sólo durante menos de la mitad del tiempo. Se contemplan dos posibles inconvenientes:

- No aprovechar toda la capacidad del canal puede conducir a que los usuarios que lo comparten observen tiempos de respuesta peores en sus operaciones.

- En muchos casos el canal de transmisión es alquilado y su coste es por tiempo sin considerar la cantidad de datos que se transmiten, si no se aprovecha el coste de transmisión de los datos se incrementa.

Un posible **algoritmo de asignación dinámica del turno** se puede basar, por ejemplo, en conceder más tiempo a aquellas líneas que presenten utilizaciones del 100% e intervalos menores a las que tienen utilizaciones inferiores, haciendo así que la duración del turno de cada estación sea proporcional al volumen de tráfico generado, siempre y cuando la suma del tráfico de todas las estaciones no supere la capacidad del canal.

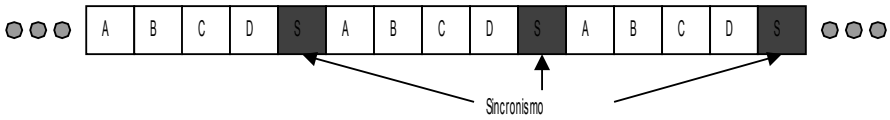
En ocasiones, podemos encontrar que alguna de las estaciones cuya transmisión estamos multiplexando, presenta largos intervalos de silencio. Cuando el algoritmo de planificación del multiplexor detecte esta circunstancia dejará de asignarle turno de transmisión a esa estación.

3.8.1 Sincronismo en multiplexores MDT

Como hemos visto, los multiplexores por división del tiempo, tanto los síncronos como los asíncronos, asignan turnos de transmisión a las estaciones. Para que no haya errores y se entregue el tráfico de forma correcta es preciso que multiplexor y demultiplexor se encuentren en perfecto sincronismo.

Debemos considerar además que es posible que haya errores en la comunicación, con lo cual los esquemas de sincronismo propuestos deberán ser capaces de recuperarse en situaciones de error.

Estudiemos primero un posible método para sistemas MDTS (síncronos) que tienen la particularidad de que el turno se asigna siempre en el mismo orden. En este caso podríamos sincronizar ambos extremos señalando únicamente el comienzo de cada nueva ronda de turnos, veamos como podría ser esto en el siguiente dibujo.



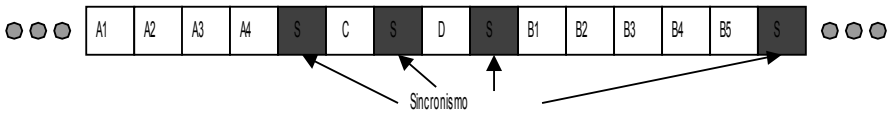
Sincronización en sistemas MDTS

Se ha indicado con una letra (A, B, C y D) los datos transmitidos par cada estación en el turno rotativo, la S representa una información especial de sincronismo que indica al demultiplexor que ha finalizado una ronda y comienza la siguiente. En el caso de que una estación no tenga información para enviar en su turno se enviaría un código especial de relleno que el multiplexor descartaría sin más.

Si suponemos que todas las estaciones transmiten de manera permanente se produce una utilización del 100% del canal pero sólo se aprovecha el 80% de su capacidad debido a la sobrecarga del sincronismo (1 de cada 5 datos). Si las estaciones no transmiten al 100%, entonces la capacidad del canal que se aprovecha decrece rápidamente y siempre por debajo de la media de utilización de las estaciones (debido a la sobrecarga del sincronismo).

Para esta última circunstancia se han ideado los multiplexores asíncronos (MTDA) que evalúan la transmisión de cada estación y le asignan un turno proporcional a su demanda. Sin embargo, puesto que estos multiplexores no asignan el turno de manera secuencial, el problema de la sincronización entre multiplexores y demultiplexores se acentúa.

Cada cambio del turno deberá ser comunicado al demultiplexor para que no haya problemas puesto que no se trabaja secuencialmente y cada estación puede transmitir un número variable de datos. Veamos en la siguiente figura un posible método:



Sincronización en sistemas MDTA

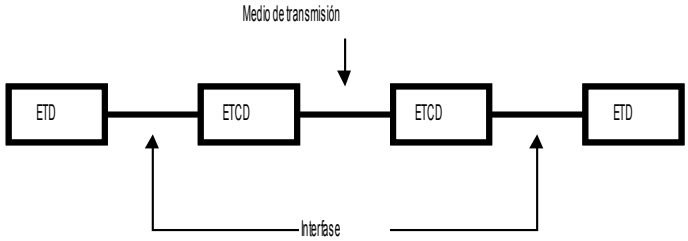
Aunque no indicada en el dibujo, cada sincronización con el dato “S” se supone que informará al demultiplexor que estación va a tener el turno seguidamente.

Esta técnica resulta ventajosa únicamente si se emplea en los escenarios para los que está pensada, o sea, aquellas comunicaciones en las que no todas las estaciones tienen datos para emitir durante todo el tiempo. La MTDA mejora el aprovechamiento del canal frente al MDTS y los tiempos de respuesta de las estaciones que están transmitiendo.

Sin embargo, cuando el tráfico de todas las estaciones es constante, la MDTA impone una sobrecarga del 50% a la utilización del canal con lo que resulta muy poco recomendable, ya que en esas circunstancias la MTDS proporciona un aprovechamiento del 80% (una sobrecarga del 20% en el sincronismo).

Por supuesto que diferentes mejoras pueden ser introducidas en los esquemas simplificados que se han expuesto. De hecho, los multiplexores comerciales emplean muchas más técnicas (incluida, a veces, la compresión de datos) para conseguir mayor aprovechamiento del canal que el indicado aquí, que sólo pretenden ilustrar sobre ambos métodos.

La mayoría de los ordenadores dispone de una capacidad limitada de transmisión. Habitualmente son capaces de proporcionar los datos codificados como una señal NRZ o algún otro código, pero estas señales no son adecuadas para ser transmitidas por los medios de transmisión. Existen unos dispositivos especializados que realizan la función de conversión de las señales que proporciona el ordenador a las señales que se pueden transmitir con facilidad a través del medio de transmisión escogido. Según la terminología empleada por el CCITT estos dispositivos reciben el nombre de **ETCD (Equipos Terminales de Circuito de Datos)** y los ordenadores reciben la denominación de **ETD (Equipos Terminales de Datos).**



El tipo de ETCD más conocido es el **módem**, cuya función es precisamente convertir las señales digitales que suministra el ordenador en señales analógicas que puedan ser transmitidas con facilidad largas distancias.

Para facilitar la conexión entre ETD y ETCD se han desarrollado múltiples estándares que determinan todas las características físicas, eléctricas, mecánicas y funcionales de la conexión constituyendo lo que denominamos la **definición de un *interfase***. Estos estándares constituyen un ejemplo de los protocolos del nivel físico, y se encuadrarían en el nivel más bajo del modelo de referencia OSI.

Posiblemente el más conocido y popular es el "Recomended Standard 232". El **RS-232** es una norma para la conexión entre un ETD y un ETCD que define :

0 El tipo de conector a emplear

1 Las características eléctricas

2 Los niveles de tensión

3 Las longitudes máximas a distintas velocidades

4 Los nombres de las señales que intervienen en el funcionamiento y la estrucutra del protocolo de comunicación

Esta norma establece una señalización eléctrica bipolar, según la siguiente tabla:

Nivel lógico 0	+15..+3 Voltios
Nivel lógico 1	-15..-3 Voltios

Las velocidades de transmisión que puede soportar este estándar van desde los 0bps hasta los 20Kbps. Con respecto a las distancias máximas se propone que no sean superiores a 15 metros. Aunque un diseño cuidadoso puede permitir distancias muy superiores, hay que suponer que esta limitación teórica se puede manifestar en la práctica en dispositivos que

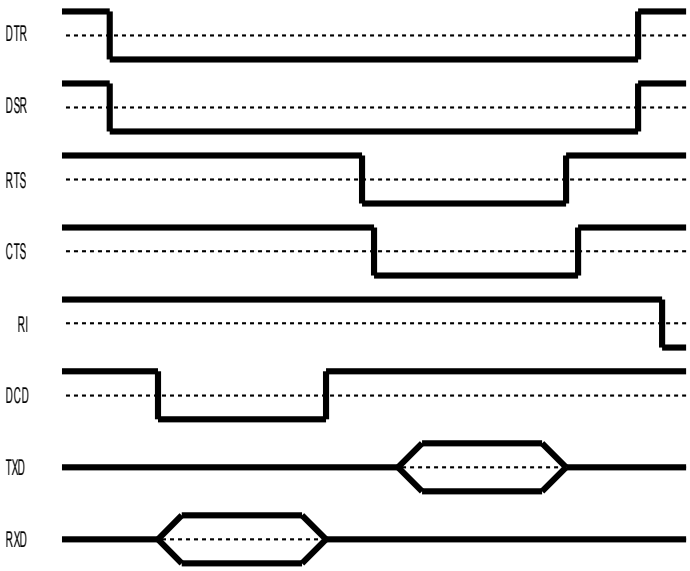
cumplan la norma.

En la siguiente tabla se muestran las principales señales que componen este interfase y su descripción. La columna E/S toma como referencia el ETD.

Mnemónico	Nombre en Inglés	E/S	Descripción
DTR	Data Terminal Ready	S	El ETD indica que está preparado
DSR	Data Set Ready	E	El ETCD indica que está preparado
RTS	Request To Send	S	Solicita permiso para transmitir
CTS	Clear To Send	E	El ETCD autoriza la transmisión solicitada
TXD	Transmitted Data	S	Línea de transmisión de datos serie
RXD	Received Data	E	Línea de recepción de datos serie
GND	Ground	-	Masa de referencia 0v
RI	Ring Indicator	E	Detección de llamada
DCD	Data Carrier Detect	E	Detección de portadora

Seguidamente vamos a comentar el funcionamiento conjunto de las señales descritas en su uso normal, es decir, en una comunicación entre ordenador (ETD) y módem (ETCD).

La siguiente figura detalla la evolución de señales que hemos relacionado anteriormente. No debe olvidarse que estamos viendo sólo un subconjunto de todas las señales que se definen en la norma.



Funcionamiento del interfaz RS-232

Las señales DTR y DSR son un requisito previo a la comunicación a través de esta interfaz, puesto que informan a cada dispositivo - ETD y ETCD - del estado del otro, indicando si

están o no preparados para funcionar.

La señal DCD indica si se detecta portadora en el canal de transmisión. La detección de portadora es un requisito previo a la recepción, debe existir portadora antes y durante la recepción. Cuando desaparece la portadora, sabemos que no se van a recibir más datos.

En canales semi-duplex habrá que esperar a que finalice la recepción antes de poder iniciar una transmisión, lo que no será necesario si el canal es dúplex. Para iniciar una transmisión, el ordenador activará la línea RTS y esperará a que le autorice el módem por medio de la línea CTS. En cuanto esta se active, el ordenador podrá proceder a transmitir los datos.

Ocasionalmente, y siempre en una situación de inactividad de los equipos (estado de reposo), se puede producir la activación de la señal RI, que indica que el módem está recibiendo una llamada por la red. Esta situación podría servir para iniciar el establecimiento de una nueva comunicación.

Subnivel de Acceso al Medio (M.A.C.)

Para este estudio dividiremos las redes de ordenadores en dos categorías. La primera de ellas comprende todas aquellas redes cuyas líneas de transmisión son líneas dedicadas punto a punto. Esta comunicación puede ser utilizada por una comunicación, con lo cual en los tiempos muertos de la misma la línea permanece inactiva. El segundo grupo comprende aquellas redes en las cuales el medio de transmisión consiste en un canal de difusión, de tal modo que un mismo canal puede ser utilizado por varias transmisiones. En este bloque nos centraremos en las redes de difusión y sus protocolos respectivos.

En este tipo de redes el problema está en determinar quien utilizará el canal para la transmisión cuando existen varios emisores compitiendo por él. Estos canales de gran ancho de banda se les conoce también como *canales multiacceso* o *canales de acceso aleatorio*. La subcapa MAC es especialmente importante en las LAN (redes de área local), dado que casi todas ellas utilizan un canal de acceso múltiple como base para sus comunicaciones. A diferencia de esto, una WAN (red de área extendida) utiliza enlaces punto a punto, con excepción de las redes satélite. Dado que los canales de acceso múltiple y las LAN están muy relacionadas, en este capítulo se discutirán las LAN en general, así como los satélites y algunas otras redes de difusión.

Para todos los protocolos que veremos a continuación realizaremos unos supuestos que comentamos a continuación.

- **Modelo de estación**: Vamos a considerar redes en las cuales existen N estaciones independientes, cada una de las cuales con un usuario o programa generando tramas con una probabilidad constante. Una vez que una estación ha generado una trama, permanece bloqueada hasta el momento en que consigue emitirla satisfactoriamente.

- **Canal único**: Asumimos que existe un único canal para la comunicación. Todas las estaciones pueden emitir y recibir de él. En este sentido, todas las estaciones serán equivalentes a pesar de que el protocolo pueda asignarles diferentes prioridades.

- **Colisión**: Asumimos que si dos tramas intentan transmitir simultáneamente por el canal, se superponen las señales de cada una de ellas dando lugar a otra señal distorsionada. Este evento es el que denominamos colisión. Toda trama que ha intervenido en una colisión debe ser retransmitida nuevamente. No existen más errores que los que se producen por colisiones.

- **Tiempo continuo**: Bajo esta hipótesis, la transmisión de una traza puede comenzar en cualquier momento. No existe un reloj centralizado que divida el tiempo en intervalos discretos.

- **Tiempo discreto**: El tiempo se encuentra dividido en intervalos discretos, a los que denominaremos ranuras o slots. La transmisión de una trama siempre comienza en el principio de una ranura. Una ranura puede contener 0, 1 o más tramas, correspondiendo este estado a una ranura ociosa, una transmisión correcta o a una colisión, respectivamente.

- **Detección de portadora**: Bajo este supuesto, las estaciones pueden detectar si el canal está siendo utilizado antes de transmitir por él. Si se detecta que el canal está ocupado, ninguna estación podrá utilizarlo hasta que quede disponible.

- **Sin detección de portadora**: Las estaciones no pueden espiar el canal antes de utilizarlo. Simplemente, envían su trama, y únicamente entonces son capaces de detectar si la transmisión ha sido correcta.

En 1970 Norman Abramson y su grupo diseñaron un método, basado en la difusión por radio, para la solución del problema de que varios usuarios compartieran simultáneamente el mismo recurso, dando lugar al sistema **ALOHA**.

4.2.1 El protocolo ALOHA

El protocolo ALOHA se basa en un principio muy simple. Se permite que los usuarios transmitan cuando tengan datos para ello. Por tanto, existirían colisiones y los bloques involucrados en las mismas deberán ser invalidados. Asimismo, será necesaria una retransmisión de las tramas colisionadas. Esta retransmisión se realizará después de un tiempo de espera por parte del emisor, tiempo que deberá ser aleatorio para evitar que los sucesivos intentos sufran colisiones de manera continuada. En general, los sistemas en los que los múltiples usuarios comparten un canal común, de tal forma que pueda llevarlos a conflictos, se conocen como **sistemas de contienda**.

Puesto que un emisor puede comenzar la transmisión en cualquier momento, es necesario destacar que si el primer bit de una trama se emite a la vez que el último bit de otra, ambas tramas se consideran perdidas.

Todas las tramas han de tener la misma longitud, porque el rendimiento de un sistema ALOHA es máximo cuando se tiene un tamaño uniforme de trama, en vez de permitir tener tramas de longitudes variables.

Para una mayor eficacia del algoritmo conviene minimizar la probabilidad de colisión. Para ello se divide el tiempo en ranuras de igual duración. Así, cuando una estación quiere enviar debe esperar al comienzo de una ranura para iniciar la transmisión. A esta variante se le denomina **ALOHA ranurado** (Slotted ALOHA).

4.2.2 Protocolo CSMA-1 Persistente

Con este protocolo comenzamos un conjunto de los mismos orientados a redes locales. En todos estos protocolos de L.A.N., a diferencia del *ALOHA ranurado*, las estaciones escuchan si hay presencia de portadora en el canal antes de iniciar la transmisión. Por lo tanto estos protocolos se denominan **Protocolos de detección de portadora** *(*Carrier Sense Protocols).

El primero que veremos es el **CSMA 1-Persistente** (Carrier Sense Multiple Access). Como hemos dicho, cuando una estación tiene datos para transmitir, primero escucha el canal para ver si hay alguien transmitiendo. Si el canal está ocupado, la estación espera a que quede ocioso. Cuando la estación detecta que el canal está disponible comienza a transmitir un *frame.*

Si ocurre una colisión mientras transmite, la estación espera un tiempo aleatorio y comienza de nuevo la transmisión.

Este protocolo se denomina **1-Persistente** porque la estación emite siempre (con probabilidad 1) si encuentra el canal libre.

Este protocolo es mejor que el ALOHA, ya que las estaciones verifican que el canal se encuentre libre antes de emitir. Por tanto, el número de colisiones disminuye con relación a este otro protocolo.

Sin embargo, cuando el canal se encuentra disponible puede ocurrir que dos o más estaciones

intenten ocuparlo simultáneamente, ya que el algoritmo es idéntico para todos. En ese momento se producirá una colisión. Resulta especialmente probable que suceda una colisión en el momento en que finaliza una transmisión, ya que puede ocurrir que existe más de una estación esperando a que el canal quede libre. En este caso se producirá una colisión entre éstas estaciones.

4.2.3 Protocolo CSMA No persistente

Otro de los protocolos con detección de portadora es el **CSMA no persistente**. En este protocolo, la estación escucha el canal antes de empezar a transmitir; si nadie está transmitiendo, la estación empieza a hacerlo sola. Sin embargo, si el canal ya se encuentra en uso, la estación no está escuchando el canal continuamente, con el propósito de utilizarlo en el momento en que detecte la terminación de la transmisión anterior, sino, más bien, espera un intervalo de tiempo aleatorio, para después repetir el algoritmo. Intuitivamente se ve que este algoritmo nos conduce a una mejor utilización del canal y a mayores retardos que los encontrados en el CSMA 1-persistente.

4.2.4 Protocolo CSMA-P Persistente

Este algoritmo intenta evitar la alta probabilidad de colisión tras una transmisión, como comentábamos anteriormente. Para ello estableceremos un canal ranurado. Su funcionamiento será el siguiente: tras una transmisión, las estaciones que estaban esperando a que se liberara la línea no acceden a la misma directamente, sino que accede con una probabilidad *P*. Esto quiere decir que, cuando una estación que quiere transmitir ve que se libera la línea, existe una probabilidad *P* de que comience una transmisión, mientras que con probabilidad *Q=1-P* esperará la siguiente ranura de tiempo.

Frente a este tipo de algoritmos la pregunta que se plantea es la duración del ancho de ranura o *slot*. Supongamos que una estación emisora *A* envía un paquete y en el preciso instante en que va a llegar a otra estación *B*, esta última emite otro paquete, produciéndose una colisión.

El tiempo transcurrido hasta la colisión ha sido el de propagación de la señal por el canal t_p, y desde que se produce hasta que *A* sé de cuenta de la misma transcurre otro periodo t_p. Por tanto, podemos decir que, si las dos estaciones están lo más alejadas posible, el tiempo de ranura será superior o igual a dos veces el tiempo máximo de propagación de la señal por el canal. Con esto conseguiremos reducir el número de colisiones de una manera significativa, mejorando el rendimiento de la red. Hay que destacar que éste rendimiento se ve muy afectado por el número de colisiones, ya que una colisión implica las retransmisiones de todas las tramas implicadas, lo cual afecta negativamente el rendimiento de la línea.

4.2.5 Protocolo CSMA/CD

El protocolo **CSMA/CD** (Carrier Sense Multiple Access with Collision Detection) es ampliamente utilizada en redes L.A.N. en el ámbito de M.A.C.

Hemos visto anteriormente los protocolos persistentes y no persistentes como mejoras sobre el original ALOHA. Sin embargo estos protocolos no aprovechan suficientemente el canal, debido a que cuando se produce una colisión las estaciones no reaccionan hasta que terminan de transmitir la trama que ha colisionado. Por ello, se desperdicia bastante tiempo emitiendo tramas que, puesto que han colisionado, no sirven para nada. El protocolo CSMA/CD añade a los protocolos anteriores la capacidad de detectar cuando una trama está colisionando con otras a partir del momento en que colisionan. Esto nos permite cancelar la transmisión de dicha trama, ahorrando tiempo, en lugar de finalizar la transmisión como en protocolos anteriores.

En este protocolo, cuando el canal está libre y hay dos o más estaciones que desean transmitir sus tramas se producirá una colisión. Cada estación implicada detectará el hecho y esperará un tiempo aleatorio. A continuación intentará la retransmisión, observando nuevamente el canal para ver si existe actividad.

Queda por resolver una cuestión que ya comentamos en protocolos anteriores, que es el tiempo máximo que tardarán dos estaciones en darse cuenta de una colisión. Como habíamos deducido, este tiempo coincide con el doble del tiempo de propagación t_p de la señal entre las estaciones más alejadas.

Este tiempo constituirá al ancho de un *slot* o ranura. Por tanto, adoptaremos un canal ranurado con ancho de ranura de $2^* t_p$.

En caso de colisión cada estación esperará un tiempo aleatorio. Normalmente se utiliza el algoritmo denominado *Exponential Backoff*, por sus buenos resultados. Este algoritmo consiste en que, en caso de colisión, la estación espera *K* slots antes de reintentar la transmisión. *K* se calcula como un valor aleatorio entre 0 y $K = 2^i - 1$, donde *i* es el número de reintentos que se ha realizado con anterioridad sobre la misma trama.

En estos protocolos, para que la colisión sea fácilmente detectable es necesario utilizar una codificación determinada. Generalmente se utiliza la codificación *Manchester* de tres estados. En dicha codificación se representan cada uno de los valores lógicos de la siguiente forma:

- Un **1** lógico se representa mediante 0,85V en la primera mitad de la celda de bit y un –0.85V en la segunda.
- Un **0** lógico se representa mediante –0,85V en la primera mitad de la celda de bit y un 0.85V en la segunda.
- La línea inactiva se representa mediante un valor de 0V.

Por tanto, resulta muy sencillo detectar una colisión, ya que existen combinaciones (ambas partes de la celda de bit a nivel bajo o nivel alto) que no están permitidas, y sólo pueden darse en caso de colisión.

En estos protocolos se van a evitar las colisiones mediante la asignación de unos turnos a cada estación. Para esto es necesario contar con un periodo de decisión o arbitraje anterior a la transmisión de las tramas para ver cual de las estaciones implicadas se le concede el derecho de emisión.

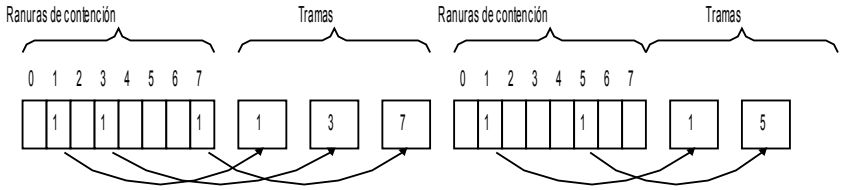
Asumiremos que tenemos *N* estaciones, cada una de ellas con una dirección única en el rango [0...*N*-1]. Las estaciones pueden estar inactivas durante indeterminados periodos de tiempo.

La diferencia fundamental entre los protocolos se basa en la decisión de qué estación obtendrá el canal tras una transmisión correcta. Dependiendo del algoritmo que se siga tendremos distintos protocolos.

4.3.1 Mapa de bits

En este método, en cada periodo de contención tendremos exactamente *N* ranuras, tantas como estaciones existentes. Durante este periodo cada estación que desea transmitir pone en su ranura un bit a uno. Pasado el plazo de contención (que todas las estaciones han observado), la transmisión se realiza ordenadamente por número de estación. Cuando finalizan las transmisiones de todas las estaciones que lo han solicitado, se establece un nuevo periodo de contención.

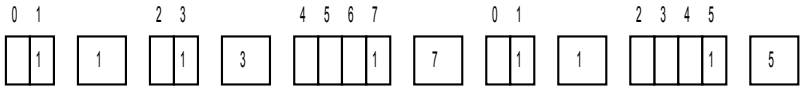
Para una mejor compresión de este método, a continuación se plantea un ejemplo con *N*=8:



4.3.2 BRAP (Broadcast Recognition with Alternating Priorities)

El esquema anterior basado en la técnica del **Mapa de bits** funciona de forma eficiente cuando la cantidad de estaciones que desean transmitir es alta. Sin embargo, para cargas bajas la eficiencia es baja. Además, una estación tiene que esperar como mínimo a que termine el tiempo de contención actual antes de poder enviar su trama. Así mismo, también se observa que las estaciones con números identificativos más elevados disponen de un tiempo menor entre una transmisión y el siguiente periodo de contención para decidir si emiten una segunda trama.

Estas cuestiones son resueltas por el **BRAP** mediante la siguiente técnica:



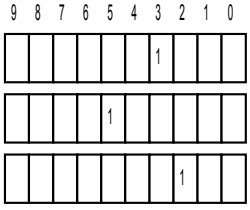
El funcionamiento del protocolo es el siguiente: Cuando una estación inserta un 1 en su ranura correspondiente, entonces puede transmitir de inmediato la trama solicitada. Cuando esta transmisión finalice se vuelve a un periodo de contención, pero la primera ranura de este periodo corresponde a la estación siguiente a la que acaba de emitir. De esta manera, todas las estaciones tienen el turno con igual probabilidad. Con este mecanismo se consigue una eficiencia idéntica a la del mapa de bits, pero se consigue reducir el tiempo de espera.

4.3.3 MLMA (Multi Level Multi Access)

El inconveniente del **BRAP** no es la utilización del canal (que es excelente cuando existen altas cargas), sino el retraso que se produce cuando existe una baja carga, es decir, pocas de las estaciones desean emitir simultáneamente. En este caso, se envían las ranuras una y otra vez hasta que un canal indique su intención de transmitir mediante un 1 en su ranura. Por tanto, una estación deberá esperar para acceder al canal una media de *N/2* ranuras, cuando el canal se encuentra desocupado.

Para conseguir condiciones óptimas en todo tipo de cargas (estados de la red), se utiliza el protocolo **MLMA**. Con este protocolo se reduce el número de ranuras necesarias para indicar las estaciones que desean transmitir.

En **MLMA** cada estación anuncia su petición de envío colocando su dirección en el canal. Dicha dirección será de la forma que ilustra el ejemplo siguiente: Supongamos que *N*=1000 estaciones. En este caso tendremos que enviar tres dígitos como identificación de la estación. Por este método, la estación 352 se identificará mediante las tramas:



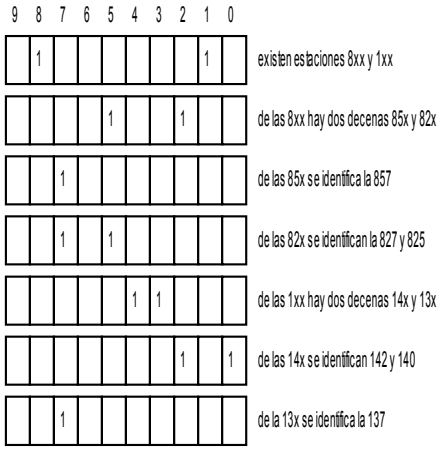
Indicando respectivamente la posición de las centenas, decenas y unidades de la dirección. Así, si la estación 352 desea enviar, tras identificarse mediante su dirección efectúa la transmisión de la trama.

Sin embargo, si existen varias estaciones que desean emitir, durante cada una de las décadas anteriores volcarán todas ellas su dirección en el canal siguiendo el algoritmo siguiente:

Todas las estaciones ponen en la línea su centena (1 década = *C*).
Para cada bit $C_j = 1$
Las estaciones con centena *i* ponen su decena en el canal (década *D*)
Para cada bit $D_j = 1$ en la década *D*
Las estaciones con decena *j* ponen sus unidades en el canal
(con este último paso las estaciones quedan identificadas)

Una vez finalizada esta fase, todas las estaciones que desean enviar tramas han sido identificadas, por lo que se puede proceder a la transmisión de tramas en orden de prioridad.

A continuación veremos un ejemplo del funcionamiento de este mecanismo: Supondremos que en un sistema con *N*= 1000 estaciones en un momento dado pretenden enviar tramas las estaciones 827,825,857,137,142 y 140. El funcionamiento del protocolo en este caso sería el siguiente:

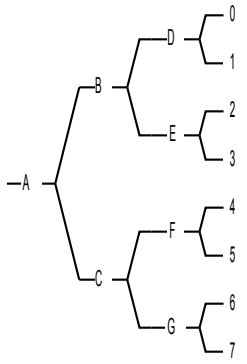


Cada estrategia (con o sin colisión) puede calificarse de acuerdo con su rendimiento con respecto al comportamiento de los dos parámetros más importantes, es decir, el retardo para situaciones de poca carga y la eficiencia del canal para condiciones de carga alta. Para condiciones de carga moderada es aconsejable el uso de un protocolo de contienda ya sea este puro o ranurado debido a que su retardo es mínimo; pero a medida que la carga se incrementa, la contienda es cada vez menos atractiva, debido a que la sobrecarga asociada con el arbitraje del canal es mayor. Lo contrario es válido para el caso de los protocolos libres de colisión Estos tienen un gran retardo para una carga baja, pero a medida que se incrementa esta, la eficiencia del canal mejora, más que tender a empeorar, como sucede con los protocolos de contienda.

Los **protocolos de contienda limitada** combinan las mejores propiedades de los protocolos de contienda y libres de colisión para dar lugar a uno nuevo, que utiliza el de contienda para condiciones de baja carga, con el objeto de tener un retardo pequeño, y que, al mismo tiempo, utilizará una técnica libre de colisiones para cargas elevadas, y poder tener una buena eficiencia de canal.

4.4.1 Protocolo de recorrido adaptativo de un árbol

Consideremos una organización de estaciones basada en un árbol binario.



En la primera ranura de contienda que sigue a la transmisión con éxito de una trama, *ranura0*, todas las estaciones están autorizadas para tratar de hacerse con el canal. So alguna de ellas lo logra, magnífico. Si hay colisión, entonces, durante la *ranura1* solamente podrán competir por ella aquellas estaciones que se encuentren bajo el nodo *B*. Si una de ellas toma posesión del canal, la ranura que viene a continuación de la trama se reserva para aquellas estaciones que se encuentren bajo el nodo *C*. Si, por otra parte, dos o más estaciones que se encuentren bajo el nodo *B* desean transmitir, habrá una colisión durante la *ranura1*, en cuyo caso el turno será ahora para el nodo *D*, durante la *ranura2*.

En esencia, si ocurre una colisión durante la *ranura0*, se examinará todo el árbol, primero a fondo, con objeto de localizar todas las estaciones que se encuentren listas. Cada ranura de bit está asociada con algún nodo del árbol en particular. Si ocurre una colisión, el examen continúa, en forma recursiva con los descendientes (hijos) que están localizados a la izquierda y derecha de dicho nodo. Si una ranura de bit pasa a estado inactivo, o bien, si hay exactamente una estación que transmita en dicha ranura, se puede detener el examen del nodo, porque todas las estaciones que están listas ya se habrán localizado. (Si hubiera más de una, tendría que haber sucedido una colisión).

Control de enlace de datos

En ese capítulo se estudiará el diseño de la capa de enlace. En este estudio se tratarán los algoritmos por medio de los cuales se puede llevar a cabo una comunicación fiable y eficiente en la capa de enlace, entre dos máquinas adyacentes. Por adyacente, se quiere dar a entender que las dos máquinas se encuentran físicamente conectadas mediante una canal de comunicación que actúa como un cable (por ejemplo, un cable coaxial o una línea telefónica). La propiedad fundamental que hace que un canal sea *semejante a un cable*, es que los bits se entregan exactamente en el mismo orden en el cual se transmiten.

Al principio se podría pensar que este problema es tan trivial, que no es necesario estudiar ningún tipo de software en particular, es decir, la máquina *A* pone los datos en el cable, en tanto que la máquina *B*, sólo se encarga de sacarlos. Desafortunadamente, los circuitos de comunicación comenten errores de cuando en cuando. Además, sólo tienen una velocidad de transmisión de datos finita y hay un retardo de propagación diferente de cero, entre el tiempo que transcurre desde que un bit se envía hasta que se recibe. Estas limitaciones, asociadas con la velocidad finita de las máquinas para procesar los datos, tienen implicaciones muy importantes en la eficiencia de la transferencia de datos. Los protocolos que se utilizan para las comunicaciones deberán tomar en consideración todos estos factores y son el tema de este capítulo.

Es importante, para que una comunicación sea fiable, establecer una supervisión de la transmisión, comprobando que no se producen errores y, en el caso de que ocurran, corrigiéndolos de forma transparente, para evitar que un error en la transmisión pueda originar un error de comunicación.

El principal propósito de los **protocolos de enlace de datos** es garantizar que la comunicación entre dos máquinas directamente conectadas está libre de errores. Para conseguir este objetivo, habitualmente se estructura la información a transmitir en pequeños bloques de datos, cada uno de los cuales lleva asociado un código detector de error y un número de secuencia. Dichos bloques se envían de forma secuencial y si uno de ellos sufre un error será reenviado por el transmisor. Se consigue así que un error no implique la retransmisión de todo el mensaje, sino sólo una pequeña parte del mismo.

Para indicar esta estructura de la información en bloques se hace necesario que los datos transmitidos incorporen algún tipo de marca que señalice el comienzo y el final de cada bloque, para que el receptor pueda detectar esta circunstancia. Típicamente esto se consigue añadiendo cierta información extra. A toda la información requerida por el protocolo de enlace de datos para su uso interno - números de secuencia, delimitadores, etcétera - la denominaremos **información de control**.

El nivel de enlace de datos tiene un número de funciones específicas por desarrollar. Entre esas funciones, los protocolos de enlace de datos deben realizar las siguientes:

- Proporcionar un servicio bien definido al nivel superior (de red).

- Agrupar los bits o caracteres recibidos por el nivel físico en bloques de información, tramas (o frames), a los que va asociada información de control para proporcionar los servicios.

- Detectar y solucionar los errores generados en el canal de transmisión.

- Control de flujo, para evitar saturar al receptor.

- Control de diálogo, en canales semi - dúplex será necesario establecer los turnos de transmisión.

5.1.1 Servicios proporcionados al nivel de red

El nivel de enlace de datos puede ser diseñado para ofrecer diferentes clases de servicios. La clase de servicio puede variar de un sistema a otro. Las tres posibilidades más razonables son:

- **Servicio sin conexión y sin asentimiento:** Consiste en hacer que la máquina fuente mande tramas a la máquina destino sin que esta última tenga que reconocerlas. No se establece ninguna conexión (o acuerdo previo) antes de la transmisión de los datos. Si una trama se pierde o queda dañada por ruido en el canal de transmisión no será misión del nivel de enlace el corregir la deficiencia.

- **Servicio sin conexión pero con asentimiento:** Significa que por cada trama que manda espera que le llegue un reconocimiento. De esta manera, el emisor sabe si la trama ha llegado satisfactoriamente. Si no llega el reconocimiento correspondiente pasado un tiempo determinado desde la emisión de la trama, el emisor asume que su trama no llegó o llegó dañada y la retransmite.

- **Servicio con conexión:** Es el servicio más sofisticado que el nivel de enlace de datos puede proporcionar al nivel de red. Con este servicio las máquinas fuente y destino establecen una conexión antes de transmitir los datos. Cada trama que se envía, sobre la conexión establecida, se numera y el nivel de enlace garantiza que cada trama se recibe una sola vez y que se reciben en el orden correcto. Esto no se puede garantizar con el servicio sin conexión, pues la pérdida de reconocimiento provoca que una trama pueda ser enviada varias veces y, por lo tanto, recibida otras tantas.

La configuración del enlace de datos vendrá establecida principalmente por tres características de la transmisión: el tipo de canal, el modo de transmisión y la disciplina de línea.

5.1.2 Entramado

La unidad de intercambio de información en los protocolos de enlace de datos es la **trama**. Una trama es un bloque de datos que además contiene información de control, empleada por el protocolo para identificar a la trama.

Cuando el método de transmisión utilizado está orientado a bloques (como muchas transmisiones síncronas) la sincronización de trama ya está resuelta por el método de transmisión. Por el contrario, al usar métodos de transmisión orientados a caracteres, como la transmisión serie asíncrona, la sincronización de trama debe ser resuelta por algún procedimiento adicional.

Lo que se pretende conseguir es determinar dónde empieza y dónde acaba una trama en el flujo de caracteres que recibimos. Podemos considerar múltiples soluciones:

Basar la detección del final de trama en un temporizador

Transcurrido un tiempo sin recibir ningún carácter se supone que ha finalizado la recepción de la trama. Las redes sin embargo, muy rara vez llegan a garantizar esta temporización, por lo que es posible que estos intervalos lleguen a desaparecer, o bien, que otros intervalos se inserten durante la transmisión. Dado que es demasiado arriesgado usar la temporización para marcar el inicio y el final de cada trama, se han diseñado otros métodos alternativos para tal efecto.

Conteo de caracteres

En este método se agrega un campo en la cabecera, para especificar el número de caracteres en la trama. En el momento en el cual la capa de enlace ve, en el extremo destinatario, la cuenta de caracteres, se entera del número de caracteres que siguen y, por consiguiente, donde termina la trama. El problema de este algoritmo es que la cuenta puede distorsionarse por un error de transmisión. Aun cuando el código de redundancia cíclica sea incorrecto, de tal manera que el destinatario sepa que la trama es también incorrecta, todavía no tiene manera de decir dónde comienza la siguiente trama. El hecho de enviar una trama incorrecta al extremo origen para solicitarle que retransmita, tampoco ayuda de manera significativa, dado que el extremo destinatario no sabe cuántos caracteres debe omitir para llegar al inicio de la retransmisión. Por esta razón, el método de cuenta de caracteres se utiliza muy rara vez actualmente.

Utilizar caracteres especiales, o secuencias de bits

que indiquen el comienzo y el fin de las tramas. (Esta idea plantea problemas de transparencia de datos. Añadir información de control para delimitar las tramas se emplea con frecuencia, pero presenta el problema de que las mismas combinaciones utilizadas podrían parecer en la confirmación a transmitir, generándose un conflicto de interpretación en el receptor al detectar un falso final de trama.

Este es el problema de la **transparencia de datos**, donde se puede confundir entre datos y secuencias de control. Existen distintas soluciones para este problema y varían según la técnica de transmisión empleada sea orientada a carácter o a bloques.

En la transmisión orientada a caracteres se puede emplear la técnica de **relleno de carácter** (*character Stuffing*), que se basa en insertar un carácter especial en las secuencias de control para conseguir que éstas sean irrepetibles.

En la siguiente secuencia se puede observar el funcionamiento:

dle stx dato1 dato2 dato3 dle dato4 dato5 dle etx

donde:
dle: escape de enlace (data link escape)
stx: Inicio del texto (start of text)
etx: Fin del texto

El **dle** subrayado indica que ha sido insertado en el procesado del bloque aplicando la técnica de relleno de carácter, debido a que **dato4**=dle. Sólo en ese caso y para asegurar la transparencia a de datos se inserta (o duplica) cada **dle** que aparece en los datos.

Con este método, las secuencias **dle stx** y **dle etx** indican, respectivamente el principio y el final del bloque de información, y para evitar que se repitan dentro del bloque de datos se duplican los **dle**, con lo cual si se desea transmitir precisamente la secuencia **dle etx** dentro de los datos, esta se transmitirá como **dle dle etx**, con lo que el receptor no confunde con el fin de trama, ya que la recepción de los **dle** seguidos le indica que no debe interpretarlo como una secuencia de control, y además debe eliminar uno de esos dos caracteres.

Cuando utilizamos transmisiones orientadas a bit (por ejemplo: transmisión síncrona), la transparencia de datos se logra por medio de la técnica denominada **relleno de bit** (*Bit Stuffing*), que en esencia trabaja sobre el mismo principio que la anterior, pero insertando un solo bit en vez de un carácter, lo que hace *a priori* más eficiente.

El relleno de bit consiste en insertar un cero cada vez que se encuentran dentro del bloque de datos cinco unos seguidos. Esto se hace para evitar que se puedan generar secuencias en la transmisión que coincidan accidentalmente con el *flag* empleado como señalizador de

comienzo y final de bloque en la transmisión síncrona y que suele ser la secuencia 01111110.

Emplear Violaciones del código

para señalizar las condiciones de principio y final de trama. Estas violaciones consisten en condiciones anormales de la transmisión. Por ejemplo, la codificación Manchester establece la existencia de una transición *alto-bajo* o *bajo-alto* para la codificación de los bits. La utilización de combinaciones sin transición *bajo-bajo* o *alto-alto* (y por tanto inválidas bajo el esquema Manchester) permite encapsular claramente las tramas, y no requieren el proceso de inserción del Método anterior.

5.1.3 Control de Error

Habiendo solventado el problema de marcar el principio y el final de la trama el siguiente problema a tratar es como garantizar que todas las tramas llegan a su destino en condiciones. Supongamos que el emisor se limita a mandar la trama sin preocuparse de si llega correctamente a su destino. Esta estrategia puede ser adecuada para un servicio sin conexión y sin reconocimiento, pero es totalmente inadecuada para un servicio con conexión.

El método usual de asegurar una entrega es que el emisor tenga alguna forma de realimentación que le informe de lo que está ocurriendo en el otro extremo de la línea. Normalmente en los protocolos se plantea que el receptor mande al emisor una trama especial de control conteniendo un reconocimiento positivo o negativo sobre las tramas que va recibiendo. Si el emisor recibe un reconocimiento positivo de una trama sabe que esa trama ha llegado a destino correctamente. Por el contrario, un reconocimiento negativo indica que algo ha funcionado mal y la trama debe ser retransmitida.

Una complicación adicional es que existe la posibilidad de que una trama se pierda completamente por problemas en el hardware (por ejemplo, una ráfaga de ruido). En este caso el receptor no podrá reaccionar ya que no tendrá motivo para hacerlo. Debe quedar claro que en un protocolo en el que el emisor transmite la trama y entonces espera por el reconocimiento positivo o negativo podría bloquearse si la trama se pierde completamente.

Para evitar tal situación se introducen temporizadores en el nivel de enlace. Cuando el emisor transmite una trama arranca un temporizador. El temporizador se fija a un tiempo lo suficientemente grande para que la trama llegue a su destino, correctamente y que un reconocimiento venga del receptor. Normalmente la trama llegará correctamente al receptor y el reconocimiento volverá antes de que el temporizador finalice la cuenta, en cuyo caso se desconectará el temporizador. Sin embargo, si se pierde alguna de las tramas, la de datos o la de reconocimiento, el temporizador finalizará la cuenta, lo que avisará al emisor de que debe existir algún problema. La opción más lógica en este caso es retransmitir la trama. Ahora bien, si las tramas pueden ser retransmitidas varias veces existe el peligro de que el receptor acepte la misma trama más de una vez. Para prevenir que esto ocurra es generalmente necesario asignar números de secuencia a las tramas, para que el receptor pueda distinguir las tramas retransmitidas (duplicados) de las originales.

El manejo de los temporizadores y de la numeración de las tramas, así como garantizar que cada trama entregará su contenido al nivel de red de la máquina destino una y solo una vez (sin repeticiones) es una de las tareas importantes a desarrollar por el nivel de enlace de datos.

5.1.4 Control de Flujo

Otra cuestión que deben tener en cuenta en este nivel es qué hacer cuando el emisor quiere transmitir tramas con una frecuencia (velocidad) superior a la que puede aceptar el receptor. Esta situación puede darse fácilmente cuando, por ejemplo, el emisor es un proceso ejecutándose en una máquina potente o poco cargada, y el receptor es un proceso en una máquina lenta.

La solución más usual es introducir un mecanismo de control de flujo que impida transmitir al

emisor a una velocidad superior a la que puede aceptar el receptor. Este mecanismo de freno requiere generalmente algún tipo de realimentación que permita al procedimiento emisor conocer si el receptor está ya listo para aceptar otra trama.

Existen muchas posibilidades para realizar este control de flujo, pero la mayoría utilizan el mismo principio básico. El protocolo contiene reglas bien definidas acerca de cuando el emisor puede transmitir la siguiente trama. Estas reglas generalmente prohiben que se envíen tramas hasta que el receptor dé su permiso de forma implícita o explícita. Por ejemplo, cuando se establece una conexión el receptor puede decir:

> *Me puedes enviar n tramas ahora, pero después de haberlo hecho no envíes ninguna más hasta que yo te diga que continúes.*

Se estudiarán diversos mecanismos de control de flujo en este tema.

5.1.5 Gestión de Enlace

Otra de las funciones de la capa de enlace corresponde al manejo de la gestión de enlace. Con un servicio sin conexión esta gestión es mínima, pero para el caso de un servicio orientado a conexión es más compleja. Las conexiones se deben establecer y después liberar, las secuencias de números deben iniciarse y reiniciarse en caso de que sucedan errores, y así sucesivamente.

Además, la configuración del enlace se debe administrar; en el caso más sencillo, sólo un hilo físico se extiende entre dos máquinas. Sin embargo, es muy común encontrar que varias máquinas compartan el mismo canal. Una de ellas, tradicionalmente, es la **primera** (es decir, un ordenador), en tanto que las demás son **secundarias** (es decir, terminales pasivas).

La gestión de tráfico se logra cuando la primaria transmite una trama corta, denominada **sondeo**, a la primera secundaria, preguntando si tiene alguna información que transmitir. Si es así, el terminal envía dicha información; si no, la primera sondea a la siguiente secundaria. En otros sistemas, a los terminales se les permite que envíen información al ordenador, aun en ausencia del sondeo. Por último, en otros sistemas, como por ejemplo las redes tipo **LAN**, no hay primarios ni secundarios. Todas las estaciones son iguales y tienen los mismos derechos para acceder al canal. En cualquier caso, el tema completo de los primarios y secundarios versus corresponsales, es una solución que se lleva a cabo en la capa de enlace.

Como primera aproximación a los protocolos de enlace de datos vamos a ver tras protocolos de complejidad creciente. Pero antes de ver estos protocolos es interesante que se fijen ciertos conceptos acerca del modelo de comunicación. Para empezar, se asume que nivel físico, el nivel de enlace de datos y el nivel de red son procesos independientes que se comunican mediante paso de mensajes con el nivel vecino. Por ejemplo, los tres procesos pueden ejecutarse dentro del procesador de entrada/salida, o bien el nivel físico u de enlace de datos ser procedimientos que son llamados desde el proceso de nivel de red. En cualquier caso, tratar los tres niveles como procesos separados hacer la discusión conceptualmente más clara, y también sirve para enfatizar la independencia de los niveles.

También asumimos que la máquina *A* quiere enviar una larga cadena de datos a la maquina *B* utilizando un servicio fiable, servicio con conexión. Posteriormente consideraremos el caso en que *B* también quiera enviar datos simultáneamente a *A*. Se asume que se dispone de una fuente infinita de datos listos a ser enviados y que, por tanto, nunca se debe esperar a que los datos tengan que ser producidos. Cuando el nivel de enlace de *A* pide datos, el nivel de red se los proporciona sin demora. Esta limitación será eliminada en planteamientos posteriores.

Desde el punto de vista del nivel de enlace, el paquete que, a través del interfaz, le proporciona el nivel de red es contemplando como datos puros, cualquiera que sea el contenido del paquete. Cada bit del paquete debe ser entregado al nivel de red destino. El hecho de que el nivel de red interprete parte del paquete como cabecera de control no concierne al nivel de enlace.

Cuando el nivel de enlace acepta un paquete lo encapsula en una trama añadiendo una cabecera de enlace de datos. De esta forma, una trama consta de un paquete más cierta información de control, a la que denominaremos **cabecera**. La trama así formada se transmite al nivel de enlace destino. Asumiremos que disponemos de los procedimientos ***ToPhysicalLayer*** para enviar una trama y ***FromPhysicalLayer*** para recibir una trama. También asumiremos que el hardware transmisor calcula y añade el **crc**, así el nivel de enlace no tendrá que preocuparse de esto.

Inicialmente el nivel de enlace receptor no tiene nada que hacer. Simplemente está esperando a que algo ocurra. En los protocolos que se proponen de ejemplo se indica que el nivel de enlace está esperando mediante el procedimiento ***Wait(event)***. Este procedimiento sólo devolverá el control cuando haya ocurrido algo, como por ejemplo la llegada de una trama. La variable ***event*** indica qué es lo que ha ocurrido. El conjunto de posibles eventos difiere en los distintos protocolos que se plantean, debido a las distintas limitaciones que se asumen. Por lo tanto, se definen separadamente para cada protocolo. Hay que destacar que en una situación real el nivel de enlace no estaría en un bucle ocioso esperando un evento, sino que recibirá una interrupción lo que provocaría una suspensión del programa en ejecución para atender a la transmisión o recepción. No obstante, por simplicidad se ignoran estos detalles de actividades paralelas en el nivel de enlace y se asume que está dedicado todo el tiempo a un único canal.

Cuando una trama llega al receptor, el hardware calcula el **crc**. Si éste no es correcto si informa al nivel de enlace mediante un evento (***event = CheckSumErr***). El nivel de enlace no toma esta trama, puesto que como está dañada no puede hacer nada con ella. Si la trama llega correcta el nivel de enlace también es informado mediante ***event=FrameArrival***. De esta forma puede proceder a tomar la trama mediante el procedimiento ***FromPhysicalLayer***. Lo primero que hace el nivel de enlace cuando ha tomado la trama no dañada es chequear la información de control en la cabecera y si todo es correcto el paquete contenido en la trama se pasa al nivel de red mediante el procedimiento ***ToNetworkLayer***. Bajo ninguna circunstancia tendrá el nivel de red conocimiento de la información contenida en la cabecera de la trama. Esta información sólo concierne a los niveles de enlace.

Existe una buena razón por la cual el nivel de enlace nunca debe dar información de la cabecera de la trama al nivel de red: mantener los protocolos de enlace y de red completamente independientes. Sólo manteniendo esa independencia será posible modificar un protocolo o un formato de trama sin tener que corregir el software del nivel de red. Este razonamiento es válido para todos los niveles del modelo.

```
CONST LastBit = ...; {determina la longitud de trama}
      doomsday = false; {Utilizado para repetir eternamente}
TYPE bit = 0..1;
     SequenceNr = 0..Maxseq; {Utilizado para numerar las tramas}
     packet = packed array [0..LastBit] of bit; {Un paquete del nivel 3}
     FrameKind = {data,ack,nak}; {tipo de trama}
     frame = packed record {definición de la trama }
             kind : FrameKind;
             seq  : sequenceNr;
             ack  : SequenceNr;
             info : packet;
             end;

procedure wait (var event : EvType);
begin { Espera a que suceda un evento; retorna su tipo en event} end;
procedure FromNetworkLayer (var p: packet);
begin { Toma la información del nivel de red para transmitirla } end;
procedure ToNetworkLayer (p : packet)
begin { Entrega la trama recibida al nivel de red } end;
procedure FromPhysicalLayer (var r : frame);
begin { Obtiene una trama del nivel físico y la copia en r } end;
procedure ToPhysicalLayer (s : frame);
begin { Pasa la trama s al nivel físico para transmisión } end;
procedure StartTimer (k : SequenceNr);
Begin { Pone en marcha el temporizador } end;
procedure StopTimer (k : SequenceNr);
begin { Detiene  el temporizador e impide el evento Time Out}  end;
procedure StartAckTimer;
begin { Activa el temporizador auxiliar } end;
procedure StopAckTimer (k : SequenceNr);
begin { Detiene el temporizador auxiliar } end;
procedure EnableNetworkLayer;
begin { Activa el nivel de red permitiendo NetworkLayerReady } end;
procedure DisbleNetworkLayer;
begin { Inhibe el nivel de red } end;
procedure inc( var k : SequenceNr);
begin { Incrementa circularmente el número de secuencia k }
   if k < Maxseq then k:=k+1 else k=0
end;
```

La cita anterior muestra algunas declaraciones comunes a los protocolos que se discutirán mas tarde. Se definen cinco estructuras de datos: **bit**, **SequenceNr**, **packet**, **FrameKind** y **frame**. Un **bit** e 0 ó 1. Un **SequenceNr** es un entero corto utilizado para numerar las tramas. La constante **MaxSeq** se define en cada protocolo. Un **packet** es la unidad de información intercambiada entre el nivel de red y el de enlace en la misma máquina, y entre los niveles de red de ambas máquinas. En nuestro modelo siempre contiene exactamente **LastBit+1** bits, ya que se cuenta desde cero. En un planteamiento más realista sería de longitud variable entre 0 y un máximo de **LastBit+1** bits.

Un **frame** (trama) está compuesta de cuatro campos: **kind**, **seq**, **ack**, e **info**. Los tres primeros contiene información de control y el último contiene los datos a ser transmitidos, es decir, el paquete. Los campos de control se denominan de forma colectiva como **cabecera de trama**. El campo **kind** indica si hay o no datos en la trama, porque algunos protocolos distinguen entre tramas que contienen sólo información de control de aquellas que contienen también datos. Los campos **seq** y **ack** se utilizan como números de secuencia y de reconocimiento, respectivamente; su uso será descrito con detalle posteriormente. El campo **info** contiene un paquete; el campo **info** de una trama de control no es utilizado. En una implementación más realista, se utilizaría un campo **info** de longitud variable y se omitiría en las tramas de control.

Es importante insistir en la relación entre el paquete y la trama. El nivel de red construye un paquete tomando un mensaje del nivel de transporte y añadiendo una cabecera de red. Este paquete se pasa al nivel de enlace que lo deposita en el campo **info** de la trama. Cuando la trama llega a su destino, el nivel de red puede actuar como si las dos máquinas pudieran intercambiar los paquetes directamente.

También se han listado una serie de procedimientos. Son rutinas de librería, utilidades que usaremos en los protocolos y cuya implementación no nos interesa. El procedimiento **wait** es un bucle de espera a que ocurra algún evento como ya se ha mencionado anteriormente. Los procedimientos **ToNetworkLayer** y **FromNetworkLayer** los utiliza el nivel de enlace para el tráfico de paquetes con el nivel de red. Los procedimientos **FromPhysicalLayer** y **ToPhysicalLayer** son utilizados para pasar tramas entre el nivel de enlace y el nivel físico.

En la mayoría de los protocolos se asume un canal de comunicación no fiable que puede perder tramas enteras. Para poder controlar este tipo de fallos, el nivel de enlace que envía datos debe arrancar un temporizador interno o reloj cuando manda la trama. Si no hay

respuesta pasado un predeterminado intervalo de tiempo, el temporizador finaliza la cuenta y el nivel de enlace recibe una señal de interrupción que le indica que ya ha esperado demasiado tiempo por la respuesta.

En los protocolos que se proponen esta circunstancia se plantea permitiendo que el procedimiento **wait** devuelva el evento *Fuera de tiempo*, es decir **event = TimeOut**. Los procedimientos **StartTimer** y **StopTimer** se utilizan para conectar o desconectar el temporizador, respectivamente. Un **TimeOut** sólo es posible cuando el temporizador está conectado. Esta permitido hacer **StartTimer** mientras el temporizador está activo. Tal llamada reinicia la cuenta del temporizador y deberá pasar todo el intervalo predefinido de tiempo antes de producirse el **TimeOut**.

Los procedimientos **StartAckTimer** y **StopAcktimer** se utilizarán para el control de un temporizador auxiliar utilizado para generar reconocimientos (tramas de sólo reconocimiento) bajo ciertas condiciones.

Los procedimientos **EnableNetworkLayer** y **DisbleNetworkLayer** se utilizan en los protocolos más sofisticados para realizar un control de flujo entre el nivel 2 y 3. Cuando el nivel de enlace habilita al nivel de red, el nivel de red puede interrumpir al nivel de enlace mediante **event=NetworkLayerReady**. Cuando el nivel de red está deshabilitado, no puede producirse este evento. Manejando la habilitación y deshabilitación del nivel de red, el nivel de enlace puede evitar que el nivel de red desborde con demasiados paquetes la memoria donde el nivel de enlace los almacena temporalmente.

Los números de secuencia de las tramas está siempre en el rango **0** a **Maxseq** (inclusive), donde **MaxSeq** es diferente para los distintos protocolos. Es frecuentemente necesario avanzar el número de secuencia en 1 de forma circular (alcanzando el valor **MaxSeq**, el siguiente es de nuevo **0**). El procedimiento **inc** realiza este incremento. Simplemente, **inc(i)** es **i:= (i+1) MOD MaxSeq**.

Las declaraciones comentadas son parte de los protocolos que a continuación se plantean. Para ahorrar espacio y proporcionar una referencia adecuada han sido extraídos y listados aparte, pero conceptualmente deben ser mezclados con el protocolo adecuado.

5.2.1 Protocolo utópico

Como primer ejemplo vamos a considerar un protocolo lo más simple posible. Los datos se transmiten en un solo sentido. Los dos niveles de red, el emisor y el receptor, están siempre listos. El tiempo de procesamiento de la trama se desprecia. Se dispone de memoria infinita. Y lo mejor de todo, el canal de comunicación entre los dos niveles de enlace es perfecto, nunca deteriora ni pierde tramas.

```
{ El protocolo 1 (utópico) permite la transmisión de datos en un solo
sentido, desde el receptor al emisor. Asumimos que el canal de
comunicaciones se encuentra libre de errores, y que el receptor es capaz
de procesar las entradas a una velocidad infinita. Por tanto, el
transmisor únicamente ejecuta un bucle enviando datos por la línea tan
rápidamente como puede. }

type
   EvType = (FrameArrival);

procedure sender1;
var
   s: frame;                      { Buffer para la trama de salida }
   buffer: packet                 { Buffer para el paquete de salida}
begin
   repeat
      FromNetworkLayer(buffer);   { Tomamos paquete del nivel de red }
      s.info:=buffer;             { Lo copiamos en s para la transmisión }
      ToPhysicalLayer(s)          { Lo enviamos }
   until dommsday
end; {sender1}

procedure receiver1;
var
   r: frame;                      { Buffer para la trama que llega }
   event : EvType                 { Necesario para llamar a wait }
begin
   repeat
      wait(event);                { La única posibilidad es FrameArrival }
      FromPhysicalLayer(r);       { Tomamos la trama del nivel físico }
      ToNetworkLayer(r.info)      { Pasamos el paquete al nivel de red }
   until doomsday
end; {receiver1}
```

El protocolo consta de dos procedimientos, uno emisor *Sender1* y el otro receptor *Receiver1*. El emisor se ejecuta en el nivel de enlace de la máquina emisora y el receptor en el nivel de enlace de la máquina receptora, No se utilizan los números de secuencia de las tramas ni los reconocimientos, así pues la variable **MaxSeq** no es necesaria. El único evento posible es **frameArrival**, es decir, llegada de una trama no dañada.

El emisor es un bucle infinito implementado como *repeat until doomsday* (repetir hasta el día del Juicio Final), que se limita a mandar datos al canal tan deprisa como puede. En el interior del bucle infinito se realizan tres acciones:

- Tomar un paquete del nivel de red (que siempre está listo para mandar paquetes).
- Construir la trama, utilizando la variable *s*.
- Mandar la trama al nivel físico que la pondrá en el canal de comunicación.

De la trama sólo se utiliza el campo **info** en este protocolo porque los otros campos(cabecera de trama) tienen como misión realizar control de flujo, errores, etc, y en el entorno idílico que se plantea nada de esto es necesario.

El procedimiento receptor es igual de simple. Inicialmente está esperando que ocurra algo. Lo único que puede suceder es que llegue una trama en buen estado. Cuando eso ocurre, el procedimiento **wait** devuelve el control al programa con **event=FrameArrival**. Dado que es el único evento posible, se ignora el comprobar el valor de **event**. El procedimiento **FromPhysicalLayer** toma la trama de la memoria del dispositivo físico de entrada/salida y la pone en la variable *r*. Finalmente el campo de datos de la trama, es decir, el paquete, se pasa al nivel de red y el nivel de enlace vuelve a esperar a que llegue la siguiente trama.

5.3.1 Protocolos de STOP & WAIT

Uno de los planteamientos más irreales del protocolo *Utopía* (al que en lo sucesivo denominaremos como *protocolo1*) es asumir que el receptor puede procesar a suficiente velocidad todas las tramas que le mande el transmisor (o como hipótesis equivalente, que el receptor dispone de un buffer infinito para ir acumulando las tramas que llegan u no puede

procesar).

Todavía se asume que el canal de comunicación está libre de errores y que el tráfico de datos es símplex (los datos sólo evolucionarán en un sentido, de emisor a receptor, pero existirá información de control de receptor a emisor).

El problema a resolver es como evitar que el emisor desborde al receptor con datos (tramas), es decir, se trata de que el emisor envíe tramas a una velocidad a la que el receptor las pueda procesar. En esencia, si el receptor necesita un tiempo *t* para ejecutar **FromPhysicalLayer + ToNetworkLayer** el emisor debe enviar tramas a un velocidad media menor o igual a una trama por cada *t* unidades de tiempo. Si además se asume que no existen múltiples buffers con una gestión de encolado de tramas por parte del hardware o el sistema operativo, el emisor nunca deberá transmitir una nueva trama hasta que la vieja haya sido tomada del (único) buffer por **FromPhysicalLayer**. De no ser así, la trama nueva se reescribirá sobre la vieja.

En algunas circunstancias, como podría ser el caso de transmisión síncrona y un nivel de enlace dedicado totalmente a procesar una única línea de entrada, se podría plantear como solución que en procedimiento emisor se insertara un retardo que rebajara la velocidad de envío de tramas lo suficiente para no desbordar al receptor. Sin embargo, lo más usual es que el nivel de enlace tenga varias líneas que atender y el intervalo de tiempo entre la llegada de una trama y su procesamiento sea variable. Para que sea posible utilizar este método es necesario considerar el caso peor (el tiempo entre la llegada de la trama y su procesamiento es máximo). Por lo que es necesario introducir un retardo igual al máximo para garantizar que nunca se desbordará al receptor. Esta estrategia, como puede verse, resulta muy poco eficiente.

Una solución más general al problema consiste en proporcionar una realimentación al emisor. Después de mandar el paquete al nivel superior, el nivel de enlace receptor podría mandar una trama vacía de datos como señal de que ya ha procesado la información recibida e indicando al emisor que puede mandar la siguiente trama. De esta forma la estrategia consiste en que el emisor mande una trama y espere hasta recibir la trama de reconocimiento (la trama vacía) para mandar la siguiente trama.

A los protocolos en los cuales el emisor manda una trama y espera por un reconocimiento antes de proseguir se les denominan **stop & wait** (para y espera). Al igual que en el *protocolo1*, el emisor comienza por la búsqueda de un paquete del nivel de red, construye con él una trama y la manda al nivel físico. Pero a diferencia del *protocolo1*, el emisor debe esperar hasta que llegue una trama de reconocimiento antes de repetir el proceso. El nivel de enlace emisor no necesita siquiera inspeccionar el contenido de la trama de reconocimiento, le basta con saber que ha llegado para interpretar que puede continuar mandando la siguiente trama.

La única diferencia entre *receiver1* y *receiver2* es que después de entregar el paquete al nivel de red *receiver2* manda una trama de reconocimiento al emisor. Dado que lo único importante es la llegada, en sí misma, de la trama de reconocimiento en el emisor y no su contenido, el receptor no necesita poner ninguna información particular en ella.

Si bien, en este ejemplo, el tráfico de los datos es símplex (de emisor a receptor) las tramas viajan en ambos sentidos. Consecuentemente, el canal de comunicación entre los dos niveles de enlace debe permitir la transferencia de información bidireccional. Siendo más precisos, este protocolo implica una alternancia de flujo de información: primero el emisor manda una trama; después el receptor manda una trama; a continuación el emisor manda la siguiente trama, y el receptor contesta con otra trama... y así continuamente. Para este caso un canal *Half-duplex* (semi-duplex) es suficiente.

```
{ Protocolo 2 (Stop & Wait) suministra un canal unidireccional desde el
emisor al receptor. Dicho canal es libre de errores como el protocolo 1.
Sin embargo el receptor tiene un buffer limitado. Como la velocidad de
procesamiento se sigue considerando infinita, el protocolo debe evitar que
el receptor se desborde al recibir las tramas que genera el emisor. }

type EvType = (FrameArrival);

procedure sender2;
var s:frame;
buffer :packet;
event : EvType;
begin
   repeat
      FromNetworkLayer(buffer);
      s.info := buffer;
      ToPhysicalLayer(s);
      wait(event);
   until doomsday
end; {sender2}

procedure receiver2;
var s: frame;
event :EvType;
begin
   repeat
      wait(event);
      FromPhysicalLayer(r);
      ToNetworkLayer(r.info);
      ToPhysicalLayer(s);
   until doomsday
end; {receiver2}
```

5.3.2 Protocolo simple para canal con ruido

Consideremos ahora la situación más realista de que el canal produce errores. Las tramas pueden ser dañadas o perdidas. Sin embargo, asumiremos que si la trama es dañada en la transmisión, el equipo hardware lo detectará *siempre* cuando compruebe el *CheckSum (por ejemplo* CRC). Es necesario destacar que si la trama fuera dañada de tal forma que el *CheckSum* no lo detectara, este protocolo (y cualquier otro) fallaría y permitiría que se entregara al nivel de red un paquete con contenido erróneo.

A primera vista puede parecer que se podría trabajar con una pequeña variación del *protocolo2*. El emisor mandaría una trama y el receptor sólo mandaría una trama de reconocimiento si los datos fueran correctamente recibidos. Si una trama llega dañada al receptor, sería descartada. Después de un cierto tiempo (TimeOut) el emisor la volvería a transmitir. Este proceso se repetiría hasta que la trama llegara intacta.

Esta estrategia contiene un error fatal. Para ver qué es lo que va mal hay que recordar que misión de los niveles de enlace es proporcionar una comunicación transparente y libre de errores entre los niveles de red. El nivel de red en la máquina *A* entrega una serie de paquetes a su nivel de enlace, el cual debe asegurar que una serie idéntica de paquetes es entregada al nivel de red en la máquina *B*, por medio del nivel de enlace de *B*. En concreto, el nivel de red en *B* no tiene manera de saber si un paquete se ha perdido o duplicado, así que el nivel de enlace debe garantizar que ninguna combinación de errores de transmisión, sea la que sea, pueda hacer que se entregue un paquete duplicado al nivel de red.

Consideremos la siguiente secuencia de eventos:

1) El nivel de red en A entrega un paquete 1 a su nivel de enlace. La trama con el paquete se recibe en B correctamente y el paquete se pasa al nivel de red en B. B manda un reconocimiento a A.

2) La trama de reconocimiento se pierde por un error (por ejemplo, un corte temporal de la línea, un mal contacto, etc).

3) Vence el tiempo de espera (TimeOut) en el nivel de enlace de A. No habiendo recibido el reconocimiento, A asume que su trama de datos se ha perdido o dañado y envía otra vez una trama conteniendo el paquete 1.

4) La trama duplicada llega correcta al nivel de enlace B y éste pasa por segunda vez el paquete 1 a su nivel de red. En este caso, se recibirán datos duplicados por el nivel de red, por lo que el protocolo fallará.

Claramente lo que se necesita es una manera de que el receptor pueda distinguir entre una trama nueva y una trama duplicada. La forma más lógica de realizarlo es que el emisor ponga un número de secuencia en un campo de cabecera de cada trama que envía. Así, el receptor puede chequear el número de secuencia de cada trama que llega y ver si se trata de una trama nueva o duplicada.

Dado que es deseable un campo de cabecera lo más pequeño posible la pregunta es: ¿Cuál es el número mínimo de bits necesarios para el número de secuencia?. La única ambigüedad que puede plantearse en un protocolo *Stop&Wait* es entre una trama **m** y la siguiente **m+1**. Si la trama **m** se pierde o se daña el receptor no la reconocerá. Por vencimientos del TimeOut se repetirá las veces que haga falta hasta que por fin llegue correcta al receptor. Cuando llegue correcta, el receptor mandará un reconocimiento al emisor. Es aquí donde puede plantearse el problema. Dependiendo de si el reconocimiento llega al emisor correctamente o no, el emisor deberá mandar **m+1** o **m**, respectivamente.

Basta utilizar un campo de 1 bit para los número de secuencia (0 ó 1). En un determinado instante, el receptor esperará un determinado número de secuencia. Cualquier trama que llegue con el número de secuencia erróneo será rechazada como duplicada. Cuando llegue una trama con el número de secuencia esperado, será aceptada, se pasará su contenido (el paquete) al nivel de red y el número de secuencia esperado se incrementará módulo 2.

```
{Protocolo 3. Permite anviar datos de forma bidireccional entre emisor y
receptor sobre un canal de comunicaciones con ruido. Por tanto, se pueden
perder o dañar las tramas. }

Const MaxSeq = 1;
type EvType = (FrameArrival, ChksumErr, TimeOut);

procedure sender3;
var NextFrameToSend; SeqNumberNr;
    s: frame;
    buffer: packet;
    event: Evtype;
begin
   NextFrameToSend := 0;
   FromNetworkLayer(buffer);
   repeat
      s.info := buffer;
      s.seq := NextFrameToSend;
      ToPhysicalLayer(s);
      StartTimer(s.seq);
      wait(event);
      if event = FrameArrival then
         begin
            FromNetwrokLayer(buffer);
            inc(NextFrameToSend);
         end
   until doomsday
end; {sender3}

procedure receiver3;
var FrameExpected; SeqNumberNr;
    r, s ; frame;
    event: EvType;
begin
   FrameExepcted := 0;
   repeat
      wait(event);
      if event = FrameArrival then
         begin
            FromPhysicalLayer(r);
            if r.seq = FrameExpected then
               begin
                  ToNetworkLayer(r.info);
                  inc(FrameExpected);
               end;
            ToPhysicalLayer(s);
         end
   until doomsday;
end; {receiver3}
```

El protocolo anterior es un ejemplo de protocolo *Stop & Wait* que maneja pérdidas de tramas, utilizando un temporizador que mide una cuenta de TimeOut. Se requiere que el intervalo de TimeOut sea lo suficientemente largo para evitar TimeOuts prematuros. Si en el emisor vence el tiempo de TimeOut demasiado pronto mientras el reconocimiento está llegando, se enviará un duplicado. Cuando el reconocimiento finalmente llegue, el emisor creerá erróneamente que la trama recién mandada es la reconocida cuando en realidad la trama recién mandada va a

generar, si no hay errores, otro reconocimiento extra. Si la próxima trama mandada se pierde y el reconocimiento extra llega correctamente, el emisor interpretará que se reconoce la trama perdida y no volverá a retransmitirla. En estas condiciones el protocolo falla.

En los siguientes protocolos veremos que las tramas de reconocimiento contendrán información para prever el problema de que el reconocimiento de una determinada trama se asocie a otra trama. En este protocolo asumimos una alternancia estricta de emisor y receptor, siendo los reconocimientos genéricos.

En *protocolo3* difiere de sus predecesores en el uso de las variables **NextFrameToSend**, con la que va numerando las tramas que se mandan, y la variable **FrmaExpected** con la que controla cual es el número de trama que espera.

Después de transmitir una trama, el emisor comienza a contar un intervalo de tiempo. Si todo va bien la cuenta se detendrá antes de llegar al final del intervalo (TimeOut). El valor del intervalo de tiempo a contar o TimeOut debe ser, como hamos comentado, suficiente para permitir que la trama llegue al emisor. Solamente cuando se haya consumido todo el intervalo de tiempo se podrá asumir que la trama transmitida o su reconocimiento se ha perdido y se debe mandar una trama duplicada.

Después de transmitir una trama y arrancar el temporizador el emisor espera a que ocurra algo. Existen tres posibilidades: llega una trama de reconocimiento sin daño, llega una trama de reconocimiento dañada o vence el TimeOut. Si llega un reconocimiento en buen estado, el emisor toma el siguiente paquete del nivel de red y lo pone en una zona buffer de memoria sobre escribiendo el paquete anterior. También avanza (incremento módulo 2) el número de secuencia de la trama. Si llega un reconocimiento dañado o si no llega ningún reconocimiento y se produce un TimeOut, no se cambia el contenido del buffer ni el número de secuencia, así que se mandará un duplicado.

En el receptor, cuando llega una trama sin daño se comprueba se número de secuencia par ver si es un duplicado. Si no lo es, se pasa el paquete contenido al nivel de red y se genera un reconocimiento. No se pasa al nivel de red nada que venga en tramas duplicadas o dañadas.

5.4.1 Técnica de ventana deslizante

En los protocolos anteriores las tramas de datos se transmitían en un solo sentido. No obstante, la mayor parte de las veces se necesita una transmisión de datos en ambos sentidos. Una forma de disponer de una transmisión de datos *Full-duplex* sería tener dos canales de comunicación *simplex*, uno en cada sentido. Si procedemos de esta manera tendríamos dos circuitos separados. Cada uno de ellos constaría de una canal de oda (para datos) y otro de vuelta (para reconocimientos). Por el circuito físico de *A* a *B* se mandarían entremezcladas las tramas de datos y los reconocimientos para *B*. Con este planteamiento, para distinguir las tramas de datos de las de reconocimiento es necesario un campo **kind** (tipo) en la cabecera de la trama que permita al receptor identificar el contenido.

Es posible mejorar la eficiencia de la comunicación mediante el siguiente método: Puesto que la trama de reconocimiento es una trama sin datos asociados, mandaremos en una sola trama los datos y los reconocimientos necesarios. Cuando llega una trama, en vez de mandar inmediatamente una trama de control, el receptor espera hasta que tenga que mandar una trama de datos. El reconocimiento se añade en un campo de cabecera a la trama saliente. Esta técnica de retrasar temporalmente el envío de los reconocimientos para que puedan ser insertados en la primera trama de salida se conoce como **piggybacking**.

La ventana principal de *piggybacking* frente a utilizar tramas separadas de reconocimiento es el mejor uso de la capacidad del canal. Un campo **ack** (de reconocimiento) en la cabecera de una trama tiene el coste de unos pocos bits, mientras que una trama de reconocimiento precisa de

una cabecera, el campo reconocimiento objeto de toda la trama y un campo de *CheckSum*. Además mandando menos tramas conseguimos reducir el número de interrupciones por el evento **FrameArrival** y, según implementaciones, reducimos también el número de buffers en el receptor. En el próximo protocolo veremos, el campo de reconocimiento sólo ocupa un bit en la cabecera. Rara vez tiene un coste mayor que unos pocos bits.

Por el contrario, el **piggybacking**, también plantea un inconveniente que no encontramos en la estrategia de enviar tramas de reconocimiento. El tiempo que debe esperar el nivel de enlace receptor para enviar el reconocimiento depende de la actividad de su nivel de red. Si el nivel de enlace espera un tiempo mayor que el que permite el **timeOut** del emisor, la trama se retransmitirá, lo cual no es deseable. El nivel de enlace receptor no tiene forma de saber cuando va a enviar su nivel de red un paquete para transmitir. Por lo tanto, no le es posible decidir si envía una trama de sólo reconocimiento o si, por el contrario, espera el paquete del nivel de red para realizar un **piggybacking**. Para solucionar este problema se suele implementar una solución de tipo intermedio: el nivel de enlace, una vez recibe la trama de datos correctamente, espera un intervalo de tiempo determinado a que su nivel de red desee enviar un paquete. Si tras ese periodo no se produce esta circunstancia, el nivel de enlace envía una trama de sólo reconocimiento. Si el nivel de red sí envía un paquete, el reconocimiento se incluye en la trama de este paquete.

El *protocolo4*, además de ser bidireccional, soluciona un problema que tiene el *protocolo3*; éste se puede bloquear si el tiempo de TimeOut es pequeño y provoca retransmisiones prematuras.

Todos los protocolos a partir de éste (inclusive) son robustos y libres de error frente a cualquier combinación de tramas dañadas o duplicadas y TimeOuts prematuros. Los tres protocolos que veremos a continuación pertenecen a la clase denominada de **Slidding Window (Ventana deslizante)**.

En todos los protocolos de ventana deslizante cada trama que se manda contiene un número de secuencia en el rango de 0 hasta un máximo de **MaxSeq**. El máximo es normalmente $\mathbf{2^n-1}$ para que el número ocupe justamente un campo de **n** bits. Para implementar un protocolo **Stop & wait** con la técnica de ventana deslizante bastaría con hacer **n=1**, con lo que restringimos los números de secuencia a 0 y 1. Si utilizamos un valor de **n** mayor que 1 estamos implementando un protocolo **pipeline**, los cuales estudiaremos posteriormente.

La esencia de un protocolo de ventana deslizante es que en todo instante el emisor mantiene una lista consecutiva de números de secuencia, correspondiente a las tramas que ha mandado pero de las que no ha recibido el reconocimiento. Estas tramas se dice que están en la **ventana de emisión**. De forma similar, el receptor mantiene una **ventana de recepción** correspondiente a las tramas que le es permitido aceptar. La ventana de transmisión y la de recepción no tienen porqué tener los mismos límites superiores e inferior, ni tampoco el mismo tamaño.

Si bien estos protocolos dan al nivel de enlace gran libertad sobre el orden en que envía y recibe tramas, debe quedar claro que el orden en que el nivel de enlace receptor da los paquetes al nivel superior debe ser estrictamente el mismo en que el nivel de red en el emisor se los pasó al nivel de enlace emisor.

Cuando el nivel superior entrega un nievo paquete al nivel de enlace, se le asigna el siguiente número de secuencia al mayor en la ventana de emisión, y el límite superior de la ventana avanza para contener este nuevo número de secuencia. Cuando llegan reconocimientos, el límite inferior de la ventana avanza, liberando de la ventana los números de las tramas reconocidas.

Las tramas en la ventana de transmisión deben ser retenidas en memoria por el emisor hasta que sena reconocidas por si precisan ser retransmitidas. Por ejemplo, si el tamaño máximo de la ventana de transmisión es **n**, el emisor necesita **n** buffers para almacenar las tramas no reconocidas. Si la ventana alcanza su tamaño máximo, el nivel de enlace deberá indicar al nivel superior que no puede aceptar más paquetes hasta que disponga de algún buffer, cosa que ocurrirá cuando le reconozcan alguna de las tramas que ya ha mandado.

La ventana de recepción, en el nivel de enlace del receptor, indica las tramas que el receptor está dispuesto a aceptar. Cualquier trama que llegue al receptor con un número de secuencia fuera de esta ventana será rechazada. Si llega una trama cuyo número de secuencia es igual al límite inferior de la ventana de recepción, el contenido de la trama se pasa al nivel superior, se gira **toda la ventana** una posición y se manda reconocimiento de la trama aceptada. Al contrario que la ventana de emisión, que tiene tamaño variable, la ventana de recepción tiene tamaño constante. Es importante observar que una ventana de recepción de tamaño 1 significa que el nivel de enlace sólo aceptará tramas que le lleguen en orden. Si la ventana de recepción tiene un tamaño mayor, podrá aceptar tramas fuera de orden, pero en cualquier caso el nivel de enlace debe entregar los paquetes al nivel superior en estricto orden de precedencia.

FALTA ejemplo gráfico de ventana de un bit

5.4.2 Protocolo de ventana deslizante de un bit

Vamos a ver el caso más simple de protocolo de ventana deslizante, que es el de tamaño máximo de ventana de emisión igual a uno y ventana de recepción también uno. Tal y como se comentó con anterioridad, esto significa que únicamente podrá enviarse una trama, no estando permitido enviar la siguiente hasta la llegada del reconocimiento correspondiente. Asimismo, tampoco se permite la aceptación de más de una trama, es decir, en cada momento el receptor espera una trama concreta, y en caso de que llegara otra sería ignorada. Este planteamiento se refinará en protocolos sucesivos.

Por lo que acabamos de decir resulta evidente que se trata de un protocolo del tipo *Stop & Wait*, ya que el emisor manda una trama y no puede enviar la siguiente hasta recibir el reconocimiento.

A continuación podemos ver el algoritmo que implementa este protocolo:

```
{Protocolo 4 (Ventana deslizante). Este protocolo es bidireccional, más
robusto que el protocolo 3. Puede soportar cualquier combinación de
errores y timeouts sin perder o duplicar paquetes del nivel 3. }

procedure protocol4:
var NextFrameToSend: SeqNumberNr;
    FrameExpected: SeqNumberNr;
    r, s: frame;
    buffer: packet;
    event: EvType;
begin
   NextFrameToSend := 0; FrameExpected := 0;
   FromNetworkLayer(buffer);
   s.info := buffer; s.seq := NextFrameToSend;
   s.ack := 1 - FrameExpected;
   ToPhysicalLayer(s);
   StartTimer(s.seq);
   repeat
      wait(event);
      if event = FrameArrival then
         begin
            FromPhysicalLayer(r.info);
            if r.seq = FrameExpected then
               begin
                  ToNetworkLayer(buffer);
                  inc(NextFrameToSend);
               end;
            if r.ack = NextFrameToSend then
               begin
                  FromNetworkLayer(buffer);
                  inc(NextFrameToSend);
               end
         end
      s.info := buffer;
      s.seq := NextFrametoSend;
      s.ack := 1 - FrameExpected;
      ToPhysicalLayer(s);
      StartTimer(s.seq);
   until doomsday;
end; {protocol4}
```

La variable ***NextFrameToSend*** indica cual es la trama que el emisor está tratando de mandar. Igualmente, ***FrameExpected*** indica al receptor cual es la trama que está esperando. En ambos casos, 0 y 1 son las únicas posibilidades.

El algoritmo que se muestra se ejecuta tanto en la máquina emisora como en la receptora, si bien uno de los dos niveles de enlace comienza primero a transmitir. En otras palabras, sólo uno de los dos programas debe contener los procedimientos ***ToPhysycalLayer*** y ***StartTimer*** fuero del bucle principal. En caso de que los dos niveles de enlace ejecutaran a la vez el mismo algoritmo propuesto se plantea una situación peculiar. Se deja como ejercicio propuesto el plantear esta situación.

Suponiendo que un máquina inicia el protocolo, ésta toma un paquete del nivel superior, construye una trama con él y la manda. Cuando esta trama llega al receptor, este comprueba si se trata de un duplicado de forma análoga a como lo hacía el *protocolo3*. Si la trama es la esperada, su contenido se pasa al nivel superior y se hace evolucionar la ventana del receptor, de forma que si se ha recibido la trama 0 se pasa a esperar la 1, y viceversa.

El campo de reconocimento contiene el número de la última trama recibida sin error. Si ese número coincide con el que el transmisor tiene en su ventana de emisión, éste hace evolucionar el límite inferior de la ventana con lo que la cierra. Si no coincide, la ventana no evoluciona, el temporizador de TimeOut sigue contando y cuando transcurra el tiempo de TimeOut se retrasmitirá la trama.

A continuación estudiaremos el comportamiento de este protocolo frente a errores. Puesto que este protocolo comprende el *protocolo3* visto con anterioridad, es capaz de reaccionar frente a cualquier secuencia de errores que éste último solventara. Por tanto, trataremos únicamente el caso de que el tiempo de TimeOut fuera excesivamente corto, que es el problema que planteaba el anterior protocolo.

Asumamos que la máquina *A* está tratando de mandar la trama 0 a *B* y que *B* intenta enviar la trama 0 a *A*. Supongamos que *A* manda la trama a *B* pero que el intervalo de TimeOut es excesivamente pequeño, por lo que *A* manda varias tramas repetidas con *seq=0* y *ack=1*.

Cuando llegue la primera trama válida a *B*, éste la aceptará y pondrá ***FrmaExpected*** a 1 (ha cambiado la ventana de recepción). Todas las demás tramas mandadas por *A* serán rechazadas porque sus números de secuencia ya no estarán en la ventana de recepción. Por otro lado, dado que todas las tramas duplicadas llevan el reconocimiento *ack=1* y *B* está esperando el reconocimiento de 0, *B* no tomará un nuevo paquete de su nivel de red.

Cada vez que llega una trama duplicada, *B* manda una trama que contiene *seq=0* y *ack=0*. Eventualmente, uno de estos paquetes llegará a *A* haciendo que *A* mande un nuevo paquete. Ninguna combinación de TimeOuts prematuros o tramas perdidas puede producir un mal funcionamiento o bloqueo del protocolo.

5.3.2 Protocolo de repetición no selectiva o “vuelta atrás”

Hasta ahora, en los protocolos anteriores, no hemos considerado el tiempo de transmisión de una trama ni el de la llegada del reconocimiento. En el caso que este tiempo sea bajo, los protocolos anteriores cumplen con su misión. Sin embargo, en las comunicaciones a largas distancias o vía satélite la suma de tiempos entre la emisión de una trama y la llegada del reconocimiento puede ser muy alto. Bajo estas circunstancias, el tiempo de propagación del medio en cuestión limitaría notablemente el rendimiento de la transmisión, ya que el emisor perdería mucho tiempo esperando la aceptación de la trama. Durante este tiempo el emisor permanece bloqueado, dejando el canal muy desaprovechado.

Para solventar este problema de eficiencia sobre el canal de comunicación podemos permitir que el emisor envíe *n* tramas sin esperar el reconocimiento de la primera. Esto implica un protocolo con una ventana de tamaño *n*. Por lo tanto, el emisor podrá enviar *n* tramas sin ser reconocidas antes de bloquearse. A este esquema se le denomina **Pipelining**. Como puede apreciarse, es similar al protocolo 4, pero manteniendo una ventana de emisión de tamaño *n*. Esto implica dotar al nivel de enlace transmisor de un conjunto de buffers que contengan las tramas transmitidas, de tal modo que le permitan retransmitir algún paquete si fuera necesario. Para evitar que la capacidad del emisor se vea desbordada por el nivel de red es necesario impedir al nivel de red que deposite paquetes cuando no existe espacio para ellos. Esto se

consigue mediante los procedimientos ***EnableNetworkLayer*** y ***DisableNetworkLayer***.

La utilización de esta técnica sobre un canal con ruido o poco fiable nos presenta dificultades. Si una trama es dañada o perdida, las tramas posteriores pueden llegar al receptor antes incluso que el emisor pueda conocer este evento. El receptor debe ignorar la trama errónea, pero ¿Qué debe hacer con las tramas correctas pero con numeración posterior a la dañada?. Hay que tener en cuenta que el nivel de enlace receptor debe pasar los paquetes a su nivel de red en el mismo orden en el que el nivel de enlace de la máquina emisora los recibió del suyo.

Existen dos enfoques a este problema. Los protocolos 5 y 6 reflejan cada una de ellas. L a primera, denominada *vuelta atrás*, soluciona el problema descartando todas las tramas que siguen a la dañada, y no enviando reconocimiento. Es un esquema análogo a lo que hace el receptor del protocolo de ventana deslizante de tamaño 1.

La segunda estrategia para manejar errores en estas condiciones es la llamada *retransmisión selectiva*. Consiste en que el nivel de enlace del receptor almacena las tramas que lleguen correctamente después de detectar una trama errónea. Cuando el emisor se da cuenta de que algo va mal, retransmite sólo la trama dañada y no toda la sucesión como en el caso anterior. Si la retransmisión tiene éxito, el nivel de enlace receptor tendrá muchas tramas en secuencia y además correctas, por lo que podrá pasar muchos paquetes consecutivos a su nivel de red e inmediatamente mandar un reconocimiento con el número de trama recibido más alto.

Esta estrategia corresponde con una ventana de recepción con capacidad de recoger tramas fuera de orden, es decir, necesitamos una ventana con un tamaño mayor de 1. Estas tramas fuera de orden serán almacenadas hasta conseguir una secuencia completa.

Puesto que el protocolo5 no almacena las tramas que llegan tras el error, el emisor deberá retransmitir todas cuando ese error se resuelva. Cuando un reconocimiento llega para la trama *n*, automáticamente las tramas anteriores son reconocidas. Esto permite que, en caso de pérdida de reconocimiento, podamos asegurar que todas las tramas recibidas van a ser reconocidas. Cuando llega un reconocimiento, el nivel de enlace puede liberar los buffers en los cuales había almacenado los paquetes transmitidos anteriormente, y permitir a su nivel de red que deposite nuevos paquetes para su envío.

Otra cosa a tener en cuenta es el número máximo de tramas pendientes de reconocimiento. Supongamos que disponemos de un número de secuencia en el rango 0..**MaxSeq**, es deicr, **MaxSeq+1** números de secuencia. Puesto que la efectividad del método es mejor cuanto mayor sea la ventana de transmisión, asumiremos que podemos enviar todas las tramas antes de esperar reconocimiento. Sin embargo, no es posible utilizar una ventana de emisión de tamaño **MaxSeq+1**. Para demostrar esta afirmación pondremos un ejemplo con **MaxSeq=7**; supondremos la siguiente secuencia de enventos:

- El emisor manda 8 tramas numeradas de 0 a 7.
- El receptor recibe correctamente estas tramas y envía un reconocimiento de las última, es decir, de la 7.
- El emisor recibe el reconocimiento de la trama 7.
- El emisor envía otras 8 tramas, numeradas de la misma forma que las anteriores.
- El emisor recibe un reconocimiento de la trama 7.

Ante esta situación, el emisor no tiene forma de saber si la segunda serie de ocho tramas ha sido recibida o no. Supongamos que la segunda secuencia de tramas ha llegado satisfactoriamente. En tal caso, el receptor enviará un reconocimiento de la última, es decir, de la 7. Sin embargo, supongamos que la trama 0 llega con error. En tal caso las tramas sucesivas serán descartadas por el receptor, y éste enviará un reconocimiento de la trama 7 que es la última que le llegó correctamente. Si las tramas de datos 1 hasta 7 han salido antes de la

llegada del reconocimiento, el receptor puede interpretar que se le reconocen todas las tramas, cuando en realidad ninguna de ellas ha sido aceptada.

Este problema no se producirá si no hubiéramos permitido que la secuencia de nuevas tramas enviadas alcanzara el último reconocimiento recibido. Si permitimos una ventana de 7 tramas, la recepción correcta proporcionará un reconocimiento de la trama 6, mientras que en caso de problemas se reconocería la 7. Por tanto, deducimos que el número máximo de tramas sin reconocer (o lo que es lo mismo, el tamaño de la ventana de emisión) debe ser menor o igual a **MaxSeq,** a pesar de tener **MaxSeq+1** números de secuencia.

Por último trataremos el problema de los TimeOuts. Cada trama pendiente de reconocimiento tendrá un temporizador (timer) asociado. Por ello debemos disponer de **MaxSeq** temporizadores. Todos estos timers se pueden simular vía software usando un único reloj hardware que causa interrupciones periódicamente. Los TimeOuts pendientes se colocarán en una lista enlazada, donde cada elemento de esa lista contiene datos acerca del número de ticks (pulsos de reloj) que faltan para que se produzca el TimeOut, la trama asociada a dicho TimeOut y un puntero al siguiente TimeOut. Cada interrupción de reloj se recorrerá la lista decrementando los campos correspondientes en cada nodo. Cuando en alguno de los nodos se alcance el valor cero, se elimina de la lista y se produce el evento TimeOut.

```
{Protocolo5 (vuelta atrás). Este protocolo puede enviar "MaxSeq" tramas
consecutivas sin necesidad de recibir ningún reconocimiento. Además, el
nivel de red no siempre tendrá un paquete disponible para su envío. Para
indicar la disponibilidad de un paquete, el nivel de red generará un
evento del tipo NetworkLayerReady. }

type EvType = (FrameArrival, ChkSumErr, TimeOut, NetworkLayerReady);

procedure protocol5;
var NextFrameToSend: SeqNumberNr;
    Frameexpected: SeqNumberNr;
    AckExpected: SeqNumberNr;
    r, s; frame;
    buffer: array[SeqNumberNr] of packet;
    nbuffered: SeqNumberNr;
    i: SeqNumberNr;
    event: EvType;

function between (a, b, c : SeqNumberNr) : boolean;
{ devuleve cierto, si a <= b < c de forma cirvular, sino falso}
begin
   if ((a<=b) and (b<c)) or ((c<a) and (a<=b)) or ((b<c) and (c<a)) then
      between := true;
   else
      between := false;
end;

procedure SendData (frameNr: SequenceNr);
begin
   s.info := buffer[FrameNr];
   s.seq := FrameNr;
   s.ack :=  (FrameExepcted + MaxSeq) mod (MaxSeq +1);
   ToPhysicalLayer(s);
   StartTimer(FrameNr);
end; {SendData}

begin {protocolo 5}
   EnableNetworkLayer;
   NextFrameToSend := 0;
   AckExpected := 0;
   FrameExpected := 0;
   nbuffered := 0;

   repeat
      wait(event);
      case event of
         NetworkLayerReady:
            begin
               FromNetworkLayer(buffer[NextFrmaeToSend]);
               nbuffered := nbuffered +1;
               SendData(nextFrameToSend);
               inc(NextFrameToSend);
            end;
         FrameArrival:
            begin
               FromPhysicalLayer(r);
               if (r.seq = FrameExpected then
                  begin
                     ToNetwrokLayer(r.info);
                     inc(FrameExpected);
                  end;
               while btween(AckExpected, r.ack, NextFrameToSend) do
                  begin
                     nbuffered := nbuffered -1;
                     StopTimer(AckExpected);
                     inc(AckExpected);
                  end
            end;
         ChkSumErr: ;

         TimeOut:
            begin
               NextFrameToSend := AckExpected;
               for i:=1 to nbuffered do
                  begin
                     SendData(NextFrameToSend);
                     inc(NextFrameToSend);
                  end
            end
      end;
      if nbuffered < MaxSeq then
         EnableNetworkLayer
      else
         DisableNetworkLayer
   until doomsday
end; {protocolo 5}
```

Como podemos apreciar el protocolo5 contiene diferencias notables respecto al protocolo4. En primer lugar, no asumimos que el nivel de red está constantemente activo, sino que cuando tiene un paquete para depositar causa un evento ***NetworkLayerReady***. Por otro lado, y respondiendo al método de ventana múltiple de emisión, existe un buffer de paquetes emitidos

pero aún no reconocidos que se gestiona mediante las variables ***NextframeToSend*** y ***AckExpected***. La primera indica donde se sitúa el margen superior de la ventana, mientras que la segunda apunta a su límite inferior. Tras las inicializaciones entramos en un bucle en el cual se espera un evento, se gestiona el mismo y posteriormente se decide si se habilita o deshabilita el nivel de red.

5.3.3 Protocolo de repetición selectiva

El protocolo de *vuelta Atrás* trabaja bien si la tasa de errores en la línea de transmisión no es alta, es decir, que los errores se producen muy esporádicamente. Sin embargo, si la calidad de la línea no permite bajas tasas de error, este protocolo pierde mucho tiempo retransmitiendo tramas que en realidad ya han llegado correctamente al receptor, pero no han sido aceptadas por este. Por lo tanto se desperdicia una gran cantidad de tiempo y recursos, tal como se vió en ejemplos anteriores.

La estrategia alternativa a la anterior en cuanto al manejo de errores en la línea es la ya comentada anteriormente. Consiste en permitir al receptor que acepte en memoria las tramas correctas que lleguen después de la recepción de una trama dañada o perdida.

En este protocolo el emisor y el receptor mantienen ventanas de tamaño mayor que la unidad. Este tamaño depende de los número de secuencia que implementemos. La ventana del emisor comienza en 0 y crece hasta un máximo **MaxSeq**. El emisor tendrá un buffer reservado para cada número de secuencia dentro de esta ventana.

El receptor, además de mantener un buffer para cada número de secuencia posible, tendrá un bit asociado a cada buffer que indicará si el buffer está lleno o vacío. En principio, la ventana del receptor es siempre fija, y de un tamaño igual a **MaxSeq**.

Cuando llega una trama correcta, su número de secuencia se chequea mediante la función **between** para comprobar que cae dentro de la ventana. Si es así, y no había sido recibido antes, se acepta y almacena. El paquete contenido en ella no se pasará al nivel de red hasta que todas las tramas con número de secuencia inferiores hayan llegado y sean entregadas al nivel de red en el orden en que se enviaron.

La recepción no secuencial introduce una serie de problemas que no están presentes en los protocolos anteriores, en los cuales las tramas se aceptan únicamente en orden. Para ver estos problemas supondremos un ejemplo en el cual disponemos de un número de secuencia de tras bits, de manera que el emisor puede transmitir hasta 7 tramas sin necesidad de recibir ningún reconocimiento.

- En un primer momento se envían las tramas numeradas de 0 a 6.

- El receptor puede recibir las tramas 0 a 6, ambas inclusive. Supondremos que las recibe correctamente, pasa los paquetes asociados a su nivel de red y avanza su ventana para recibir las tramas 7,0,1,2,3,4 y 5. Todos los buffers se marcan como vacíos.

- Supongamos que todos los reconocimientos se pierden por errores en a línea. En tal caso, tras un cierto tiempo, el emisor sufriría un evento de ***TimeOut***, por lo que retransmitirá la trama 0.

- La trama 0 llega al receptor. Como se encuentra dentro de la ventana de recepción, es aceptada y almacenada.

- Si en ese momento el receptor manda un reconocimiento de la trama 6, indicando que se han aceptado las tramas anteriores...

- El emisor ve reconocida la trama 6, luego emite las tramas 7,0,1,2,3,4 y 5. La trama 7 será aceptada por el receptor y pasada al nivel de red. A continuación el receptor pasará la trama 0 que tiene almacenada al nivel de red.

Como puede verse, el nivel de red recibe en dos ocasiones la trama 0 enviada en primer lugar, en vez de recibir dos tramas cero distintas. Por lo tanto, en este caso falla el protocolo.

Este fallo se produce porque al avanzar la ventana del receptor, el nuevo rango de números de secuencia que se admite se solapa con el anterior. Si se pierden los reconocimientos anteriores, el receptor tomaría la siguiente trama como nueva, siendo un duplicado de las anteiores. En el caso en que los reconocimientos lleguen correctamente, las tramas que llegan serían nuevas.

Para resolver este problema obligaremos a que el tamaño máximo de la ventana sea al manos la mitad del rango de números de secuencia. Así, si tulizamos 4 bits para los números de secuencia obtendremos un rango entre 0 y 15, por lo que el tamaño máximo de la ventana de emisión debe limitarse a 8 tramas sin recibir reconocimiento.

De esta manera el receptor al aceptar las tramas 0 a 7 enviadas avanzará su ventana para permitir acptar las tramas de 8 a 15. Cualquier retransmisión de las tramas anteriores caerá fuera de la ventana, y por tanto no será considerada. En general, el tamaño de la ventana para el protocolo6 debe ser ***(MaxSeq+1)/2***.

En este caso, el receptor tendrá un número de buffers igual al tamaño de la ventana de recepción, ya que nunca contendrá un número de tramas mayor que este tamaño. En el caso anterior (**MaxSeq=15**), el tamaño es de 8 tramas, por lo que a la trama *i* le corresponderá el temporizador *i Mod 8.*

En el protocolo 5 cuando se recibía una trama no se enviaba el reconocimiento inmediatamente, sino que se esperaba a la transmisión de una trama en sentido contrario para realizar el **Piggybacking**. Si el tráfico en un sentido es escaso, el reconocimiento para las tramas del sentido contrario se retrasaría excesivamente.

Esto se soluciona en el protocolo 6 mediante el siguiente mecanismo: después de que una trama en secuencia llegue, se lanza un temporizador auxiliar mediante el procedimiento ***StartAckTimer***. Si existe poco tráfico hacia la primera máquina y no se presenta ninguna trama a enviar antes de que se cumpla este temporizador, éste provoca el evento ***NetworkLayerIdle*** y el receptor envía una trama de reconocimiento (sin datos).

El protocolo 6 realiza un mejor tratamiento de errores que el protocolo 5. Cuando el receptor sospoecha que se ha producido un error, manda al emisor una trama de reconocimiento negativo **NAK** (Not acknowledgement). De esta forma no hay que perder el tiempo esperando que el TimeOut del emisor provoque la retransmisión de la trama supuestamente errónea.

El receptor sospechará un error si se produce alguna de estas situaciones:

- Llega una trama dañada.
- Llega una trama no esperada, lo cual indica la pérdida de una trama.

El receptor mantendrá una variable **NoNak** que se encargará de evitar múltiples emisiones de **Nak** para una misma trama perdida. Si **NoNak** es cierta, entonces se puede mandar una trama **Nak** para la trama ***FrameExpected***. Si es falsa, ya se ha mandado una vez y no se deben mandar más.

Si el **Nak** se pierde no ocurre nada ya que en el emisor se producirá, el algún momento, un TimeOut para la trama errónea, con lo que se producirá su retransmisión.

En cuanto al período de TimeOut, hay que distinguir dos casos. En el caso más sencillo el tiempo transcurrido entre la transmisión de una trama y la llegada de su reconocimiento será casi constante. En esas circunstancias, el ajuste del TimeOut del receptor es sencillo, tomando un tiempo ligeramente mayor que el necesario. Sin embargo, en otras ocasiones este tiempo es muy variable. En este caso se presenta el problema de elegir este TimeOut. Si el TimeOut es excesivamente corto se producirán retransmisiones innecesarias con alta frecuencia. Por otro lado, si es demasiado largo el emisor permanecerá mucho tiempo bloqueado esperando un

reconocimiento que quizá no llegue. Si el tráfico en sentido contrario es esporádico, el tiempo hasta el reconocimiento será irregular, corto cuando exista tráfico y largo cuando no exista. En este caso el uso de tramas **Nak** mejora considerablemente la eficiencia del protocolo.

El último punto a tratar acerca de este protocolo consiste en determinar cual es la trama que ha producido un TimeOut. En el caso del procotolo5, este evento siempre era producido por la trama más vieja (AckExpected). Sin embargo, en este protocolo no es necesariamente la trama con el menor número de secuencia. Para comprobarlo, veremos un ejemplo. Supongamos que se han transmitido las tramas de 0 a 4, por lo que la lista de tramas enviadas será 01234, ordenada de más antigua a más nieva. Imaginemos que 0 sufre un TimeOut y 6 (otra nueva trama es enviada. Bajo estas circunstancias, la lista de tramas enviadas será 3405126, de más vieja a más nueva. Su en este punto no se recibiera ningún reconocimiento, las tramas saldrían en este orden. Para no complicar el protocolo excesivamente, no se ha implementado el gestor de los temporizadores. Sin embargo suponemos que la variable **OldestFrame** es modificada por esta rutina para indicar cual de las tramas ha causado el TimeOut.

Hay que destacar la importancia de la implementación de los temporizadores en el esquema del protocolo. Resulta bastante obvio que únicamente necesitaremos (**MaxSeq+1)/2** temporizadores. Sin embargo, en el protocolo se han utilizado **MaxSeq+1**. El motivo es sencillo. A pesar de que solo van a encontrarse en memoria la mitad de tramas de las podemos enviar, necesitamos conocer el número exacto de la trama que ha causado el TimeOut para ser capaces de retransmitirla. Ya comentamos que la variable **OldestFrame** era modificada por esta rutina. Sin embargo, según la implementación comentada con anterioridad la rutina de gestión de temporizadores no puede saber si la trama que causa el TimeOut del temporizador *i* es la *i* o la *i + (Maxseq+1)/2* (en el ejemplo anterior, el temporizador 1 es utilizado por la trama 1 y por la 5). Por tanto, difícilmente podrá actualizar el valor de la variable **OldestFrame.**

A continuación se incluye el listado del protocolo 6:

```
{Protocolo 6 (con recepción no secuencial) acepta tramas desordenadas,
pero pasa los paquetes a la capa de red en forma ordenada. Hay un
temporizador asociado con cada trama pendiente. Cuando el temporizador
expira, sólo se retransmite esa trama, y no todas las tramas pendientes,
como en el caso del protocolo 5. }

const NrBuf = ... {NrBuffs = (MaxSeq + 1) divv 2 }
      MaxBuf = ...{MaxBuf = NrBufs -1 }

type EvType = (FrameArrival, ChkSumErr, TimeOut, NetworkLayerReady);
     bufnr = 0..MaxBuf;

procedure procotol6;
var NextFrameToSend: SeqNumberNr;
    Frameexpected: SeqNumberNr;
    AckExpected: SeqNumberNr;
    TooFar: SeqNumberNr;
    OldestFrame: SeqNumberNr;
    i: bufnr;
    r, s; frame;
    OutBuf: array[bufnr] of packet;
    InBuf: array[bufnr] of packet;
    arrived: array[bufnr] of boolean;
    nbuffered: SeqNumberNr;
    NoNak: boolean;
    event: EvType;

function between (a, b, c : SeqNumberNr) : boolean;
{ devuleve cierto, si a <= b < c de forma cirvular, sino falso}
begin
   if ((a<=b) and (b<c)) or ((c<a) and (a<=b)) or ((b<c) and (c<a)) then
      between := true;
   else
      between := false;
end;

function SendFrame(fk: FrameKind; frameNt: SequenceNr);
begin
   s.kind := fk;
   if fk = data then s.info := OutBuf[FrmaeNr mod NrBufs];
   s.seq := FrameNr;
   s.ack := (FrameExpected + MaxSeq) mod (MaxSeq +1);
   if fk = nak then NoNak := false;
   ToPhysicalLayer(s);
   if fk = data then startTimer(frameNr);
   StopAcktimer;
end; {SendFrame}

begin {protocolo 6}
   EnableNetworkLayer
   NextFrameToSend := 0;
   AckExpected := 0;
   FrameExpected := 0;
   nbuffered := 0;
   TooFar := 0;
   NoNak := true;
   for i :=0 to MaxBuf do arrived[i] := false;
   repeat
      wait(event);
      case event of
         NetworkLayerReady:
         begin
            FromNetworkLayer(OutBuf[NextFrameToSend mod NrBuf]);
            nbuffered := nbuffered +1;
            SendFrame(data, NextFrameToSend);
            inc(NextFrameToSend);
         end;
         FrameArrival:
         begin
            FromPhysicalLayer®;
            if r.kind = data then
            begin
               if (r.seq <> FrameExpected) and NoNak then
                  SendFrame(nak, 0);
               if between(FrameExpected, r.seq, TooFar) and
                  (arrived[r.seq mod NrBufs] = false) then
               begin
                  arrived[r.seq mod NrBufs] := true;
                  InBuf[r.seq mod NrBufs] = r.info;
                  while arrived[FrameExepcted mod NtBufs] do
                  begin
                     ToNetworkLayer(InBuf[FrameExpected mod NrBufs]);
                     NoNak := true;
                     arrived[FrameExpected mod NrBufs] := false;
                     inc(FrameExepcted);
                     inc(TooFar);
                     StartAckTimer;
                  end
               end
            end;
            if (r.kind = nak) and
               between(AckExpected,(r.ack+1)mod(MaxSeq+1),NextFrameToSend)
               then SendFrame(data, (r.ack+1)mod(MaxSeq+1));
            while between(AckExpected, r.ack, NextFrameToSend) do
            begin
               nbuffered := nbuffered -1;
               StopTimer(AckExpected mod NrBufs);
               inc(AckExpected);
            end
```

```
            end;
             ChkSumErr: if NoNak then SendFrame(nak, 0);
             TimeOut: SendFrame(data, OldestFrame);
             NetworkLayerIdle: SendFrame(ack, 0);
          end; {fin del case}
          if nbuffered < NrBufs then
             EnableNetworkLayer
          else
             DisbleNetwrokLayer
       until doomsday
    end; {protocolo 6}
```

Nivel de Red

Como ya se ha indicado en capítulos anteriores, para una comunicación entre dos estaciones

se precisa de una conexión física entre ambas. Esta conexión puede tener lugar con diferentes tipos de medios, tanto guiados como no guiados. En el caso de los medios guiados su aspecto más frecuente es el de un cable.

Cuando se desea interconectar a una comunidad de usuarios numerosa, una primera opción es la interconexión total, es decir, la conexión de todos con todos, de este forma es posible la comunicación entre cualquier pareja de usuarios. Sin embargo, esta opción es impensable en general, debido a su alto coste y al poco aprovechamiento de recursos que supone.

Como opción alternativa, se plantea el desplazar el problema a un sistema que se encargue de la interconexión y que recibe el nombre de red comunicación (que en el caso de interconexión de ordenadores se particulariza a **red de ordenadores**). El objetivo de esta red es facilitar la comunicación entre cualquier pareja de estaciones.

La interconexión a la red se produce ahora sobre la base de una conexión (un cable) que enlaza a cada usuario con la red de comunicación lo que permite la comunicación desde o hasta la red, generalmente suele tratarse de un enlace punto a punto dedicado. Y, en segundo lugar existe una serie de interconexiones dentro de la red que facilitan la comunicación de las estaciones.

Habitualmente las redes están compuestas de una serie de canales de comunicación y unos elementos activos de conmutación. No se suele emplear la interconexión total (por su elevado coste), aunque suele introducirse cierta redundancia para evitar que un fallo en un enlace aísle partes de la red.

La misión de los elementos de conmutación, denominados nodos o IMPs (*Interface Message Processor*), es conducir la información en cada bifurcación de la red hacia su destino final. A esta tarea se le denomina **encaminamiento**.

Puesto que existen más de dos estaciones conectadas a la red, es preciso que exista un esquema de selección o direccionamiento para que se pueda especificar origen y destino de la información. Cada estación debe tener una dirección que la identifique de manera única.

6.1.1 Clasificación de las redes de comunicación

Las redes de comunicación pueden clasificarse atendiendo a distintos criterios entre los que citaremos los siguientes:

- **Según el tipo de enlace utilizado:** redes punto a punto y redes multipunto.

- **Dependiendo de las técnicas empleadas para transferir los datos:** Redes de conmutación de circuito, de mensaje y de paquete.

- **Según la extensión geográfica que ocupan:** Redes de área extensa (*WAN*), redes metropolitanas (*MAN*), redes de área local (*LAN*).

- **Teniendo en cuenta de quién es propiedad la red:** Redes pública y redes privadas. Las redes públicas son redes establecidas y controladas por alguna autoridad de administración de redes nacionales específicamente para la transmisión de datos.

6.1.2 Redes de conmutación

Pasamos revista en este apartado a las técnicas que se emplean para transferir los datos en una red. Cada una de esta técnicas tiene unas propiedades que pasamos a resumir.

6.1.2.1 Redes de conmutación de circuito

Este tipo de redes se desarrolla para el soporte del sistema telefónico. Cada comunicación

viaja por un canal independiente y la red se encarga de conmutar (conectar) a los dos usuarios.

En esta técnica, cada comunicación emplea de manera exclusiva un circuito, que debe establecerse previa petición de los usuarios, una vez establecido permite la comunicación entre ambos y, finalmente, debe ser liberado cuando se termina la comunicación. La primera etapa implica un retardo antes de la transferencia de datos para el establecimiento del circuito. Una vez que éste se establece la red es transparente a los usuarios. Los datos se transmiten a una velocidad fija, sin otro retardo que el de propagación a través de los enlaces de transmisión, el retardo en cada nodo es despreciable.

El principal inconveniente de la conmutación de circuito es que produce la asignación de un circuito de manera exclusiva a una comunicación, sin tener en cuenta el uso que esta hace del mismo. Así vemos que una comunicación consume los mismos recursos con independencia de la cantidad de información transmitida, el factor determinante del coste es la duración de una conexión. Esto puede suponer en algunos casos un uso poco eficiente de los recursos.

6.1.2.2 Conmutación de mensaje y conmutación de paquete

Para la transmisión de datos, que tienen una naturaleza distinta de la voz humana, se han desarrollado técnicas que se basan en principios diferentes de los utilizados en las redes de conmutación de circuito.

La comunicación entre ordenadores, suele tener un carácter más esporádico que el diálogo entre seres humanos, es decir, el tráfico se suele concentrar en *ráfagas* de información de corta duración. Por este y otros motivos, el uso d redes de conmutación de circuito en este tipo de comunicación puede resultar altamente ineficaz. Por ello, para transmisión de datos digitales se utilizan técnicas alternativas.

La idea fundamental de las dos técnicas de conmutación de mensaje y de paquete es incorporar unas estaciones de conmutación que **almacenen** y **retransmitan** la información, de manera que no sea preciso asignar un circuito dedicado a cada comunicación, sino que para comunicar a dos estaciones sólo es necesario que exista un camino a través de la red que puede estar compartido entre varias comunicaciones.

La comunicación se lleva a cabo, ahora, como intercambio de una serie de mensajes entre las dos estaciones que se comunican. La información (el mensaje), se envía en primer lugar del origen a un nodo de la red, este nodo almacena el mensaje y seguidamente la retransmite al nodo siguiente, o eventualmente a su destino cuando este se encuentre físicamente conectado con ese nodo. De esta forma, el mensaje recorre el camino entre origen y destino a través de una serie de nodos de la red, siendo almacenado y retransmitido varias veces, tantas como *saltos* realice en su recorrido. A este funcionamiento lo denominaremos **conmutación de mensaje**.

Entre las ventajas de este tipo de conmutación frente a la de circuito podemos citar:

- Mejor aprovechamiento de la línea, puesto que un único canal entre dos nodos puede ser compartido por varios mensajes.
- No se requiere disponibilidad simultánea de emisor y receptor.
- Un mensaje puede enviarse a muchos destinos
- Permite el uso de prioridades.
- La red puede realizar control de errores y control de flujo.
- La red puede realizar conversiones de velocidad y de código.

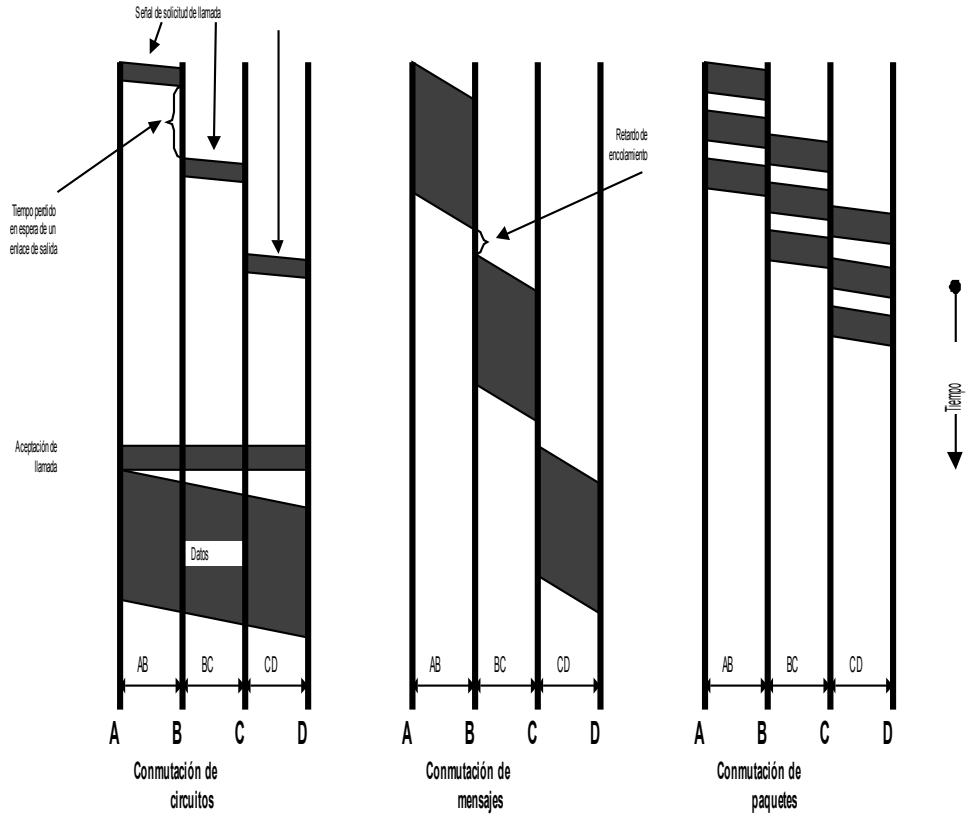
El funcionamiento de este tipo de redes hace que el tiempo necesario para comunicar un mensaje que debe recorrer un camino con tres nodos en la red sea igual al tiempo necesario para transmitirlo cuatro veces. (Cuatro veces superior al tiempo necesario en una red de conmutación de circuito). Por otro lado, es necesario que cada nodo de la red disponga de una cantidad *ilimitada* de memoria para almacenar cualquier mensaje que le llegue y que luego debe retransmitir.

Los dos problemas que acabamos de apuntar hacen que la conmutación de mensaje, si bien en un principio puede suponer una ventaja importante respecto a la conmutación de circuito, no consigue unos tiempos de transmisión aceptables y la comunicación se ve ralentizada en exceso. Este problema es especialmente importante en comunicaciones interactivas.

Las redes de **conmutación de paquete** resuelven el problema del gran retardo en las redes de conmutación de mensaje. Para ello, esta técnica requiere que se limite el tamaño máximo de cualquier bloque de información (que se denomina **paquete**), de manera que un mensaje cuya longitud exceda el tamaño máximo de un paquete deberá ser dividido en varios fragmentos (paquetes). Las longitudes máximas típicas están entre mil bits y unos pocos miles.

Respecto a cómo se consigue que los paquetes recorran el camino entre origen y destino, el método es análogo al empleado en la conmutación de mensaje, con la salvedad de que ahora los retardos serán del mismo orden que la transmisión de circuito una sola vez (y no cuatro cono en el ejemplo aludido en la conmutación de mensaje), ya que paquetes en distintos nodos pueden transmitirse paralelamente.

Por lo tanto podemos afirmar que la comunicación de paquete hereda de la de mensaje sus ventajas pero no sus inconvenientes puesto que los tiempos son del mismo orden que en la conmutación de circuito pero sin emplear circuitos dedicados. Además, la conmutación de paquete no necesita de ingentes cantidades de memoria en los elementos de conmutación (nodos), debido a que los paquetes tienen un tamaño máximo conocido y generalmente pequeño (del orden de unos pocos kilobits).



En la siguiente tabla se muestran los tiempos necesarios para conseguir la comunicación de un mensaje de un megabit, suponiendo circuitos de 10,000 bps de velocidad de transmisión y el empleo de cada una de las técnicas de conmutación aludidas y la ausencia de errores. (No se han considerado los retardos de propagación ni los retardos introducidos por el procesamiento en los nodos).

Conmutación de...	
Circuito	$10^6 / 10^4 = 10^2$
Mensaje	$10^6 / 10^4 \times 4 = 4 \times 10^2$
Paquete	$3 \times 10^{-2} + 1000 \times 10^{-2} = 100,3$
Supóngase un tamaño de paquete de 1000 bits	
El camino pasa por tres nodos	

Como se puede ver, tanto la conmutación de circuito como la de paquete nos proporciona un tiempo del orden de los 100 segundos, por el contrario, la conmutación de mensaje requiere de 400 segundos para transmitir la misma información.

La técnica más empleada en redes de ordenadores es la conmutación de paquetes, que intenta tomar las ventajas de la conmutación de circuito en cuanto a la velocidad y de la de mensaje en cuanto al uso compartido de los enlaces de datos entre varias comunicaciones. En estas redes, la comunicación de paquetes se emplea de dos modos distintos, que dependen de cómo maneje internamente la red los paquetes que recibe, dando lugar a dos tipos:

- Redes de **circuito virtual**
- Redes de **datagrama**

El circuito virtual puede proporcionar un servicio libre de errores y garantiza la secuencialidad de los paquetes, mientras que las redes de datagrama no. Esto no quiere decir que necesariamente las redes de datagrama sean inferiores a las de circuito virtual. Son dos técnicas alternativas y cada una de ellas tiene sus ventajas e inconvenientes.

6.2.1 Circuito virtual y datagrama

Una de las principales características que determinan el comportamiento de una subred de conmutación de paquetes es el funcionamiento mediante **datagrama** o mediante **circuito virtual**. Dicha característica debe analizarse a dos niveles distintos:

- Funcionamiento interno de la subred.
- Tipo de servicio que la subred ofrece a las estaciones que se conectan a ella.

Si la subred funciona utilizando internamente **circuitos virtuales**, previamente a la transmisión de los datos debe establecerse un circuito lógico. Este establecimiento requiere el intercambio de una serie de paquetes de control entre los nodos extremos que desean comunicarse, lo que permite además, negociar una serie de condiciones que regularán el diálogo posterior (caudal deseado, retardo máximo permitido, etc.). Si las condiciones deseadas por el iniciador de la conexión no pudieran conseguirse, ésta no se llevaría a cabo. Durante esta primera fase se establece la ruta que seguirán posteriormente todos los paquetes intercambiados. Por lo tanto, las decisiones de encaminamiento no necesitan realizarse para cada paquete transmitido, sino únicamente para el paquete iniciador de la conexión (*Call request*). Para poder memorizar estas decisiones a cada circuito virtual establecido se le asocia un identificador numérico, y cada IMP dispone de una tabla donde se asocia el número del circuito virtual con alguna de sus líneas de salida.

Superada con éxito esta fase de **establecimiento de la conexión**, se entra en la de **transferencia de datos**. Durante ella se lleva a cabo el intercambio de paquetes de datos, objeto de la comunicación. Cada paquete contiene un identificador de circuito virtual, que permite a los nodos encaminarlo por la ruta adecuada, sin necesidad de hacer uso de las direcciones origen y destino. Una vez finalizado el intercambio de datos, cualquiera de los dos interlocutores puede iniciar la liberación del circuito mediante un paquete específico: *Clear Request*. El intercambio de paquetes realizado a tal efecto, constituye la fase de **liberación de la conexión**.

El funcionamiento mediante datagramas es bastante más simple. Cada paquete se trata de forma independiente, al igual que ocurría con los mensajes en la conmutación de mensaje. Por lo tanto, todos los paquetes deben contener la dirección de destino completa y se llevarán a cabo decisiones de encaminamiento para cada uno de ellos. Esto permite que diferentes paquetes puedan seguir distintas rutas, y puedan llegar desordenados al receptor.

Pero no todo son inconvenientes, además de su simplicidad pueden citarse como ventajas del método su mayor flexibilidad frente a la congestión y su mejor comportamiento frente a caídas en los IMPs. En el caso del circuito virtual el fallo de un IMP provoca la pérdida de todos los circuitos virtuales que lo atraviesan, mientras que en el caso del datagrama se ven afectados únicamente aquellos paquetes que se encuentran en el IMP en ese instante. Además de lo dicho anteriormente, en una comparación ente ambos métodos es necesario comentar que aparte del secuenciamiento el circuito virtual tiene la posibilidad de proporcionar funciones como el control de flujo y control de errores.

Cuestión	Subred por datagramas	Subred de circuitos virtuales
Establecimiento de conexión	Imposible	Necesario
Direccionamiento	Direcciones fuente y destino completas en cada paquete	Cada paquete lleva un número de circuito virtual
Información de estado	La subred no la necesita	Consume memoria en los nodos (tablas para los C.V.)
Encaminamiento	Independiente en cada paquete	Se elige la ruta al establecer el circuito, todos los paquetes la siguen
Efecto del fallo de un IMP	Ninguno, excepto para los paquetes que se pierden en la caída	Se pierden todos los C.V. que pasan a través del IMP
Apropiado para	Servicio externo de Circuitos virtuales o de datagrama	Servicio externo de Circuitos virtuales

Respecto al tipo de servicio ofrecido por la subred, puede presentar también las dos modalidades que acabamos de comentar, pero ahora referido únicamente a la interfaz estación-red, es decir, la visión que el usuario tiene de la red que está utilizando. ¿Necesita establecer una conexión, previamente al intercambio de los datos?¿Recibe los paquetes de forma ordenada?.

Función	Datagramas	Circuitos Virtuales
Dirección de destino	En cada paquete	Sólo en la iniciación
Manejo de errores	Lo hacen las estaciones	Lo hace la subred
Control de Flujo	No lo hace la subred	Lo hace la subred
Secuenciamiento	No	Sí
Iniciación	No	Sí

El funcionamiento a los dos niveles comentados no tiene porqué coincidir, pudiendo darse estas cuatro posibilidades:

- **Circuito virtual interno, servicio de circuito virtual:** Cuando un usuario solicita un circuito virtual, un canal lógico se establece a través de la red. Todos los paquetes siguen la misma ruta al atravesar la red.

- **Circuito virtual interno, servicio de datagramas:** No suele utilizarse. Requeriría el establecimiento de un circuito virtual a través de la red para cada paquete de datos enviado. En este caso no se dispondría de las ventajas de la comunicación mediante circuito virtual, y se tendrían sus inconvenientes magnificados.

- **Datagrama interno, servicio de circuito virtual:** La red manejará cada paquete independientemente, pero la estación destino los recibirá de forma ordenada. Un IMP perteneciente a la red, probablemente el IMP destino, debe encargarse de realizar la ordenación.

- **Datagrama interno, servicio de datagramas:** Los paquetes se tratan de forma independiente en la subred. La estación destino debe encargarse de ordenar la información que recibe, así como de llevar a cabo el control de flujo y el control de errores en colaboración con la estación fuente.

La conveniencia de emplear cualquiera de las combinaciones señaladas dependerá del tipo de aplicaciones para las que se diseñe la red. No necesita los mismos requisitos una aplicación en tiempo real, que una de correo electrónico, por ejemplo.

Este tema se centrará en el funcionamiento interno de la red. El aspecto que presenta la red a las estaciones ligadas a ella está determinado por el protocolo de acceso a la red utilizado, que se estudiará en temas posteriores.

6.3.1 Generalidades

El objetivo principal de una subred de comunicaciones es facilitar el diálogo entre las estaciones conectadas a ella. Debe encargarse de recoger los paquetes que dichas estaciones transmitan y conducirlos a través de la red, hasta el destino deseado. Generalmente, existen varias rutas alternativas posibles y, por ello, se requiere el uso de un procedimiento de **encaminamiento**. Su objetivo será el envío del tráfico desde el origen al destino, de la forma más rápida posible y con la mínima utilización de recursos.

La selección de una ruta se basa, generalmente, en algún criterio de rendimiento. El criterio más simple es la elección de la ruta más corta, es decir, la que atraviesa el menor número de IMPs. Una generalización de este criterio es la **ruta de coste mínimo**, en cuyo caso se asocia un coste a cada enlace y se determina la ruta de menor coste. La asignación del coste está ligada a objetivos de diseño. Por ejemplo, podría estar asociado a la capacidad del enlace, en el sentido de asignar costes más bajos a los enlaces de más alta capacidad. Podría también asignarse un coste en relación con el tiempo de retardo de la cola para utilizar el enlace. En el primer caso, una ruta de coste mínimo proporcionaría el máximo caudal de datos y en el segundo minimizaría el retardo.

Existen varias características deseables en una función ideal de encaminamiento, sin embargo algunas son incompatibles entre sí. Por lo tanto, el objetivo de los diseñadores será llegar a una solución de compromiso entre todas ellas:

- **Simplicidad:** El algoritmo debe responder a reglas sencillas y a programas pequeños y estructurados. Esta característica resulta especialmente deseable en redes grandes o muy cargadas.

- **Bajo consumo:** Debe tenerse en cuenta que los mecanismos de encaminamiento emplean para su funcionamiento los recursos de la red. El coste de implantación del algoritmo debe requerir un consumo moderado de dichos recursos, tanto de memoria y capacidad de cálculo, como de recursos de comunicación para el intercambio de los mensajes necesario en el algoritmo.

- **Fiabilidad:** El algoritmo debe responder de forma adecuada en caso de fallos en la red. La caída de un nodo no debe provocar la interrupción de las comunicaciones que estén teniendo lugar a través de la red en ese instante.

- **Estabilidad:** Dado un tráfico estático se debe alcanzar pronto un régimen estacionario y no producir oscilaciones. Por ejemplo, seleccionar rutas a través de un área de la red incrementa la longitud de los correspondientes enlaces, si la métrica utilizada está basada en el retardo. Como consecuencia de esto, al producirse la siguiente actualización en el encaminamiento el algoritmo tiende a seleccionar rutas a través de áreas distintas. En otros casos, la red podría reaccionar a la congestión en una de sus áreas desplazando la mayor parte de la carga a otra zona, que pasaría entonces a estar congestionada, quedando la primera zona infrautilizada. Este tipo de desplazamientos puede provocar que los paquetes circulen en bucles a través de la red, tardando en alcanzar su destino y aumentando la carga de la red.

- **Adaptación:** El mecanismo de encaminamiento debe adaptarse a los cambios en el tráfico y en la topología. Esta adaptación debe ser rápida, para poder funcionar en tiempo real, y producirse de manera uniforme, sin brusquedades ni oscilaciones.

- **Optimalidad:** El algoritmo debe conducir a soluciones globales óptimas.

La información de encaminamiento suele almacenarse en estructuras conocidas como **tablas de encaminamiento**. Cada IMP de la red posee una de estas tablas. Como mínimo, contiene una entrada por cada destino posible y asociado a este, el enlace de

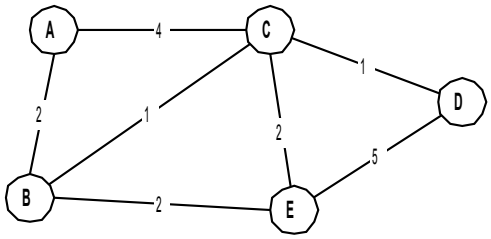
salida que debe utilizarse para alcanzar el nodo siguiente de la red. Las tablas pueden contener además información de coste asociada a la ruta elegida, rutas alternativas, etc.

6.3.2 Clasificación de estrategias de encaminamiento atendiendo a diversos criterios

Existen una gran diversidad de algoritmos de encaminamiento, con distintos niveles de sofisticación y eficiencia. Esta variedad se debe en parte a razones históricas y en parte a las distintas necesidades en redes diferentes.

6.3.2.1 Encaminamiento estático o fijo

Es una de las estrategias más sencillas de encaminamiento. En este caso se selecciona una única ruta para cada pareja de nodos fuente-destino en la red. Las rutas son fijas, o al menos sólo cambian cuando se producen cambios en la topología de la red.



Red ejemplo para encaminamiento fijo

La siguiente tabla muestra como podría implementarse este tipo de encaminamiento. Se crea un directorio central que puede almacenarse en el centro de control de la red. No es necesario almacenar la ruta completa para cada par de nodos, es suficiente conocer para cada par la identidad del primer nodo de la ruta.

		Nodo destino				
Nodo Fuente		A	B	C	D	E
	A	-	B	B	B	B
	B	A	-	C	C	E
	C	B	B	-	D	E
	D	C	C	C	-	C
	E	B	B	C	C	-

Tabla resumen para encaminamiento fijo

	A	B	C	D	E
A	-	B	B	B	B

Tabla del nodo A para encaminamiento fijo

Así, en cada punto de la ruta únicamente es necesario conocer la identidad del próximo nodo; por lo tanto, en cada nodo sólo tiene que almacenarse una fila del directorio.

Con encaminamiento fijo, no hay diferencia entre le encaminamiento para datagramas y circuitos virtuales. Todos los paquetes desde una fuente a un destino determinado siguen la misma ruta. La ventaja de este esquema es evidentemente su simplicidad, además opera correctamente en una red fiable con carga estable. Su inconveniente es la falta de flexibilidad, ya que no reacciona adecuadamente ante fallos o congestión de la red.

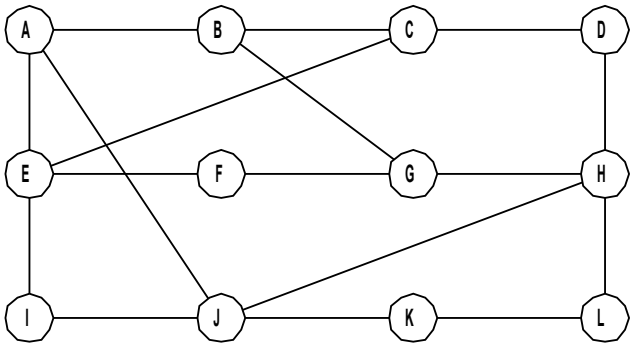
Un refinamiento que permite mejorar la respuesta ante fallos de nodos y enlaces es proporcionar rutas alternativas, para cada dirección.

6.3.2.2 Encaminamiento de camino múltiple

En muchas redes hay varios caminos entre pares de nodos, que son casi igualmente buenos. Con frecuencia, se puede obtener un mejor rendimiento al dividir el tráfico entre varios caminos, para reducir la carga en cada una de las líneas de comunicación. La técnica de utilizar encaminamiento múltiple entre un sólo par de nodos se conoce como **encaminamiento de camino múltiple**.

Este algoritmo se aplica tanto en subredes con datagramas como en subredes con circuitos virtuales. Para el primer caso, cuando un paquete llega a un IMP para su reexpedición, se hace una selección entre varias alternativas, para ese paquete en particular, en forma independiente de las selecciones que se hicieron para otros paquetes que se dirigieron al mismo destino, en el pasado. Para subredes de circuitos virtuales, cada vez que se establece un C.V., se selecciona una ruta, pero el encaminamiento para los diferentes C.V. se lleva a cabo de forma independiente.

Este encaminamiento se puede realizar de la siguiente manera. Cada IMP mantiene una tabla con una ristra reservada para cada uno de los posibles IMP destinatarios; cada ristra ofrece la mejor, la segunda mejor, la tercera mejor, etc. línea de salida para ese destino en particular, junto con un paso o ponderación relativa. Antes de que se reexpida un paquete, un IMP genera un número aleatorio y después selecciona entre las diferentes alternativas que se presentan, haciendo uso de los pesos como probabilidades. Los operadores calculan las tablas manualmente, cargándolas en los IMP antes de que arranque la red y que no se modifique después.



Red ejemplo para encaminamiento de camino múltiple

		Primera Opción		Segunda Opción		Tercera opción	
Destino	**A**	**A**	0,63	**I**	0,21	**H**	0,16
	B	**A**	0,46	**H**	0,31	**I**	0,23
	C	**A**	0,34	**I**	0,33	**H**	0,33
	D	**H**	0,5	**A**	0,25	**I**	0,25
	E	**A**	0,4	**I**	0,40	**H**	0,20
	F	**A**	0,34	**H**	0,33	**I**	0,33
	G	**H**	0,46	**A**	0,31	**K**	0,23
	H	**H**	0,63	**K**	0,21	**A**	0,16
	I	**I**	0,65	**A**	0,22	**H**	0,13
	J	**-**	-	**-**	-	**-**	-
	K	**K**	0,67	**H**	0,22	**A**	0,11
	L	**K**	0,42	**H**	0,42	**A**	0,16

Tabla de rutas para el nodo J

6.3.2.3 Encaminamiento centralizado

Las estrategias discutidas anteriormente, aunque simples, presentan el inconveniente de no adaptarse a las condiciones cambiantes de la red: modificaciones de la topología, variaciones del tráfico, etc. En este apartado analizaremos un método más elaborado, en el que la adaptación se realiza mediante modificaciones de las tablas de encaminamiento. El proceso se realiza siguiendo un esquema centralizado. Para ello la red dispondrá de un nodo central al que llamaremos **centro de control de encaminamiento o RCC** (*Routing Control Center*). Periódicamente, cada nodo enviará información de estado (número de vecinos activos, longitud de las colas de salida, cantidad de tráfico procesado por línea desde el último informe, etc.) al centro de control. A partir de la información recibida y utilizando algún algoritmo de cálculo de ruta de coste mínimo, el RCC construirá las nuevas tablas de encaminamiento y las difundirá a todos los nodos de la red. Por lo tanto, las tablas se construyeron a partir de información global sobre el estado de la red, lo que debe conducir a las soluciones más óptimas. Además, se libera a los nodos de tener que realizar los cálculos relativos al encaminamiento.

Como inconvenientes del método pueden citarse:

- **Vulnerabilidad del centro de control:** como toda la responsabilidad del encaminamiento reposa en el RCC, si este falla o queda aislado de la red por problemas en las líneas, la subred tendrá graves problemas. Puede solucionarse disponiendo de una segunda máquina.

- **Posibilidad de inconsistencias en la actualización de las tablas de encaminamiento:** debido a que los nodos más próximos al RCC reciben la información antes que los nodos situados en la periferia. Esto puede causar bucles en el encaminamiento, que retarden los paquetes en tránsito; entre los paquetes afectados pueden encontrarse algunos con información sobre las nuevas tablas de encaminamiento, lo que agravaría el problema.

- **Alta concentración de tráfico de control:** en las líneas que convergen al RCC.

6.3.2.4 Encaminamiento distribuido. El problema de la actualización topológica

Este método evita el uso del RCC y por lo tanto, sus inconvenientes. La actualización de las tablas de encaminamiento se realiza ahora, en cada nodo, a partir de la información que recibe de los nodos adyacentes a él. Para ello, periódicamente, los nodos de la subred intercambian información de encaminamiento con cada uno de sus vecinos. Las tablas de encaminamiento deberán contener la lista de posibles destinos y una estimación del coste asociado para alcanzar cada uno, en alguna métrica determinada. Cada nodo conoce la distancia que le separa de sus vecinos; lo que le permite, a partir d la información de encaminamiento que posee después de un intercambio, estimar las distancias a cada destino. Por ejemplo, conociendo que la distancia de B a su vecino A es de *m* segundos y sabiendo que la distancia desde A al destino C es de *l* segundos, el nodo B puede deducir que el retardo para alcanzar C vía A es de *m+l* segundos.

Entre las deficiencias más importantes en este tipo de algoritmos pueden citarse:

- Cuando cambian las longitudes de los enlaces, los caminos inducidos por las tablas pueden fomentar la formación de bucles.

- Los algoritmos son muy lentos para rehacerse de fallos en nodos o líneas.

Existe una modificación al método, conocido como ***Hold-Down*** que permite detectar los problemas ocasionados por modificaciones en la topología. En ella, un nodo que ha detectado un empeoramiento brusco, en la que hasta ahora era su mejor ruta para alcanzar un destino determinado, sigue utilizando esa ruta durante un tiempo (tiempo de Hold-down), lo que permitirá a sus vecinos detectar el fallo producido.

En el fondo, el problema con los algoritmos adaptativos distribuidos básicos, como los vistos hasta ahora, es que cuando un nodo conoce que un vecino dispone de una ruta para alcanzar otro nodo distante, no tiene forma de averiguar si esa ruta para a través de él. Esto facilita la aparición de bucles en determinadas condiciones. Este escollo se solucionó en los algoritmos basados en árboles de confluencia.

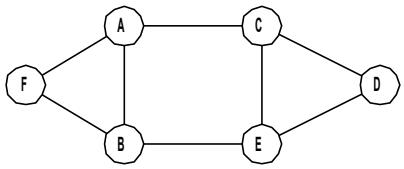
6.3.2.5 Encaminamiento óptimo. Principio de optimalidad. Arboles de confluencia

Independientemente de los detalles del tráfico y la topología de la subred, pueden hacerse ciertas afirmaciones. Una de ellas es el **principio de optimalidad:** Si el nodo J se encuentra en la ruta óptima que une los nodos I y K, entonces el camino óptimo desde J hasta K también forma parte de esa ruta.

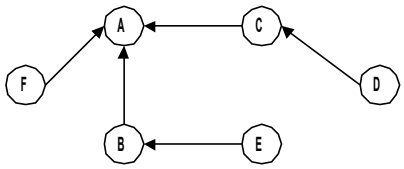
A partir de este principio puede demostrarse que, el conjunto de rutas óptimas que conducen a un destino determinado desde cualquier nodo fuente de una subred constituyen un árbol con raíz en el destino. Dicho árbol recibe el nombre de **árbol de confluencia** (*sink-tree*).

Utilizar un método de encaminamiento basado en una estructura arborescente permite eliminar los bucles de las rutas y garantiza que cada paquete será entregado a su destino en un número finito de saltos. Resultará necesario definir un sentido de **flujo a favor de la**

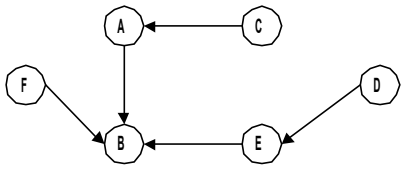
corriente que será desde las hojas a la raíz, el sentido contrario será **contra corriente**. Los caminos entre dos nodos que pertenecen totalmente al árbol de confluencia del nodo destino (y cuyos enlaces se recorran todos a favor de la corriente) serán **caminos óptimos**. Por el contrario, aquellos caminos que impliquen atravesar alguna rama del árbol de confluencia en sentido contrario a la corriente serán **caminos prohibidos**. Los caminos que no emplean ninguna rama del árbol de confluencia en contra de la corriente, pero que tampoco pertenecen en su totalidad al árbol de confluencia son **caminos posibles** pero no óptimos.



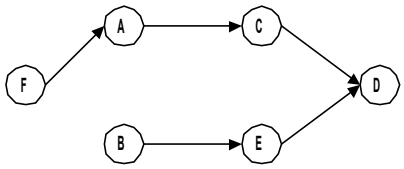
Topología original



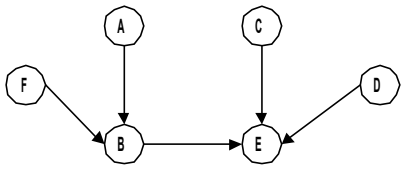
Árbol de confluencia asociado al nodo A



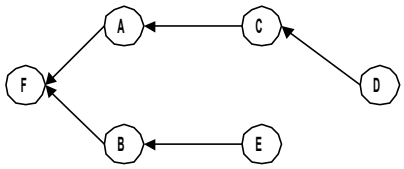
Árbol de confluencia asociado al nodo B



Árbol de confluencia asociado al nodo D



Árbol de confluencia asociado al nodo E



Árbol de confluencia asociado al nodo F

		Enlaces		
Destino		Vía A	Vía D	Vía E
	A	**1**		3
	B	**2**	3	2
	C	-	-	-
	D		**1**	2
	E	3	2	**1**
	F	**2**		3

Tabla de encaminamiento del nodo C

Cada nodo deberá tener información completa sobre los árboles de confluencia

correspondientes a los nodos que desee utilizar como destino. La tabla de encaminamiento de un nodo determinado tendrá una entrada, por ejemplo: fila, por cada posible destino y una columna por cada línea de salida del nodo al que pertenece la tabla. Dicha línea de salido formará parte de alguno de los tres tipos de caminos mencionados.

En la tabla se registrarán únicamente los caminos óptimos y posibles, utilizándose para el envío de paquetes únicamente las rutas óptimas, mientras no se produzcan modificaciones en la topología. Cuando se produce un fallo en algún enlace o nodo perteneciente a algún árbol de confluencia de un destino determinado, el nodo vecino que lo detecta deberá encaminar los paquetes por rutas alternativas; para ello buscará en su tabla de encaminamiento la existencia de algún camino posible, no óptimo que le permita alcanzar el destino deseado. Si lo encuentra lo tomará como nueva ruta. Si no existe ninguno, ya que todos sus vecinos se encuentran contra corriente, les enviará un paquete de control explicando la situación y solicitando caminos alternativos que no pasen a través de el. Si sus vecinos se encontraran en la misma situación – tampoco tuvieran caminos posibles – se repetiría el proceso hasta detectar un camino alternativo. El método garantiza que siempre se encontrará alguno a no ser que el fallo hubiera ocasionado la división de la red en dos partes aisladas.

6.3.2.6 Inundación

A diferencia de los métodos que hemos visto anteriormente, existen otros que no requieren el intercambio de información de control para realizar el encaminamiento, son los métodos de encaminamiento **aislado**.

El primero de los métodos de este tipo de analizaremos es la **inundación**. Se trata de un método de encaminamiento bastante simple. Al recibir un paquete, el nodo lo retransmite por todos sus enlaces, excepto aquél por el que le llegó el paquete.

El principal inconveniente que plantea este método es el gran número de paquetes que se generan, que llegaría a ser infinito si no se establece alguna forma de limitación. Existen diversas posibilidades para ello. Por ejemplo, cada nodo puede mantener una lista de los paquetes ya transmitidos, y al recibir un duplicado destruirlo y no retransmitirlo. Otra posibilidad, más simple, es limitar el tiempo de vida del paquete. En uno de sus campos puede incluirse un contador de saltos, que se decrementará cada vez que el paquete atraviese un enlace; Cuando el contador llega a cero el paquete se descarta. El valor del contador puede inicializarse al diámetro de la red.

Este método de encaminamiento permite encontrar todas las rutas posibles entre origen y destino, entre ellas la ruta mínima; por lo que puede utilizarse como métrica para comparar con otros métodos o para establecer la ruta de un circuito virtual. Por el mismo motivo es, además, muy robusto lo que hace adecuada su aplicación en entornos militares.

El gran número de paquetes que se generan al utilizar este tipo de encaminamiento presenta el inconveniente de que, en condiciones de carga alta de la subred, puede incrementar sensiblemente el retardo de los paquetes transportados por ella.

6.3.2.7 Patata caliente

Este método, aunque maneja únicamente información local para realizar el encaminamiento, se adapta a las condiciones cambiantes del tráfico.

Cada paquete se encamina por el enlace de salida, cuya cola asociada Q, sea más corta. Esto tiene el efecto de equilibrar la carga de los enlaces de salida, pero puede hacer que los paquetes no se dirijan hacia las direcciones más adecuadas.

Una posible mejora al método consiste en tener en cuenta las direcciones preferentes para un determinado destino. Para ello, asigna a cada enlace de salida un peso B_i dependiendo de lo adecuado que resulte para un destino dado *i*. Para cada paquete destinado al nodo *i*

se elige aquel enlace que minimice B_j + Q. Así, ambos factores se tendrán en cuenta. Otras posibilidades son: escoger la mejor elección estática, a menos que su cola exceda un cierto umbral o utilizar la cola de salida más corta, a menos que su peso estático sea demasiado bajo.

6.3.2.8 Aprendizaje retrospectivo

Utiliza la información de los paquetes que llegan procedentes de un nodo fuente dado para recoger información sobre ese nodo actuando como destino. Una posible implementación del método consiste en incluir en cada paquete la identidad del nodo fuente y un contador de saltos. De esta forma, cada vez que el paquete atraviesa un nodo, éste puede conocer cuál es la distancia hasta el nodo fuente del paquete, y de esa forma actualizar los valores de su tabla de encaminamiento, si la nueva ruta es mejor que la que él utilizaba. Cada nodo acabará encontrando las mejores rutas para todos los destinos posibles.

El inconveniente del método deriva de que al recoger únicamente las mejoras en los caminos, si una ruta sufre un empeoramiento, éste no quedará registrado. Para solucionar esto, será necesario que los nodos olviden periódicamente todo lo que han aprendido y comiencen de nuevo su aprendizaje.

Los esquemas de encaminamiento adaptativo aislado se utilizan con menor frecuencia que los centralizados y distribuidos porque aprovechan poco la información disponible.

6.3.2.9 Encaminamiento jerárquico

Independientemente de cual sea el método de encaminamiento empleado a medida que crece el tamaño de la subred, aumenta proporcionalmente el de las tablas de encaminamiento de sus nodos. Lo que puede llegar a hacer inmanejable el tener una entrada por cada nodo destino. El encaminamiento tendrá entonces que hacerse de forma jerárquica, como es el caso de las redes telefónicas.

En este caso, los nodos se agrupan en regiones. Cada nodo conoce todos los detalles sobre los nodos destino pertenecientes a su región, pero no conoce la estructura interna de las otras regiones. Para enviar paquetes a varios destinos pertenecientes a una misma región externa a la suya, lo hará siempre a través del mismo nodo de entrada a dicha región. Esto suele resultar frecuente en la interconexión de redes.

Aunque en algunos casos esta forma de encaminamiento puede conducir a que se incremente la longitud del camino, en general este incremento es pequeño y las ventajas del método lo compensan ampliamente.

6.3.2.10 Encaminamiento para difusión

La difusión consiste en el envío de un paquete simultáneamente a todos los destinos de la red. Tiene diferentes aplicaciones como son: actualización de bases de datos distribuidas, distribución de información de encaminamiento, etc.

Puede llevarse a cabo por distintos métodos:

- **Envío de un paquete distinto a cada destino:** Presenta los inconvenientes de que requiere disponer de la lista completa de destinos y malgasta mucho ancho de banda.

- **Inundación:** Posee características similares al método de inundación utilizado para enviar paquetes de un único destino.

- **Encaminamiento multidestino:** Cada paquete contiene una lista de destino, o un mapa de bits mediante el cuál se indican los destinos deseados. Cuando el paquete llega a un nodo, éste comprueba los destinos alcanzables a través de cada una de sus líneas de

salida y envía copias del paquete a través de esas líneas. Cada copia contiene únicamente las direcciones de los destinos alcanzables a través de la línea por la que es enviado.

Este método requiere disponer de la lista de destinos posibles, pero realiza un mejor aprovechamiento de los servicios de transmisión de la subred que el método 1.

- **Utilización de árbol de confluencia del emisor:** Este método se basa en la utilización de los árboles de confluencia contra corriente. Los paquetes se propagarán desde la raíz del árbol a las hojas. Si cada nodo conoce el árbol de confluencia ligado al nodo fuente del paquete, puede enviar copias del paquete por todas sus líneas de salida que sean ramas del árbol de confluencia, exceptuando aquélla a través de la cuál recibió el paquete.

Este método, genera el número mínimo de paquetes necesarios para la difusión, pero requiere que los nodos de la subred conozcan los árboles de confluencia asociados a cada uno de sus nodos.

- **Marcha atrás *(Reverse Path Forwarding):*** Intenta aprovechar la idea del método anterior sin imponer el conocimiento de los árboles de confluencia. En esta técnica cuando un nodo recibe un paquete de difusión, que proviene de un nodo fuente determinado, a través de la ruta que él utiliza para enviar paquetes a dicho nodo, envía nuevas copias del paquete a través de todas las líneas de salida excepto aquélla por la que recibió el paquete. Si por el contrario, el paquete llega por una línea distinta de la que utiliza para enviar paquetes al nodo fuente, se descarta por ser un duplicado.

Como ventajas del método pueden citarse que resulta ser razonablemente eficiente y fácil de implementar, aunque no proporciona tan buenos resultados como el del caso anterior.

Cuando en una zona de la subred se concentran demasiados paquetes, las prestaciones se degradan. Esta situación se conoce como congestión.

Para entender el fenómeno de la congestión es necesario analizar el comportamiento de la subred de conmutación de paquete como una subred de colas. En cada nodo, asociado a cada canal de entrada o salida habrá una cola de entrada o salida respectivamente. Si la velocidad de llegada de los paquetes al nodo excede la velocidad con que estos pueden ser transmitidos, la cola asociada al canal de salida empieza a crecer y los paquetes irán experimentando un retardo creciente, que podría llegar a tender a infinito si la longitud de las colas lo permitiera. El retardo crece de forma alarmante cuando la tasa de ocupación de la línea, para la que los paquetes están encolados, sobrepasa el 80%.

Cuando se alcanza el punto de saturación y el nodo no puede absorber más paquetes, tiene dos posibilidades:

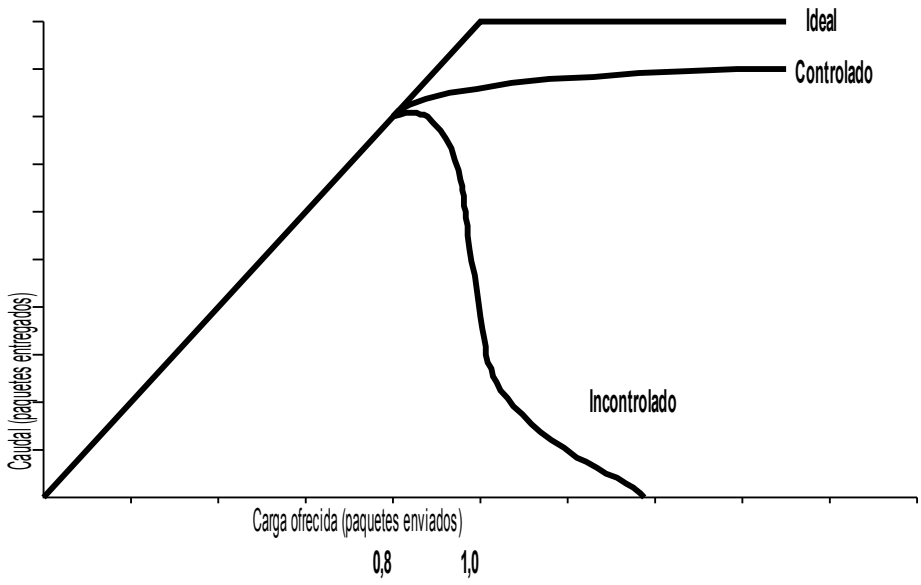
- Rechazar los nuevos paquetes que van llegando.

- Ejercer un control de flujo sobre sus vecinos, impidiendo el envío de nuevos paquetes.

Ambas estrategias conducirán a la saturación de los nodos vecinos al que inicialmente tenía problemas, debido a que no podrán deshacerse de los paquetes que tenían para enviar. Así, la congestión en un punto de la subred se propaga rápidamente hacia las zonas vecinas. Por ello, será deseable establecer algún tipo de control para evitar estas situaciones. Ello disminuirá las prestaciones de la subred respecto al caso ideal en una tasa aproximadamente igual a la sobrecarga de control generada; pero evitará que se

produzcan situaciones catastróficas que podrían conducir al bloqueo total de la subred.

Veamos a continuación algunas de las estrategias utilizadas en el control de la congestión.



Efecto de la congestión sobre el tráfico de la subred

6.4.1 Método de la reserva de buffers

Puede utilizarse únicamente en subredes que utilizan internamente circuitos virtuales. Al establecer el circuito virtual puede reservarse también memoria para almacenar los paquetes en cada nodo de la ruta. Esto garantiza que los sucesivos paquetes que se envíen tendrán sitio y no serán rechazados.

Si lo reserva de memoria no fuera posible, se podría plantear la búsqueda de una nueva ruta para el circuito virtual (o devolver una señal de ocupado al iniciador de la conexión).

Existe una objeción importante a este método: la gestión ineficaz del recurso memoria en los nodos. El reservar a priori este recurso asociándolo de forma exclusiva a circuitos virtuales determinados puede llevarnos al punto de que ninguno de los circuitos virtuales tenga nada que transmitir, que no haya por tanto tráfico alguno en la subred y que paradójicamente la subred esté totalmente ocupada (reservada). Este comportamiento es similar al de las redes de conmutación de circuito.

Debido a este inconveniente algunas subredes permiten únicamente esta reserva para determinados tipos de conexiones, donde un retardo bajo y un elevado ancho de banda resultan esenciales.

6.4.2 Rechazo de paquetes

Contrariamente al caso anterior no se reserva nada, si un paquete llega y no tiene sitio el nodo lo descarta.

Será necesario tener en cuenta algunas consideraciones básicas respecto a los paquetes que se rechazan:

- Sería absurdo rechazar un paquete cuyo contenido incluyera reconocimientos de otros paquetes que están retenidos en el nodo a la espera de su

reconocimiento. Se deberá, por tanto, reservar un *buffer* por cada línea de entrada con objeto de poder inspeccionar los paquetes que llegan. Podría examinarse el paquete, aceptar el reconocimiento por *piggybacking* y rechazar el resto. Otra opción sería utilizar el *buffer* que queda libre, gracias al reconocimiento, para almacenar el nuevo paquete que llega.

- Otra cuestión de interés es la distribución de los *buffers* del nodo (exceptuando los indicados en el apartado anterior) entre las líneas de salida disponibles. La solución trivial de asignar los *buffers* a los paquetes, según su orden de llegada es poco conveniente, ya que podría darse el caso de que inicialmente todos los paquetes que llegaran estuvieran destinados a la misma línea de salida. Si llega otro paquete antes de que se libere un *buffer* debería ser rechazado, aunque este pudiese ser encaminado por alguna de las otras líneas de salida sin tráfico. Este tipo de asignación conduce a una posible distribución del tráfico de salid muy desigual y poco conveniente.

  Una solución mejor es limitar el tamaño de la cola de salida en función del número de *buffers* disponibles y el número de salidas. Otra solución es dedicar un mínimo de *buffers* a cada línea, si no hay tráfico los *buffers* vacíos se reservan.

El principal inconveniente es que se requiere ancho de banda extra para los duplicados. Una manera de minimizar el ancho de banda desperdiciado en la retransmisión de paquetes descartados es rechazar sistemáticamente paquetes que no se hayan desplazado aún lejos del nodo fuente. Un caso límite en esta estrategia es descartar paquetes recién llegados de la estación (host) frente a tráfico que ya esté en tránsito.

6.4.3 Método de los paquetes de restricción

Presenta la ventaja de limitar el tráfico únicamente en situaciones de carga elevada de la red. Para ello, cada nodo monitoriza la utilización de sus líneas de salida. Cuando la tasa de utilización supera un umbral determinado la línea entra en un estado de *alerta*. Al recibir un nuevo paquete, con destino a la estación A, que deba ser enviado a través de es línea, se envía un **paquete de restricción** a la estación fuente del paquete, indicando la estación destino A que constaba en el paquete. El paquete de restricción hará que la estación ralentice sus siguientes envíos hacia ese destino.

Además, el paquete que se envía a través de la línea congestionada se marca (mediante un bit) para que no genere nuevos paquetes de restricción en su camino hacia el destino.

Al recibir un paquete de restricción la estación disminuirá su número de envíos hacia el destino especificado, durante un cierto tiempo, pasado el cuál, si no se han recibido nuevos paquetes de restricción, recuperará su funcionamiento normal.

6.4.4 Control iso-rítmico

Ya que la congestión se produce por un número excesivo de paquetes en la subred, una forma de controlarla sería limitar el número de paquetes presentes en la subred. Este método se denomina iso-rítmico porque mantiene el número de paquetes en la subred constante.

Consiste en tener circulando permanentemente en la subred un número de paquetes especiales de permiso o *crédito*. Cuando un nodo fuente desee mandar un paquete proveniente de su estación debe conseguir primero uno de esos paquetes de permiso y destruirlo, colocando en su lugar el paquete que se desea enviar. Así mismo, cuando un paquete llega al nodo destino y es extraído de la subred, el nodo destino debe regenerar un nuevo paquete de permiso.

Este método posee los siguientes inconvenientes:

- El mecanismo contempla la subred de forma global, por lo que sólo resultará eficaz si el tráfico se encuentra distribuido uniformemente en la subred; en caso contrario, no garantiza que no puedan congestionarse algunos de los nodos a lo largo de las rutas más

utilizadas.

- La distribución de los paquetes de crédito a lo largo de la subred plantea ciertos problemas. Esta distribución debe ser uniforme a lo largo de la subred, par facilitar la captura de estos paquetes por parte de los nodos que desean enviar algún paquete a través de la subred. Hay que tener en cuenta, la posible necesidad de conseguir varios paquetes de forma rápida, para garantizar un ancho de banda elevado necesario en determinados tipos de transmisión. Podría satisfacerse esta demanda mediante un control centralizado.

- Si por algún tipo de error se destruyen paquetes de crédito (errores de transmisión, mal funcionamiento de los nodos, descartados por algún nodo congestionado, etc.) la capacidad de la subred se verá reducida para siempre. No existe una manera sencilla de detectar el número de paquetes de crédito presentes en la subred cuando ésta está en funcionamiento.

Nivel de Transporte

El nivel de transporte es el cuarto nivel en el modelo de referencia ISO/OSI y su principal objetivo es garantizar una comunicación fiable y eficiente entre dos computadoras, con independencia de los medios empleados par su interconexión.

Para conseguir este objetivo se emplea un protocolo de transporte, que regula el intercambio de información entre extremos. En el nivel de transporte aparece, por primera vez, el concepto de comunicación de *extremo a extremo*. El modelo de esta comunicación supone que se dispone de dos computadoras que se comunican (y son los extremos de la comunicación) que están conectados por algún tipo de medio de transmisión y/o red de comunicación. El protocolo de transporte debe conseguir que sea cual sea el medio de comunicación entre ambos extremos, en particular su fiabilidad (que puede no ser muy buena), se pueda asegurar una comunicación fiable entre los extremos.

Por lo tanto, existe una fuerte dependencia entre el servicio de red y el nivel de transporte, puesto que para que el nivel de transporte proporcione un servicio fiable y eficiente deberá tener en cuenta las características del servicio de red y deberá corregir todas las deficiencias que éste pueda presentar utilizando un protocolo de transporte.

Es importante destacar que los objetivos de fiabilidad y eficiencia del servicio de transporte pueden tener una interpretación subjetiva. Distintos usuarios pueden considerar diferentes grados de fiabilidad como deseables, por ejemplo, un usuario que intercambia textos literarios puede considerar aceptable un servicio de transporte que puede cometer un error cada cien megabits recibidos, por el contrario, otro usuario que intercambia información de cuentas bancarias puede considerar ese servicio como inaceptable.

Al igual que sucedía en el nivel de red, el servicio de transporte puede ser orientado a la conexión o sin conexión. Puesto que hemos indicado que uno de los objetivos del nivel de transporte es garantizar la fiabilidad de extremos a extremo, resulta difícil conseguirlo proporcionando un servicio sin conexión. Por lo tanto, nos centraremos en el estudio del servicio de transporte orientado a la conexión.

Con el fin de facilitar una clasificación de los protocolos de transporte y de los requisitos en cada caso, el ISO ha procedido a establecer unos tipos estandarizados para identificar el servicio que proporciona el nivel de red. Existen tres tipos de servicio de red.

	Tasa de errores no detectados	Tasa de errores señalados
Tipo A	Baja	Baja
Tipo B	Baja	Media

Tipo C	Media	Media

Como se aprecia, se distinguen dos tipos de tasa de errores. Los **errores señalados** serán aquella parte de todos los errores que se producen (tasa de error) que son capturados por los mecanismos de detección del nivel de red pero que no son subsanados por él, sin embargo se avisa de su existencia al nivel de transporte. Ejemplos de este tipo de errores serían las perdidas y las reinicializaciones de conexiones de red. Los **errores no detectados** (tasa de error residual) son situaciones de error que no son detectadas (y por tanto no pueden ser corregidas) por el nivel de red. Los casos más típicos son pérdida, corrupción o duplicación de **TPDU** (unidad de datos del protocolo de transporte), así como entrega fuera de secuencia.

En cualquier caso hay que destacar que depende de los usuarios la consideración de qué valor para las tasas de error es considerada baja (aceptable) o media (inaceptable). En función de cual sea el tipo del servicio de red, el nivel de transporte deberá ocuparse de más o menos tareas. Básicamente deberá resolver aquellos problemas no resueltos satisfactoriamente por el servicio de red.

También se ha establecido una clasificación de los protocolos de transporte, atendiendo a las tareas a desarrollar y al tipo de servicio de red sobre el que va a trabajar el nivel de transporte. Se consideran un total de cinco clases distintas:

- **Clase 0 (TP0):** Es la clase más simple, pensada para redes con servicio tipo A, sólo soporta una conexión de transporte para cada conexión de red. Cuando ocurren problemas en el nivel de red que provocan que su usuario reciba un *N_RESET.indication* o un *N_DISCONNECT.indication* si libera la conexión de transporte, invocando una desconexión iniciada por el proveedor. Fue desarrollada para el teletex y otras transmisiones de textos.

- **Clase 1 (TP1):** Desarrollada por el CCITT para emplearse sobre redes X.25 y proporcionar unos mecanismos básicos de recuperación de error. La diferencia con la clase 0 es que maneja los errores señalados sin involucrar al usuario del nivel de trnasporte. Para ello las TPDUs deben numerarse, lo que permite la resincronización después de un RESET de X.25 y reiniciar una conexión de transporte después de un RESTART de X.25.

- **Clase 2 (TP2):** Similar a la clase 0, supone un servicio de red tipo A. Permite la multiplexación de múltiples conexiones de transporte sobre una misma conexión de red. Incorpora un control de flujo basado en créditos.

- **Clase 3 (TP3):** Reúne características de las clases 1 y 2. Incorpora la multiplexación sobre servicios de red tipo B (con errores).

- **Clase 4 (TP4):** Supone que el servicio de red no es fiable en absoluto (tipo C) y que puede presentar todo tipo de fallos. Es el protocolo más complejo porque debe resolver multitud de problemas.

Vamos a considerar tres situaciones ejemplo, que suponen tres casos de servicio de red de cada uno de los tipos, lo que nos permitirá analizar los distintos mecanismos empleados por protocolos de transporte que soportan estos tipos de servicio de red.

Un problema común, con independencia del servicio de red, va a ser que el servicio de transporte admite unidades de datos del nivel de sesión (TSDU) de tamaño arbitrario y debe enviar al servicio de red unidades de datos de tamaño limitado (TPDU). Para ello es preciso un proceso de **segmentación**, que debe ser realizado por el nivel de transporte.

El otro problema clave del nivel de transporte es el **Direccionamiento**. En efecto, aunque el servicio de red nos proporciona un esquema para direccionar cada conexión de red no siempre existe una relación biunívoca entre estas y las conexiones de transporte. Se ha indicado que algunas clases de protocolos de transporte permiten la multiplexación, esto hace que el direccionamiento de red sea insuficiente para el nivel de transporte. Por ejemplo: En una casa, donde viven varias personas y donde disponen de un único teléfono, desde el punto de vista del direccionamiento de red, ese domicilio es un sólo punto. Sin embargo, para atender correctamente a una conexión es preciso identificar con quién se desea hablar en ese teléfono pues son varios los posibles interlocutores. Habitualmente se empleará un direccionamiento jerárquico basado en el direccionamiento de red, que puede consistir en añadir un cierto número de bits a la dirección de red para formar la de transporte.

Seguidamente estudiaremos distintas posibilidades de protocolos de transporte según el servicio de red.

Servicio de red fiable y ordenado (A)

Tenemos un servicio e red que consideramos de tipo A, no se producen errores, el servicio es orientado a la conexión (para poder garantizar el orden y la ausencia de errores). El nivel de transporte no deberá solucionar los posibles errores pues su ocurrencia es despreciable y en casos graves, por ejemplo, pérdida de la conexión de red compensa abortar la conexión de transporte pero mantener la simplicidad del protocolo. A pesar de esto, el nivel de transporte deberá realizar una serie de tareas:

- **Establecer y liberar la conexión de transporte.**

- **Control de flujo:** dado que pueden haber múltiples conexiones de transporte simultáneamente en cada ordenador, es preciso evitar que un receptor lento sea desbordado por el transmisor, ya que entonces se perderían datos por culpa del propio nivel de transporte.

  El problema del control de flujo es más importante en el nivel de transporte, debido a la gran demanda de *buffers* de los protocolos de transporte. Dos motivos provocan esta demanda: el ya comentado de múltiples conexiones y el gran retardo que introduce la red en la transmisión de paquetes.

  La estrategia más simple de control de flujo en el nivel de transporte es, simplemente, apoyarse en el control de flujo proporcionado por el nivel de red. Si cuando la entidad de transporte empieza a tener problemas se niega a aceptar nuevas TPDUs de dicho nivel, su acción provocará la saturación del nivel de red, ya que este no podrá desprenderse de los paquetes recibidos y liberar los *buffers* asociados a ellos. La situación de bloqueo provocará que se activen los mecanismos de control de flujo del nivel de red. La situación de bloqueo actuará como una onda de presión que se transmitirá hasta el otro extremo y acabará originando que el nivel de red n acepte nuevas TPDUs del de transporte. Este técnica se utiliza en los protocolos del tipo TP0 y TP1, sin embargo resulta totalmente inadecuada cuando se emplea multiplexación; en este caso bloquear la conexión de red penaliza a todas las conexiones de transporte que la estén empleando. Resulta, entonces, más indicado un control individualizado sobre cada conexión de transporte.

  Una alternativa es realizar el control de flujo empleando mecanismos de ventana deslizante como los ya vistos en el nivel de enlace de datos. El inconveniente aquí es que se requerirían tamaños de ventana mayores (debido al gran retardo) y esto multiplicado por el número de conexiones activas. Claramente esta estrategia requeriría mucha memoria, por lo que al nivel de transporte suelen emplearse

estrategias más dinámicas.

La solución más habitual consiste en emplear una modificación de la ventana deslizante que se denomina **método de créditos**. La idea es que para poder transmitir una TPDU debemos disponer del *permiso* del receptor. Este permiso se expresa enviando un *crédito* al emisor. Así se puede asegurar que siempre que se envía un TPDU es con el consentimiento expreso del receptor. Puesto que este método combinado con un protocolo de ventana deslizante limita, en ocasiones, el crecimiento de la ventana de transmisión, es denominado de **ventana deslizante de tamaño variable**. Esta técnica se emplea con protocolos de los tipos TP2, TP3 (en ambos de forma opcional) y TP4.

- **Direccionamiento:** Las direcciones de transporte especifican cada uno de los destinos de la información, mientras que las de red especifican cada posible *máquina* destino. El problema surge porque dentro del mismo ordenador (la misma *máquina*) pueden existir varios procesos con distintas conexiones de transporte activas.

  Se suele emplear, un direccionamiento jerárquico, puesto que así es más fácil determinar la dirección de red conociendo la de transporte.

- **Multiplexación:** Como varias conexiones de transporte pueden emplear la misma conexión de red (siempre que todas ellas se establezcan con la misma máquina destino), con el fin de reducir el coste de la comunicación. (Es más barato emplear un sólo circuito virtual que varios).

  El nivel de transporte deberá indicar en cada TPDU a qué conexión pertenece para luego poder entregarla correctamente a sus usuarios. Con este propósito se le asigna a cada conexión un identificador numérico que se incrementa con cada nueva conexión.

- **Segmentación:** Cualquier mensaje puede ser enviado al servicio de transporte, y como estos mensajes pueden tener cualquier longitud, el nivel de transporte debe encargarse de su segmentación (puesto que el servicio de red sólo admite TPDUs de tamaño limitado, que serán enviadas dentro de paquetes en una red que limita el tamaño máximo de esos paquetes).

  Para que no haya errores, la secuencia de TPDUs que se genera como resultado de este proceso de segmentación, deberá ser entregada en orden en el destino. Cuando la red empleada es fiable y ordenada (este es el caso) no es preciso ninguna precaución adicional.

Servicio de red fiable y ordenado, pero con errores detectados (B)

Este servicio de red es, básicamente similar al anterior, pero con fallos ocasionales que son detectados por el nivel de red y por lo tanto se nos informa de los mismos. Entre estos errores, los casos más habituales son:

- **La pérdida de sincronismo:** Por la pérdida de un paquete, o por otros motivos se puede perder la secuencia en los paquetes transmitidos. Para recuperar ese error es preciso **numerar** las TPDUs, y en caso de error retransmitir la TPDU que falta.

- **La interrupción de la conexión:** Si una conexión de red se interrumpe, todas las conexiones de transporte que la utilizaban se ven interrumpidas. Para recuperar ese error, el nivel de transporte –del extremo que había iniciado la conexión- deberá asociar la conexión de transporte con una nueva conexión de red y resincronizar las conexiones de transporte, pues se pueden haber perdido TPDUs.

  Como se ha visto, el cambio más importante en este caso es la necesidad de numerar las TPDUs, lo que nos permite recuperar los errores que se pueden producir en este servicio de red.

Servicio de red no fiable (C)

Se trata de un servicio de tipo datagrama, sin conexión, que no nos asegura ni el orden ni la fiabilidad de los paquetes que enviamos. Para conseguir una conexión de transporte fiable sobre un servicio de red no fiable, debemos tener en cuenta los siguientes aspectos:

- **Retransmisiones;** Como las TPDUs se pueden perder, es preciso un esquema de reconocimientos que nos indique si una TPDU llegó correctamente. Para no penalizar la eficiencia, no se enviarían reconocimientos de todas las TPDUs recibidas correctamente, se emplearán reconocimientos implícitos (se reconoce la TPDU indicada y todas las anteriores).

  ¿Cómo se sabe que una TPDU no ha llegado correctamente? Cuando no llega el reconocimiento dentro de un intervalo temporal determinado. Se van a emplear unos temporizadores que nos indicarán cuando una TPDU se ha perdido. Pero esto nos plantea varios problemas: EL tiempo que tarda un TPDU en recorrer la red no es fijo, con lo que es difícil fijar un valor para el temporizador, por esto mismo es posible que el reconocimiento de una TPDU llegue después de que el temporizador haya vencido. Lo que provocaría la retransmisión de la TPDU.

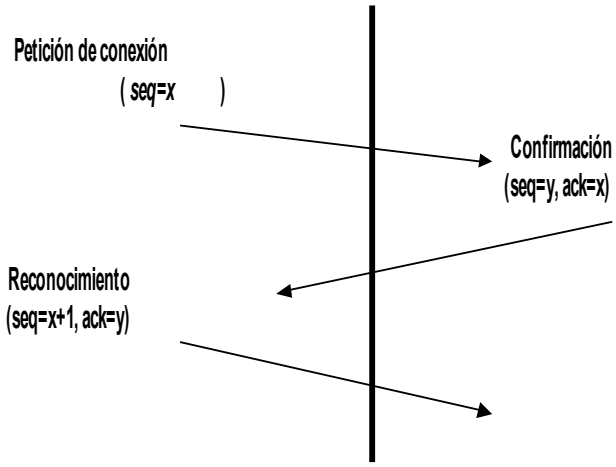
- **Detección de duplicados:** Debido a las retransmisiones es posible que llegue varias veces la misma TPDU, esto es lo que llamamos un **duplicado**. Si las TPDUs están numeradas (y tienen que estarlo en este caso) no hay gran problema en detectar un duplicado que llega mientras aún existe la conexión de transporte (al menos si los números de secuencia no se repiten).

  EL principal problema son los **duplicados retardados** que llegan después de finalizada la conexión de transporte. Si se trata de TPDUs de datos, estas pueden ser descartadas si más problemas siempre y cuando no se haya establecido una nueva conexión de transporte con el mismo identificador. En caso contrario, se pueden emplear distintos sistemas.

  Un primer método consiste un utilizar un número de secuencia que no se repita para la numeración de las TPDU de cada conexión, con ello, si sabemos cuál fue el último número empleado y nos llega una TPDU podemos determinar si es duplicado o es nueva. El inconveniente es que cada ordenador debe *recordar* los números de secuencia empleados en sus conexiones con los demás ordenadores.

  Otro método puede suponer que el servicio de red limita el tiempo de vida de los paquetes, si esperamos el tiempo suficiente antes de abrir una nueva conexión empleando el mismo identificador, garantizamos que entonces habrán desaparecido todas las TPDUs antiguas asociadas con dicho identificador.

  ¿Y si se trata de una petición de conexión? ¿Cómo podemos determinar si esta TPDU es un duplicado si todavía no hemos establecido la conexión? Podemos emplear el método denominado **protocolo a tres bandas**. Se basa en la idea de que las TPDUs están numeradas de forma consecutiva, pero no es necesario que cada ordenador *recuerde* los últimos números de los demás. Para ilustrar su funcionamiento veamos la siguiente figura:



Protocolo a tres bandas

La idea de este método es que con estas tres interacciones cada extremo informa en la primera de su número de secuencia y reconoce en la siguiente el número del otro. Así si se trata de un duplicado retardado, llegará con un número de secuencia que, al ser reconocido por el otro extremo nos alertará del problema.

■ **Control de flujo:** Aunque se puede mantener el esquema de control de flujo empleado en los casos anteriores, es necesario incorporar protección frente a la posible pérdida de las TPDUs de crédito, puesto que de no hacerlo así se podrían producir bloqueos. Además como pueden llegar desordenadas es necesario numerarlas.

■ **Liberación de la conexión de transporte:** El problema al liberar una conexión (al finalizarla) es el riesgo de que esa desconexión se produzca de una forma brusca para no de los extremos. Si uno de los dos extremos está transmitiendo datos cuando el otro le comunica el fin de la conexión se pueden perder datos. Como el nivel de transporte debe garantizar la ausencia de errores en la comunicación, parece claro que deberá resolver este problema.

Sin embargo, este es un problema sin solución: No existe algoritmo que, en tiempo finito, garantice que el cierre de una conexión se produzca siempre de acuerdo entre los dos extremos y sin pérdida de datos. (Por ejemplo, pensemos en una conversación telefónica; normalmente finaliza por mutuo acuerdo entre ambos interlocutores, pero en alguna ocasión, uno de ellos recuerda algo una vez realizada la *despedida* e intenta mantener la comunicación avisando al otro, pero no siempre con la misma suerte y en muchos casos el otro cuelga antes de oír nuestro aviso).

A pesar de lo anterior, existen métodos que pueden reducir la probabilidad de que tal situación se produzca y la conexión quede en un estado no definido (medio abierta, esto es abierta para un extremo y cerrada para el otro).

■ **Recuperación de caídas:** En el funcionamiento de todo sistema informático es posible que se produzcan, con mayor o menor frecuencia, situaciones de error irrecuperables, que dejan paralizado al sistema (o simplemente, cortes de luz). Estas situaciones suceden de manera totalmente imprevisible y su recuperación suele tardar varios minutos cuando menos, llevando además aparejada la pérdida de todos los datos de la memoria principal.

Desde el punto de vista del nivel de transporte, estas situaciones suponen un riesgo de pérdida de información. Veamos cuales son las posibles estrategias para protegernos frente a estas situaciones:

1) Puesto que la información de la memoria principal se pierde en caso de caída, procederemos siempre a grabar las TPDUs recibidas correctamente *antes de enviar los reconocimientos*.

2) Ya que la información podría llegar duplicada, primero enviaremos los reconocimientos y *después almacenaremos las TPDUs recibidas*.

Si reflexionamos, ambos casos pueden causar problemas, el primero puede generar duplicados, al aceptar TPDUs duplicadas (grabadas pero no reconocidas), mientras que el segundo caso puede ocasionar pérdida de TPDUs (reconocidas pero no grabadas). El fondo de la cuestión es que ambas acciones (grabación y reconocimiento) no se ejecutan atómicamente.

DI
I

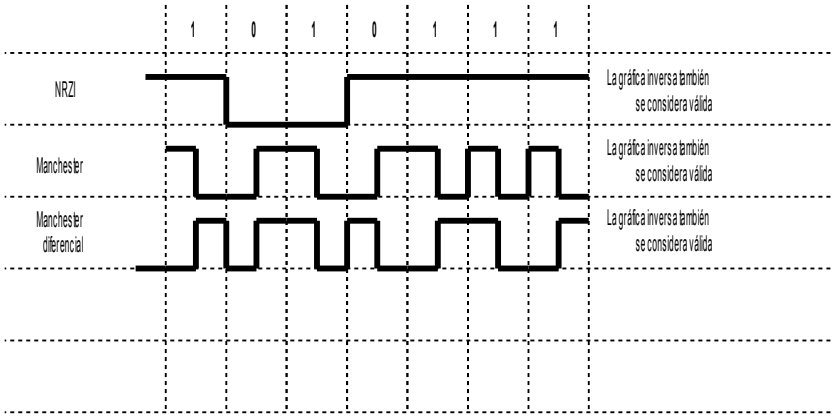
Asignatura: ***IT3518 - TELEINFORMÁTICA Y REDES***
Profesor: ***Javier Meijomence Taboada***

Cuatrimestre:	**2º**	Examen:	**Parcial**	Convocatoria:	**Ordinaria**
Grupo:	**3IT2**	Curso:	**1997/98**	Fecha:	**16 de Marzo de 1.997**

PRIMERA PARTE: Desarrollo

1. Realice un diagrama de codificación para la siguiente señal digital 1010111 empleando los métodos NRZI, Manchester y Manchester diferencial. (1.5 puntos).

Respuesta:



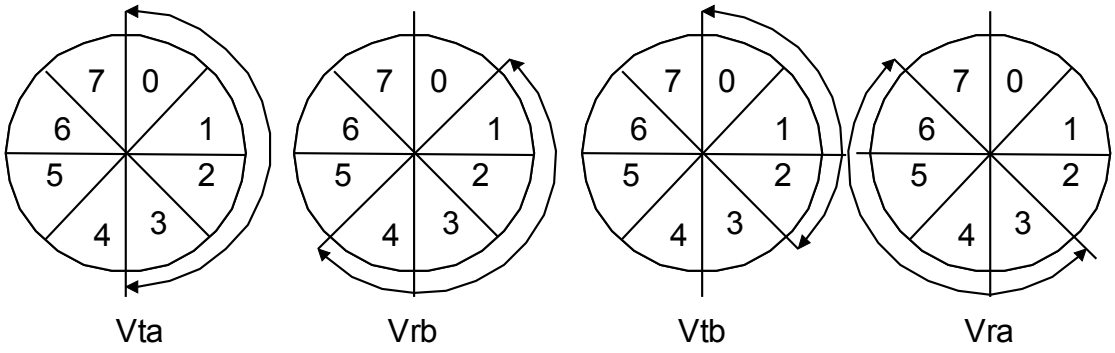
2. Asumiendo una red de N=1000 estaciones, y utilizando el protocolo MLMA, genere la secuencia de décadas necesarias en el intervalo de negociación para resolver las transmisiones de las estaciones: 471, 700, 160, 472, 107 y 168,. (2 puntos)

Respuesta:

9	8	7	6	5	4	3	2	1	0
		1			1			1	
									1
									1
		1							
							1	1	
			1						1
	1								1
		1							

3. Para un protocolo de ventana deslizante con retransmisión selectiva, dada la situación de la figura y suponiendo que no hay ninguna trama en camino y que en el emisor A se acaba

de recibir una trama de reconocimiento negativo (nak). Indíquese su número de secuencia y valor del campo de reconocimiento (0,5 puntos).



Si la trama que se envía como consecuencia de esta se pierde, indicar las acciones que tendrían lugar después en cada uno de los extremos. Debe suponerse que no se producen más errores y que en los emisores sólo se producen más eventos del tipo *NetworkLayerReady* hasta alcanzar el tamaño máximo de ventana permitido en la situación actual. (1,5 puntos)

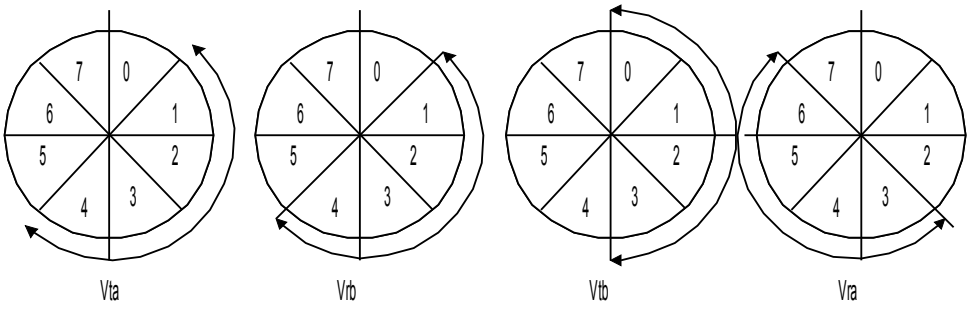
Respuesta:

A recibe: {nak, 0, 0}

La retransmisión de {data, 1, 2} por parte de A es la trama perdida. Pero A puede reducir su ventana de emisión (ha recibido confirmación de la trama 0)

A transmite: {data, 4, 2} y llega al tamaño máximo en su ventana de emisión.
B transmite: {data, 3, 0} y también llega al tamaño máximo en su ventana de emisión.

La situación final de las ventanas es:



SEGUNDA PARTE: Seleccionar la opción más CORRECTA (puntuación: 0,25 cada una; 2 respuestas incorrectas eliminan una correcta; el valor total acumulado en esta parte no puede ser negativo. Total 5 puntos).

1. Los medios de transmisión:
   - ❍ Se comportan como un filtro pasa-banda.
   - ❍ Eliminan todas las componentes de frecuencia por debajo de un cierto valor.
   - ❍ Atenúan por igual todos los componentes de frecuencia.
   - ❍ **Tienen un ancho de banda limitado.**

2. En un protocolo de ventana deslizante con retransmisión selectiva:
   - ❍ Se envía un reconocimiento siempre que se avanza la ventana de recepción.
   - ❍ El tamaño máximo de la ventana de recepción debe ser igual al tamaño medio de la ventana de transmisión.
   - ❍ El rango de número de secuencia determina el número total de tramas que se pueden enviar en una conexión.
   - ❍ **No siempre es posible que la retransmisión sea completamente selectiva, a veces retransmite un trama que ha sido recibida correctamente.**

3. En el método de acceso CSMA p-persistente:
   - ❍ **Cuando una estación quiere transmitir y encuentra el canal libre no siempre transmite.**
   - ❍ Se emplea la técnica LWT (Listen While Talk).
   - ❍ Cuando una estación desea transmitir y encuentra el canal ocupado espera un tiempo aleatorio antes de volver a escuchar.
   - ❍ Generalmente, la voracidad es mayor que en el 1-persistente.

4. ¿Cuál de los siguientes elementos no pertenece a la nomenclatura ARPANET?
   - ❍ **ETD.**
   - ❍ Host.
   - ❍ Línea de transmisión.
   - ❍ IMP.

5. Sobre las topologías multipunto, podemos decir que:
   - ❍ No es necesario ningún mecanismo especial para la gestión de la línea de transmisión.
   - ❍ **La gestión dinámica de la línea optimiza el uso del canal.**
   - ❍ En función del número de máquinas conectadas, se necesita un número determinado de canales de transmisión.
   - ❍ Ninguna afirmación es cierta.

6. En el modelo OSI, el encargado de proporcionar una transmisión libre de errores es el:
   - ❍ Nivel físico
   - ❍ **Nivel de enlace de datos**
   - ❍ Nivel de red.
   - ❍ Sub-nivel de acceso al medio.

7. La modulación FSK tiene:
   - ❍ Portadora digital, moduladora analógica.
   - ❍ Portadora analógica, moduladora analógica.
   - ❍ Portadora digital, moduladora digital.
   - ❍ **Portadora analógica, moduladora digital.**

8. La señal portadora es:
   - ❍ **La que se modifica con la información para enviarla a través del medio.**
   - ❍ La que contiene la información que se desea enviar.
   - ❍ Es la señal que se recibe cuando utilizamos las ondas de radio como canal.
   - ❍ Es la parte de la señal enviada que no contiene ningún tipo de información.

9. De las siguientes topologías punto a punto, indica cual es la que tiene mayor problema de congestión.
   - ❍ Anillo.
   - ❍ Total.
   - ❍ **Estrella.**
   - ❍ Jerárquica.

10. La técnica de *character stuffing*...
    - ❍ Se utiliza únicamente en transmisión asíncrona.
    - ❍ **Permite evitar que el receptor confunda datos con caracteres de control.**
    - ❍ Permite enviar caracteres urgentes.
    - ❍ Permite comprobar que los datos transferidos son correctos.

11. En las topologías multipunto:
    - ❍ **Existe un único canal.**
    - ❍ Hay dos canales por cada Host o IMP.
    - ❍ Hay al menos dos canales por cada HOST o IMP.
    - ❍ Ninguna de las anteriores respuestas es cierta.

12. ¿Cuál será el medio de transmisión más adecuado para comunicar dos estaciones meteorológicas situadas ambas en dos montañas próximas?
    - ❍ Fibra óptica.
    - ❍ Cable coaxial.
    - ❍ Par Trenzado.
    - ❍ **Enlaces de microondas.**

13. Los protocolos del tipo *Stop & Wait:*
    - ❍ Resultan muy eficientes al permitir multiplexar el medio de transmisión.
    - ❍ Disminuyen el tiempo de respuesta del protocolo a niveles superiores.
    - ❍ Aumentan la capacidad de la línea al aprovecharla mejor.
    - ❍ **Desaprovechan la línea al esperar el reconocimiento.**

14. ¿Cuál es la diferencia fundamental entre un protocolo *Stop & Wait* y una de ventana deslizante:
    - ❍ La capacidad de resolver ecuaciones diferenciales.
    - ❍ **La posibilidad de enviar varias tramas sin esperar el reconocimiento.**
    - ❍ La utilización de unidades lógicas de direccionamiento, en vez de físicas.
    - ❍ Los protocolos *Stop & Wait* se utilizan en el nivel de enlace, mientras que los de ventana deslizante se utilizan para la resolución de colisiones en el MAC.

15. Para utilizar la técnica de *Piggybacking* resulta imprescindible que:
    - ❍ Que el protocolo sea de ventana deslizante
    - ❍ **Que el protocolo sea bidireccional.**
    - ❍ Que el protocolo sea Full-Duplex.
    - ❍ Que el protocolo se implemente de forma distinta en ambas máquinas.

16. El propósito del nivel de red es:
    - ❍ Corregir los errores no detectados por el nivel de enlace de datos.
    - ❍ **Encaminar los paquetes y evitar la congestión de la red.**
    - ❍ Incorporar mecanismos de control de flujo.
    - ❍ Conseguir una comunicación libre de errores de extremo a extremo.

17. El modelo de referencia OSI para la interconexión de sistemas abiertos:
    - ❍ Fue desarrollado por el CCITT.
    - ❍ Indica cómo conectar un modem y un computador.
    - ❍ **Fue desarrollado para facilitar la compatibilidad entre sistemas.**
    - ❍ Sólo define los tres primeros niveles.

18. Se desea transmitir la siguiente secuencia de bits: 0000.0111.1100.1101 mediante la técnica de *bit stuffing*. ¿Cuál sería la secuencia resultante? (no se consideran los flags de comienzo y final).

- ❍ 0000.0111.1110.0110.1
- ❍ **0000.0111.1100.0110.1**
- ❍ 0000.0011.1110.0110.1
- ❍ ninguna de las anteriores.

19. ¿Cuál de estos elementos no está relacionado con la atenuación?
    - ❍ Distancia recorrida por la señal.
    - ❍ Distorsión de la señal.
    - ❍ **Interferencias entre canales de comunicación próximos.**
    - ❍ Repetidores

20. La codificación Manchester es:
    - ❍ Es diferencial.
    - ❍ Tiene la ventaja de tener componente continua.
    - ❍ **Es auto-reloj.**
    - ❍ Ninguna de las anteriores.