REDES

UNIVERSIDAD DE LA PUNTA

TEMA 5

2021

Tabla de contenido

La capa de red 3
Aspectos de diseño de la capa de red 3
Conmutación de paquetes de almacenamiento y reenvío 3
Servicios proporcionados a la capa de transporte 3
Implementación del servicio sin conexión 4
Implementación del servicio orientado a conexión 5
Comparación entre las redes de circuitos virtuales y las redes de datagramas 6
Algoritmos de enrutamiento 8
Principio de optimización 9
Algoritmo de la ruta más corta 10
Inundación 12
Enrutamiento por vector de distancia 13
Enrutamiento por estado del enlace 15
Enrutamiento jerárquico 19
Enrutamiento por difusión 20
Enrutamiento multidifusión 22
Enrutamiento anycast 24
Enrutamiento para hosts móviles 25
Enrutamiento en redes ad hoc 26
Algoritmos de control de congestión 29
Métodos para el control de la congestión 30
Enrutamiento consciente del tráfico 31
Control de admisión 32
Regulación de tráfico 33
Desprendimiento de carga 36
Calidad de servicio 38
Requerimientos de la aplicación 38
Modelado de tráfico 40
Programación de paquetes 43
Control de admisión 45
Servicios integrados 48
Servicios diferenciados 50
Interconexión de redes 52
Cómo difieren las redes 53

Cómo se pueden conectar las redes 54
Tunelización 56
Enrutamiento entre redes 56
Fragmentación de paquetes 57
La capa de red de Internet 60
El protocolo IP versión 4 62
Direcciones IP 65
IP versión 6 74
Protocolos de control en Internet 80
Conmutación mediante etiquetas y MPLS 84
OSPF: un protocolo de enrutamiento de puerta de enlace interior 86
BGP: el protocolo de enrutamiento de puerta de enlace exterior 90
Multidifusión de Internet 94
IP móvil 95
Referencias 97

La capa de red

La capa de red se encarga de llevar los paquetes todo el camino, desde el origen hasta el destino. Para llegar al destino tal vez sea necesario realizar muchos saltos en el camino por enrutadores intermedios. Esta función ciertamente contrasta con la de la capa de enlace de datos, cuya única meta es mover tramas de un extremo del cable al otro. Por lo tanto, la capa de red es la capa más baja que maneja la transmisión de extremo a extremo.

Para lograr sus objetivos, la capa de red debe conocer la topología de la red (es decir, el conjunto de todos los enrutadores y enlaces) y elegir las rutas apropiadas incluso para redes grandes. También debe tener cuidado al escoger las rutas para no sobrecargar algunas de las líneas de comunicación y los enrutadores, y dejar inactivos a otros. Por último, cuando el origen y el destino están en redes diferentes, ocurren nuevos problemas. La capa de red es la encargada de solucionarlos. En esta unidad estudiaremos todos estos temas y los ilustraremos, principalmente mediante el uso de Internet y su protocolo de capa de red, IP.

Aspectos de diseño de la capa de red

En las siguientes secciones presentaremos una introducción sobre algunos de los problemas a los que se deben enfrentar los diseñadores de la capa de red. Estos temas incluyen el servicio proporcionado a la capa de transporte y el diseño interno de la red.

Conmutación de paquetes de almacenamiento y reenvío

Antes de empezar a explicar los detalles sobre la capa de red, vale la pena volver a exponer el contexto en el que operan los protocolos de esta capa. En la Ilustración 1 podemos ver este contexto. Los componentes principales de la red son el equipo del *Provedor del Servicio de Internet* (ISP) (enrutadores conectados mediante líneas de transmisión), que se muestra dentro del óvalo sombreado, y el equipo de los clientes, que se muestra fuera del óvalo. El host *H1* está conectado de manera directa a un enrutador del ISP, *A*, tal vez en forma de una computadora en el hogar conectada a un módem DSL. En contraste, *H2* se encuentra en una LAN (que podría ser una Ethernet de oficina) con un enrutador, *F*, el cual es propiedad del cliente, quien lo maneja. Este enrutador tiene una línea alquilada que va al equipo del ISP. Mostramos a *F* fuera del óvalo porque no pertenece al ISP. Sin embargo y para los fines de este capítulo, los enrutadores locales de los clientes se consideran parte de la red del ISP debido a que ejecutan los mismos algoritmos que los enrutadores del ISP (y nuestro principal interés aquí son los algoritmos).

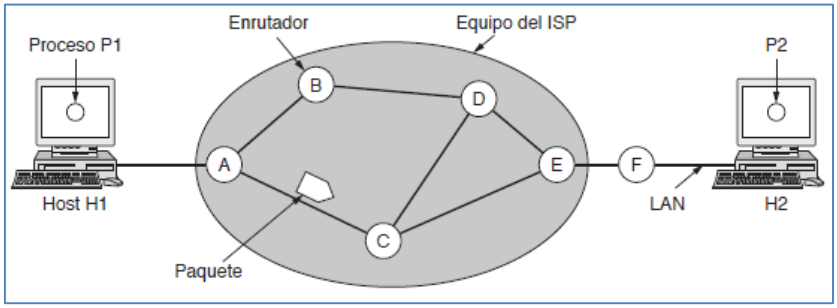


Ilustración 1 - El entorno de los protocolos de la capa de red.

Este equipo se utiliza de la siguiente manera. Un host que desea enviar un paquete lo transmite al enrutador más cercano, ya sea en su propia LAN o a través de un enlace punto a punto que va al ISP. El paquete se almacena ahí hasta que haya llegado por completo y el enlace haya terminado su procesamiento mediante la comprobación de la suma de verificación. Después se reenvía al siguiente enrutador de la ruta hasta que llega al host de destino, en donde se entrega. Este mecanismo se denomina conmutación de almacenamiento y envío.

Servicios proporcionados a la capa de transporte

La capa de red proporciona servicios a la capa de transporte en la interfaz entre la capa de red y de transporte. Una pregunta importante es qué tipo de servicios proporciona precisamente la capa de red a la capa de transporte. Hay que diseñar los servicios de manera cuidadosa, con los siguientes objetivos en mente:

1. Los servicios deben ser independientes de la tecnología del enrutador.
2. La capa de transporte debe estar aislada de la cantidad, tipo y topología de los enrutadores presentes.
3. Las direcciones de red disponibles para la capa de transporte deben usar un plan de numeración uniforme, incluso a través de redes LAN y WAN.

Dadas estas metas, los diseñadores de la capa de red tienen mucha libertad para escribir especificaciones detalladas de los servicios que se ofrecerán a la capa de transporte. Con frecuencia esta libertad degenera en una batalla campal entre dos bandos en conflicto. La discusión se centra en determinar si la capa de red debe proporcionar un servicio orientado a conexión o un servicio sin conexión.

Un bando (representado por la comunidad de Internet) declara que la tarea de los enrutadores es mover paquetes de un lado a otro, y nada más. Desde su punto de vista, la red es de naturaleza no confiable, sin importar su diseño. Por lo tanto, los hosts deben aceptar este hecho y efectuar ellos mismos el control de errores y el control de flujo.

Este punto de vista conduce a la conclusión de que el servicio de red debe ser sin conexión y debe contar tan sólo con las primitivas *SEND PACKET* y *RECEIVE PACKET*. En particular, no debe efectuarse ningún ordenamiento de paquetes ni control de flujo, pues de todos modos los hosts lo van a efectuar y por lo general se obtiene poca ganancia al hacerlo dos veces. Este razonamiento es un ejemplo del **argumento extremo a extremo** (*end-to-end argument*), un principio de diseño que ha sido muy influyente para dar forma a Internet. Además, cada paquete debe llevar la dirección de destino completa, porque cada paquete enviado se transporta de manera independiente a sus antecesores, si los hay.

El otro bando (representado por las compañías telefónicas) argumenta que la red debe proporcionar un servicio confiable, orientado a conexión. Desde este punto de vista, la calidad del servicio es el factor dominante y, sin conexiones en la red, tal calidad es muy difícil de alcanzar, en especial para el tráfico de tiempo real como la voz y el video.

Incluso después de varias décadas, esta controversia aún sigue muy vigente. Las primeras redes de datos que se utilizaron ampliamente, como *X.25* en la década de 1970 y su sucesora, *Frame Relay* en la década de 1980, eran orientadas a conexión. Sin embargo, desde los días de la *ARPANET* e Internet en sus inicios, la popularidad de las capas de red sin conexión ha aumentado en forma considerable. Ahora el protocolo IP es un símbolo constante de éxito. No se vio afectado por una tecnología orientada a conexión llamada ATM, que se desarrolló para desbancarlo en la década de 1980; en cambio, ahora ATM se usa en nichos y es IP quien domina las redes telefónicas. Sin embargo, por detrás, Internet desarrolla características orientadas a conexión a medida que la calidad del servicio adquiere más importancia. Dos ejemplos de tecnologías orientadas a conexión son MPLS (Conmutación Multiprotocolo Mediante Etiquetas), que describiremos en este capítulo, y las redes VLAN que ya vimos.

Implementación del servicio sin conexión

Puesto que ya vimos las dos clases de servicios que la capa de red puede proporcionar a sus usuarios, es tiempo de analizar el funcionamiento interno de esta capa. Se pueden realizar dos formas de organización distintas, dependiendo del tipo de servicio ofrecido. Si se ofrece el servicio sin conexión, los paquetes se transmiten por separado en la red y se enrutan de manera independiente. No se necesita una configuración por adelantado. En este contexto, por lo general los paquetes se conocen como **datagramas** (en analogía con los telegramas) y la red se conoce como **red de datagramas**. Si se utiliza el servicio orientado a conexión, hay que establecer una ruta del enrutador de origen al enrutador de destino antes de poder enviar cualquier paquete de datos. Esta conexión se conoce como **VC** (circuito virtual), en analogía con los circuitos físicos establecidos por el sistema telefónico, y la red se denomina **red de circuitos virtuales**. En esta sección examinaremos las redes de datagramas; en la siguiente analizaremos las redes de circuitos virtuales.

Ahora veamos cómo funciona una red de datagramas. Suponga que el proceso *P1* de la Ilustración 2 tiene un mensaje largo para *P2*. Dicho proceso entrega el mensaje a la capa de transporte y le indica a ésta que lo envíe al proceso *P2* en el host *H2*. El código de la capa de transporte se ejecuta en *H1*, por lo general dentro del sistema operativo. Dicho código agrega un encabezado de transporte al frente del mensaje y entrega el resultado a la capa de red, que quizá sólo sea otro procedimiento dentro del sistema operativo.

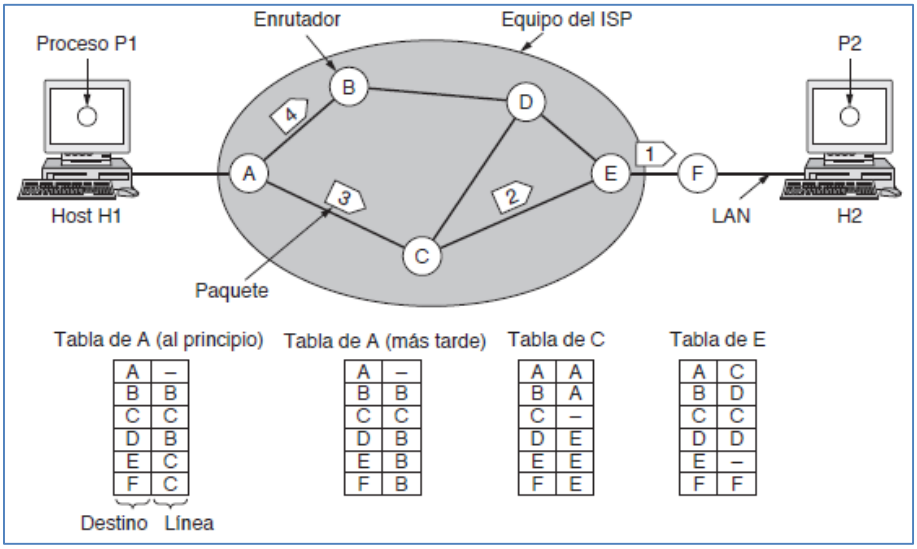


Ilustración 2 - Enrutamiento dentro de una red de datagramas.

Supongamos para este ejemplo que el mensaje es cuatro veces más largo que el tamaño máximo de paquete, por lo que la capa de red tiene que dividirlo en cuatro paquetes: 1, 2, 3 y 4; y enviar cada uno por turnos al enrutador *A* mediante algún protocolo punto a punto; por ejemplo, PPP. En este momento entra en acción el ISP. Cada enrutador tiene una tabla interna que le indica a dónde enviar paquetes para cada uno de los posibles destinos. Cada entrada en la tabla es un par que consiste en un destino y la línea de salida que se utilizará para ese destino. Sólo se pueden utilizar líneas conectadas en forma directa. Por ejemplo, en la Ilustración 2, *A* sólo tiene dos líneas de salida (a *B* y a *C*), por lo que cada paquete entrante se debe enviar a uno de estos enrutadores, incluso si el destino final es algún otro enrutador. En la Ilustración 2, la tabla de enrutamiento inicial de A se muestra bajo la leyenda "*al principio*".

En *A*, los paquetes 1, 2 y 3 se almacenan unos momentos, después de haber llegado por el enlace entrante y de haber comprobado sus sumas de verificación. Después cada paquete se reenvía de acuerdo con la tabla de *A*, por el enlace de salida a *C* dentro de una nueva trama. Después, el paquete 1 se reenvía a *E* y después a *F*. Cuando llega a *F*, se envía dentro de una trama a *H2* a través de la LAN. Los paquetes 2 y 3 siguen la misma ruta.

Sin embargo, ocurre algo diferente con el paquete 4. Cuando llega a *A* se envía al enrutador *B*, aun cuando también está destinado a *F*. Por alguna razón, *A* decidió enviar el paquete 4 por una ruta diferente a la de los primeros tres paquetes. Tal vez se enteró de que había alguna congestión de tráfico en alguna parte de la ruta ACE y actualizó su tabla de enrutamiento, como se muestra bajo la leyenda "*más tarde*". El algoritmo que maneja las tablas y realiza las decisiones de enrutamiento se conoce como **algoritmo de enrutamiento**. Los algoritmos de enrutamiento son uno de los principales temas que estudiaremos en esta unidad. Hay distintos tipos de ellos, como veremos más adelante.

IP (Protocolo Internet), que constituye la base de Internet, es el ejemplo dominante de un servicio de red sin conexión. Cada paquete transporta una dirección IP de destino que los enrutadores usan para reenviar cada paquete por separado. Las direcciones son de 32 bits en los paquetes IPv4 y de 128 bits en los paquetes IPv6. Más adelante describiremos IP con mucho más detalle.

Implementación del servicio orientado a conexión

Para el servicio orientado a conexión necesitamos una red de circuitos virtuales. Veamos ahora cómo funciona. La idea detrás de los circuitos virtuales es evitar la necesidad de elegir una nueva ruta para cada paquete enviado, como en la Ilustración 2. En cambio, cuando se establece una conexión, se elige una ruta de la máquina de origen a la máquina de destino como parte de la configuración de conexión y se almacena en tablas dentro de los enrutadores. Esa ruta se utiliza para todo el tráfico que fluye a través de la conexión, de la misma forma en que funciona el sistema telefónico. Cuando se libera la conexión, también se termina el

circuito virtual. Con el servicio orientado a conexión, cada paquete lleva un identificador que indica a cuál circuito virtual pertenece.

Como ejemplo, considere la situación que se muestra en la Ilustración 3. En ésta, el host *H1* ha establecido una conexión 1 con el host *H2*. Esta conexión se recuerda como la primera entrada en cada una de las tablas de enrutamiento. La primera línea de la tabla A indica que si un paquete con el identificador de conexión 1 viene de *H1*, se enviará al enrutador *C* y se le dará el identificador de conexión 1. De manera similar, la primera entrada en *C* enruta el paquete a *E*, también con el identificador de conexión 1.

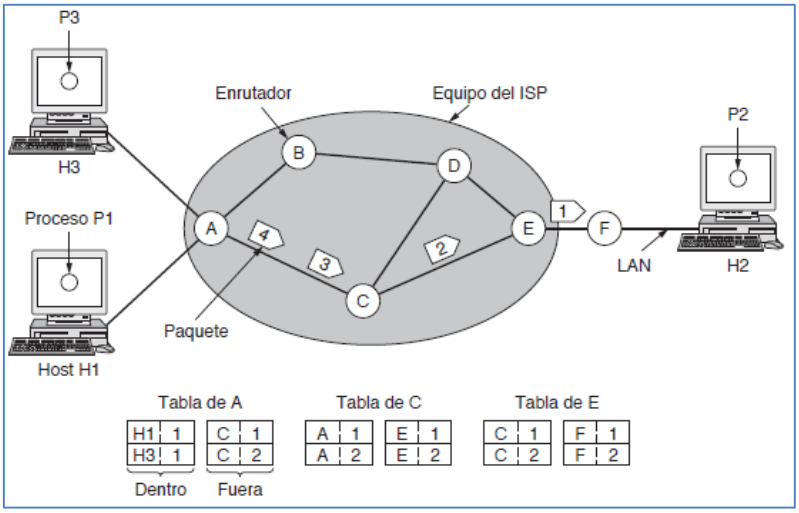


Ilustración 3 - Enrutamiento dentro de una red de circuitos virtuales.

Ahora consideremos lo que sucede si *H3* también desea establecer una conexión con *H2*. Elige el identificador de conexión 1 (debido a que está iniciando la conexión y a que ésta es su única conexión) y le indica a la red que establezca el circuito virtual. Esto nos lleva a la segunda fila de las tablas. Observe que aquí surge un problema, pues aunque *A* sí puede saber con facilidad cuáles paquetes de conexión 1 provienen de *H1* y cuáles provienen de *H3*, *C* no puede hacerlo. Por esta razón, A asigna un identificador de conexión diferente al tráfico de salida para la segunda conexión. Evitar conflictos de este tipo es la razón por la cual los enrutadores necesitan la habilidad de reemplazar identificadores de conexión en los paquetes de salida.

En algunos contextos, a este proceso se le conoce como **conmutación mediante etiquetas**. **MPLS** (Conmutación Multiprotocolo Mediante Etiquetas, del inglés *MultiProtocol Label Swithching*) es un ejemplo de servicio de red orientado a conexión. Se utiliza dentro de las redes de ISP en Internet, en donde los paquetes IP se envuelven en un encabezado MPLS que tiene un identificador de conexión o etiqueta de 20 bits. A menudo el servicio MPLS se oculta de los clientes y el ISP establece conexiones de largo plazo para grandes cantidades de tráfico, pero cada vez se utiliza más para ayudar cuando la calidad del servicio es importante, al igual que para otras tareas de administración de tráfico de ISP. Más adelante en esta unidad veremos más detalles sobre MPLS.

Comparación entre las redes de circuitos virtuales y las redes de datagramas

Tanto los circuitos virtuales como los datagramas tienen sus seguidores y sus detractores. Ahora intentaremos resumir los argumentos de ambos bandos. Los aspectos principales se listan en la Ilustración 4, aunque es probable que los puristas puedan encontrar ejemplos contrarios para todo lo indicado en la figura.

Asunto	Red de datagramas	Red de circuitos virtuales
Configuración del circuito.	No necesaria.	Requerida.
Direccionamiento.	Cada paquete contiene la dirección de origen y de destino completas.	Cada paquete contiene un número de CV corto.
Información de estado.	Los enrutadores no contienen información de estado sobre las conexiones.	Cada CV requiere espacio de tabla del enrutador por cada conexión.
Enrutamiento.	Cada paquete se enruta de manera independiente.	La ruta se elije cuando se establece el CV; todos los paquetes siguen esa ruta.
Efecto de fallas del enrutador.	Ninguno, excepto para paquetes perdidos durante una caída.	Terminan todos los CVs que pasaron por el enrutador defectuoso.
Calidad del servicio.	Difícil.	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV.
Control de congestión.	Difícil.	Fácil si se pueden asignar suficientes recursos por adelantado para cada CV.

Ilustración 4 - Comparación entre las redes de datagramas y de circuitos virtuales.

Dentro de la red existen ventajas y desventajas entre los circuitos virtuales y los datagramas. Una de ellas tiene que ver con el tiempo de configuración y el tiempo de análisis de la dirección. El uso de circuitos virtuales requiere una fase de configuración que necesita tiempo y recursos. Sin embargo, una vez que se paga este precio, es fácil averiguar qué hacer con un paquete de datos en una red de circuitos virtuales: el enrutador sólo usa el número de circuito para buscar en una tabla y encontrar hacia dónde va el paquete. En una red de datagramas no se requiere configuración, pero se requiere un procedimiento más complicado para localizar la entrada correspondiente al destino.

Un aspecto relacionado es que las direcciones de destino que se utilizan en las redes de datagramas son más largas que los números de los circuitos que se utilizan en las redes de circuitos virtuales, ya que tienen un significado global. Si los paquetes tienden a ser bastante cortos, incluir una dirección de destino completa en cada paquete puede representar una cantidad considerable de sobrecarga y, por ende, un desperdicio de ancho de banda.

Otro aspecto más es la cantidad de espacio de tabla requerido en la memoria del enrutador. Una red de datagramas necesita tener una entrada para cada destino posible, mientras que una red de circuitos virtuales sólo necesita una entrada para cada circuito virtual. Sin embargo, esta ventaja es engañosa debido a que los paquetes de configuración de conexión también tienen que enrutarse y utilizan direcciones de destino, al igual que los datagramas.

Los circuitos virtuales tienen ciertas ventajas en cuanto a garantizar la calidad del servicio y evitar congestiones en la red, dado que los recursos (por ejemplo, búferes, ancho de banda y ciclos de CPU) se pueden reservar por adelantado al establecer la conexión. Una vez que comienzan a llegar los paquetes, el ancho de banda y la capacidad de enrutamiento necesarios estarán ya disponibles. En una red de datagramas es más difícil evitar la congestión.

En los sistemas de procesamiento de transacciones (por ejemplo, las tiendas que llaman para verificar compras con tarjeta de crédito), la sobrecarga requerida para establecer y terminar un circuito virtual puede ocupar mucho más tiempo que el uso real del circuito. Si la mayor parte del tráfico esperado es de este tipo, el uso de circuitos virtuales dentro de la red tiene poco sentido. Por otra parte, para usos en donde el tiempo de ejecución sea extenso, como el tráfico VPN entre dos oficinas corporativas, pueden ser de utilidad los circuitos virtuales permanentes (que se establecen en forma manual y duran meses o años).

Los circuitos virtuales también tienen un problema de vulnerabilidad. Si falla un enrutador y pierde su memoria, incluso aunque vuelva a funcionar un segundo después, todos los circuitos virtuales que pasan por él tendrán que abortarse. En contraste, si un enrutador de datagramas falla, sólo sufrirán los usuarios cuyos paquetes se pusieron en cola en el enrutador en ese momento (y tal vez ni siquiera ellos, ya que es probable que el emisor los retransmita poco tiempo después). La pérdida de una línea de comunicación es fatal para los circuitos virtuales que la usan, pero se puede compensar con facilidad si se usan datagramas. Éstos también

permiten que los enrutadores balanceen el tráfico a través de la red, ya que las rutas se pueden cambiar a lo largo de una secuencia extensa de transmisiones de paquetes.

Algoritmos de enrutamiento

La principal función de la capa de red es enrutar paquetes de la máquina de origen a la de destino. En la mayoría de las redes, los paquetes requerirán varios saltos para completar el viaje. La única excepción importante son las redes de difusión, pero aun aquí es importante el enrutamiento si el origen y el destino no están en el mismo segmento de red. Los algoritmos que eligen las rutas y las estructuras de datos que usan constituyen un aspecto principal del diseño de la capa de red.

El **algoritmo de enrutamiento** es aquella parte del software de la capa de red responsable de decidir por cuál línea de salida se transmitirá un paquete entrante. Si la red usa datagramas de manera interna, esta decisión debe tomarse cada vez que llega un paquete de datos, dado que la mejor ruta podría haber cambiado desde la última vez. Si la red usa circuitos virtuales internamente, las decisiones de enrutamiento se toman sólo al establecer un circuito virtual nuevo. En lo sucesivo, los paquetes de datos simplemente siguen la ruta ya establecida. Este último caso a veces se llama **enrutamiento de sesión**, dado que una ruta permanece vigente durante toda una sesión (por ejemplo, durante una sesión a través de una VPN).

En ocasiones es útil distinguir entre el enrutamiento, que es el proceso que consiste en tomar la decisión de cuáles rutas utilizar, y el reenvío, que consiste en la acción que se toma cuando llega un paquete. Podemos considerar que un enrutador tiene dos procesos internos. Uno de ellos maneja cada paquete conforme llega, y después busca en las tablas de enrutamiento la línea de salida por la cual se enviará. Este proceso se conoce como **reenvío**. El otro proceso es responsable de llenar y actualizar las tablas de enrutamiento. Es ahí donde entra en acción el algoritmo de enrutamiento.

Sin importar si las rutas se eligen de manera independiente para cada paquete enviado o sólo cuando se establecen nuevas conexiones, existen ciertas propiedades que todo algoritmo de enrutamiento debe poseer: *exactitud*, *sencillez*, *robustez*, *estabilidad*, *equidad* y *eficiencia*. La exactitud y la sencillez apenas requieren comentarios, pero la necesidad de robustez puede ser menos obvia a primera vista. Una vez que una red principal entra en operación, es de esperar que funcione de manera continua durante años, sin fallas a nivel de sistema. Durante ese periodo habrá fallas de hardware y de software de todo tipo. Los hosts, enrutadores y líneas fallarán en forma repetida y la topología cambiará muchas veces. El algoritmo de enrutamiento debe ser capaz de manejar los cambios de topología y tráfico sin necesidad de abortar todas las tareas en todos los hosts.

La estabilidad también es una meta importante para el algoritmo de enrutamiento. Existen algoritmos de enrutamiento que nunca convergen hacia un conjunto de rutas fijo, sin importar el tiempo que permanezcan en operación. Un algoritmo estable alcanza el equilibrio y lo conserva. Además debe converger con rapidez, ya que se puede interrumpir la comunicación hasta que el algoritmo de enrutamiento haya llegado a un equilibrio.

La equidad y la eficiencia pueden parecer obvias, pero resulta que con frecuencia son metas contradictorias. En la Ilustración 5 se muestra un ejemplo sencillo de este conflicto. Suponga que hay suficiente tráfico entre *A* y *A'*, entre *B* y *B'* y entre *C* y *C'* para saturar los enlaces horizontales. Con el fin de maximizar el flujo total, es necesario suspender el tráfico de *X* a *X'* por completo. Por desgracia, tal vez *X* y *X'* no lo vean de esa forma. Sin duda se requiere cierto compromiso entre la eficiencia global y la equidad hacia las conexiones individuales.

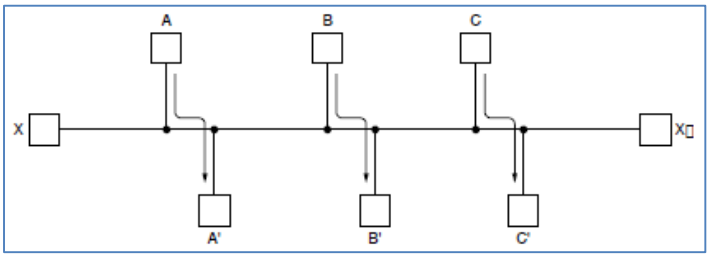


Ilustración 5 - Una red con un conflicto entre equidad y eficiencia.

Antes de que podamos siquiera intentar encontrar el punto medio entre la equidad y la eficiencia, debemos decidir qué es lo que buscamos optimizar. Minimizar el retardo promedio de los paquetes es un candidato obvio para enviar tráfico a través de la red en forma efectiva, pero también lo es aumentar al máximo la velocidad real de transferencia total de la red. Además, estas dos metas también están en conflicto, ya que operar cualquier sistema de colas cerca de su capacidad máxima implica un gran retardo de encolamiento. Como término medio, muchas redes intentan minimizar la distancia que debe recorrer el paquete, o simplemente reducir el número de saltos que tiene que dar un paquete. Cualquiera de estas opciones tiende a mejorar el retardo y también reduce la cantidad de ancho de banda consumido por paquete, lo cual tiende a mejorar la velocidad de transferencia real de la red en general.

Podemos agrupar los algoritmos de enrutamiento en dos clases principales: no adaptativos y adaptativos. Los **algoritmos no adaptativos** no basan sus decisiones de enrutamiento en mediciones o estimaciones del tráfico y la topología actuales. En cambio, la decisión de qué ruta se usará para llegar de *I* a *J* (para todas las *I* y *J*) se calcula por adelantado, fuera de línea, y se descarga en los enrutadores al arrancar la red. En ocasiones este procedimiento se denomina **enrutamiento estático**. Como no responde a las fallas, el enrutamiento estático es más útil para las situaciones en las que la elección de enrutamiento es clara. Por ejemplo, el enrutador *F* en la Ilustración 3 debería enviar al enrutador *E* los paquetes que van dirigidos a la red, sin importar el destino final.

En contraste, los **algoritmos adaptativos** cambian sus decisiones de enrutamiento para reflejar los cambios de topología y algunas veces también los cambios en el tráfico. Estos algoritmos de **enrutamiento dinámico** difieren en cuanto al lugar de donde obtienen su información (por ejemplo, localmente, de los enrutadores adyacentes o de todos los enrutadores), el momento en que cambian sus rutas (por ejemplo, cuando cambia la topología o cada ΔT segundos, a medida que cambia la carga) y la métrica que se usa para la optimización (por ejemplo, distancia, número de saltos o tiempo estimado de tránsito).

En las siguientes secciones estudiaremos una variedad de algoritmos de enrutamiento. Estos algoritmos cubren otros modelos de distribución además de enviar un paquete de un origen a un destino. Algunas veces el objetivo es enviar el paquete a varios destinos, a todos los destinos o a un destino de entre varios pertenecientes a un conjunto. Todos los algoritmos de enrutamiento que describimos aquí toman decisiones con base en la topología; aplazaremos el tema de la posibilidad de las decisiones con base en los niveles de tráfico hasta más adelante.

Principio de optimización

Antes de entrar en algoritmos específicos, puede ser útil señalar que es posible hacer una postulado general sobre las rutas óptimas sin importar la topología o el tráfico de la red. Este postulado se conoce como **principio de optimización** y establece que si el enrutador *J* está en la ruta óptima del enrutador *I* al enrutador *K*, entonces la ruta óptima de *J* a *K* también está en la misma ruta. Para ver esto, llamemos *r1* a la parte de la ruta de *I* a *J* y *r2* al resto de la ruta. Si existiera una ruta mejor que *r2* entre *J* y *K*, se podría concatenar con *r1* para mejorar la ruta de *I* a *K*, lo cual contradice nuestro postulado de que *r1r2* es óptima.

Como consecuencia directa del principio de optimización, podemos ver que el grupo de rutas óptimas de todos los orígenes a un destino dado forman un árbol con raíz en el destino. Dicho árbol se conoce como **árbol sumidero** (o **árbol divergente**) y se ilustra en la Ilustración 6(b), donde la métrica de distancia es el número de saltos. El objetivo de todos los algoritmos de enrutamiento es descubrir y usar los árboles sumidero para todos los enrutadores.

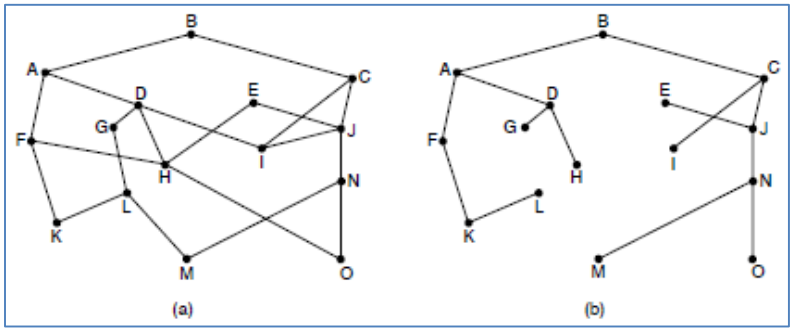


Ilustración 6 – (a) Una red. (b) Un árbol sumidero para el enrutador B.

Cabe mencionar que un árbol sumidero no necesariamente es único; pueden existir otros árboles con las mismas longitudes de rutas. Si permitimos que se elijan todas las posibles rutas, el árbol se convierte en una estructura más general conocida como **DAG** (Gráfico Acíclico Dirigido, del inglés *Directed Acyclic Graph*). Los DAG no tienen ciclos. Usaremos los árboles sumidero como un método abreviado conveniente para ambos casos, que también dependen del supuesto técnico de que las rutas no interfieren entre sí; por ejemplo, un congestionamiento de tráfico en una ruta no provocará que se desvíe a otra ruta.

Puesto que un árbol sumidero ciertamente es un árbol, no contiene ciclos, por lo que cada paquete se entregará en un número de saltos finito y limitado. En la práctica, la vida no es tan fácil. Los enlaces y los enrutadores pueden fallar y recuperarse durante la operación, así que distintos enrutadores pueden tener ideas diferentes sobre la topología actual. Además, hemos evadido la cuestión de si cada enrutador tiene que adquirir de manera individual la información en la cual basa su cálculo del árbol sumidero, o si debe obtener esta información por otros medios. Con todo, el principio de optimización y el árbol sumidero proporcionan los puntos de referencia contra los que se pueden medir otros algoritmos de enrutamiento.

Algoritmo de la ruta más corta

Empecemos nuestro estudio de los algoritmos de enrutamiento con una técnica simple para calcular las rutas óptimas con base en una imagen completa de la red. Estas rutas son las que queremos que encuentre un algoritmo de enrutamiento distribuido, aun cuando no todos los enrutadores conozcan todos los detalles de la red.

La idea es construir un grafo de la red, en donde cada nodo del grafo representa un enrutador y cada arco del grafo representa una línea o enlace de comunicaciones. Para elegir una ruta entre un par específico de enrutadores, el algoritmo simplemente encuentra la ruta más corta entre ellos en el grafo.

El concepto de la **ruta más corta** merece una explicación. Una manera de medir la longitud de una ruta es mediante el número de saltos. Con base en esta métrica, las rutas *ABC* y *ABE* en la Ilustración 7 son igual de largas. Otra métrica es la distancia geográfica en kilómetros, en cuyo caso *ABC* es sin duda mucho más larga que *ABE* (suponiendo que la figura esté dibujada a escala).

Sin embargo, también son posibles muchas otras métricas además de los saltos y la distancia física. Por ejemplo, cada arco se podría etiquetar con el retardo promedio de un paquete de prueba estándar, determinado por una serie de pruebas cada hora. Con estas etiquetas en el grafo, la ruta más corta es la ruta más rápida, en lugar de la ruta con menos saltos o kilómetros.

En el caso general, las etiquetas de los arcos se podrían calcular como una función de la distancia, ancho de banda, tráfico promedio, costo de comunicación, retardo promedio y otros factores. Al cambiar la función de ponderación, el algoritmo calcularía la ruta "*más corta*" de acuerdo con cualquiera de estos criterios, o una combinación de ellos.

Se conocen varios algoritmos para calcular la ruta más corta entre dos nodos de un grafo. Uno de éstos se debe a Dijkstra, el cual encuentra las rutas más cortas entre un origen y todos los destinos en una red. Cada nodo se etiqueta (entre paréntesis) con su distancia desde el nodo de origen a través de la mejor ruta conocida. Las distancias no deben ser negativas, como lo serán si se basan en cantidades reales como ancho de banda y retardo. Al principio no se conocen rutas, por lo que todos los nodos tienen la etiqueta infinito. A medida que

avanza el algoritmo y se encuentran rutas, las etiquetas pueden cambiar para reflejar mejores rutas. Una etiqueta puede ser tentativa o permanente. En un principio todas las etiquetas son tentativas. Una vez que se descubre que una etiqueta representa la ruta más corta posible del origen a ese nodo, se vuelve permanente y no cambia más.

Para ilustrar el funcionamiento del algoritmo de etiquetado, observe el grafo ponderado no dirigido de la Ilustración 7(a), donde las ponderaciones representan, por ejemplo, distancias. Queremos encontrar la ruta más corta posible de *A* a *D*. Comenzamos por marcar el nodo *A* como permanente, lo cual se indica mediante un círculo relleno. Después examinamos, por turno, cada uno de los nodos adyacentes a *A* (el nodo de trabajo) y reetiquetamos cada uno de ellos con la distancia a *A*. Cada vez que reetiquetamos un nodo, también lo etiquetamos con el nodo desde el que se hizo la prueba, para poder reconstruir más tarde la ruta final. Si la red tuviera más de una ruta más corta de *A* a *D* y quisiéramos encontrarlas todas, tendríamos que recordar todos los nodos de prueba que podrían llegar a un nodo con la misma distancia.

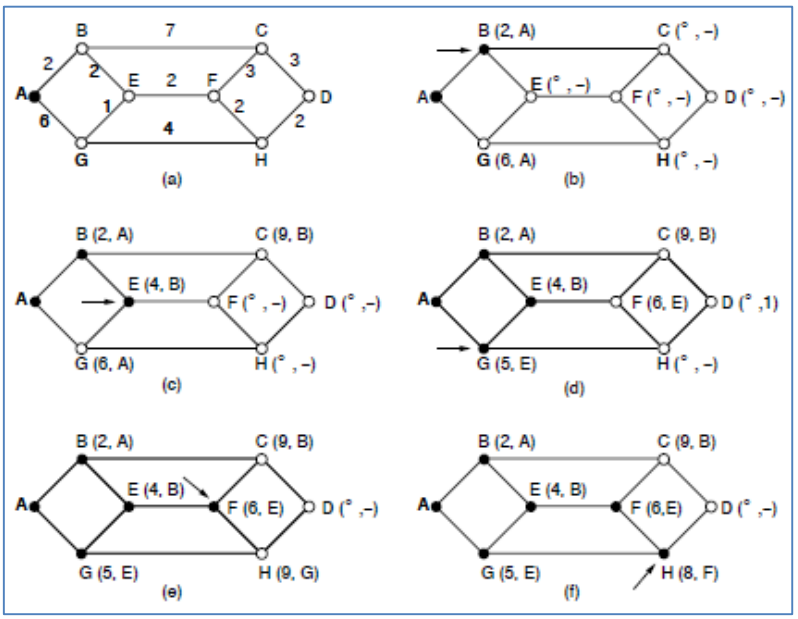


Ilustración 7 - Los primeros seis pasos utilizados para calcular la ruta más corta de A a D. Las flechas indican el nodo de trabajo.

Una vez que terminamos de examinar cada uno de los nodos adyacentes a *A*, examinamos todos los nodos etiquetados tentativamente en el grafo completo y hacemos permanente el de la etiqueta más pequeña, como se muestra en la Ilustración 7(b). Éste se convierte en el nuevo nodo de trabajo.

Ahora comenzamos por B y examinamos todos los nodos adyacentes a él. Si la suma de la etiqueta en *B* y la distancia desde *B* hasta el nodo en consideración es menor que la etiqueta de ese nodo, tenemos una ruta más corta, por lo que reetiquetamos ese nodo.

Después de inspeccionar todos los nodos adyacentes al nodo de trabajo y cambiar las etiquetas tentativas si es posible, se busca en el grafo completo el nodo etiquetado tentativamente con el menor valor. Este nodo se hace permanente y se convierte en el nodo de trabajo para la siguiente ronda. En la Ilustración 7 se muestran los primeros seis pasos del algoritmo.

Para ver por qué funciona el algoritmo, vea la Ilustración 7(c). En ese punto acabamos de hacer permanente a E. Suponga que hubiera una ruta más corta que *ABE*, digamos *AXYZE* (para alguna *X* y *Y*). Hay dos posibilidades: el nodo *Z* ya se hizo o no permanente. Si ya es permanente, entonces *E* ya se probó (en la ronda que siguió a aquella en la que *Z* se hizo permanente), por lo que la ruta *AXYZE* no ha escapado de nuestra atención y, por lo tanto, no puede ser una ruta más corta.

Ahora considere el caso en el que *Z* aún está etiquetado de forma tentativa. Si la etiqueta en *Z* es mayor o igual que la de *E*, entonces *AXYZE* no puede ser una ruta más corta que *ABE*. Si la etiqueta es menor que la de *E*, entonces *Z* y no *E* se volverá permanente primero, lo que permitirá probar a *E* desde *Z*.

Este algoritmo se muestra en la Ilustración 8. Las variables globales *n* y *dist* describen el grafo y se inicializan antes de llamar a *shortest_path*. La única diferencia entre el programa y el algoritmo antes descrito es que, en la Ilustración 8, calculamos la ruta más corta posible comenzando por el nodo terminal *t* en lugar del nodo de origen, *s*.

Debido a que las rutas más cortas desde *t* a *s* en un grafo no dirigido son iguales que las rutas más cortas de *s* a *t*, no importa en qué extremo empecemos. La razón de buscar hacia atrás es que cada nodo está etiquetado con su antecesor en vez de su sucesor. Al copiar la ruta final en la variable de salida, *path*, la ruta de salida se invierte. Al pasar esto, ambos efectos se cancelan y la respuesta se produce en el orden correcto.

```
#define MAX_NODES 1024                          /* número máximo de nodos */
#define INFINITY 1000000000                     /* un número mayor que cualquier ruta máxima */
int n, dist[MAX_NODES][MAX_NODES];              /* dist[i][j] es la distancia de i a j */

void shortest_path(int s, int t, int path[])
{ struct state {                                /* la ruta en la que se está trabajando */
    int predecessor;                            /* nodo previo */
    int length;                                 /* longitud del origen a este nodo */
    enum {permanent, tentative} label;          /* estado de la etiqueta */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) {     /* estado de inicialización*/
    p->predecessor = -1;
    p->length = INFINITY;
    p->label = tentative;
  }
  state[t].length = 0; state[t].label = permanent;
  k = t;                                        /* k es el nodo de trabajo inicial */
  do{                                           /* ¿hay una ruta mejor desde k? */
    for (i = 0; i < n; i++)                     /* este grafo tiene n nodos */
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }

    /* Encuentra el nodo etiquetado tentativamente con la etiqueta menor. */
    k = 0; min = INFINITY;
    for (i = 0; i < n; i++)
        if (state[i].label == tentative && state[i].length < min) {
            min = state[i].length;
            k = i;
        }
    state[k].label = permanent;
  } while (k != s);

  /* Copia la ruta en el arreglo de salida. */
  i = 0; k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Ilustración 8 - Algoritmo de Dijkstra para calcular la ruta más corta a través de un grafo.

Inundación

Cuando se implementa un algoritmo de enrutamiento, cada enrutador debe tomar decisiones con base en el conocimiento local, no en la imagen completa de la red. Una técnica local simple es la **inundación**, en la que cada paquete entrante se envía en todas las líneas de salida, excepto en la línea por la que llegó.

Sin duda la inundación genera grandes cantidades de paquetes duplicados; de hecho, puede ser una cantidad infinita a menos que se tomen algunas medidas para limitar el proceso. Una de estas medidas es integrar un contador de saltos al encabezado de cada paquete, que disminuya con cada salto, y que el paquete se descarte cuando el contador llegue a cero. Lo ideal es inicializar el contador de saltos con la longitud de la ruta entre el origen y el destino. Si el emisor desconoce el tamaño de la ruta, puede inicializar el contador para el peor caso, igual al diámetro total de la red.

La inundación con un conteo de saltos puede producir un número exponencial de paquetes duplicados a medida que aumenta el conteo de saltos y los enrutadores duplican los paquetes que ya han visto antes. Una técnica mejorada para ponerle freno a la inundación es llevar un registro de los paquetes difundidos en la

inundación, para evitar enviarlos una segunda vez. Una manera de lograr este objetivo es hacer que el enrutador de origen ponga un número de secuencia en cada paquete que reciba de sus hosts. Así, cada enrutador necesita una lista por cada enrutador de origen que indique los números de secuencia originados en ese enrutador que ya ha visto. Si un paquete de entrada está en la lista, no se difunde mediante inundación.

Para evitar que la lista crezca sin límites, cada lista debe aumentar mediante un contador *k* que indique que ya se han visto todos los números de secuencia hasta *k*. Cuando llega un paquete, es fácil comprobar si éste ya se difundió mediante inundación (se compara su número de secuencia con *k*); de ser así, se descarta. Lo que es más, no se necesita la lista completa por debajo de *k*, ya que *k* la resume de manera efectiva.

La inundación no es práctica para enviar la mayoría de los paquetes, pero tiene algunos usos importantes. En primer lugar, asegura que un paquete se entregue en todos los nodos de la red. Esto podría ser un desperdicio si sólo hay un destino que necesite el paquete, pero es efectivo para difundir información. En las redes inalámbricas, todos los mensajes transmitidos por una estación los pueden recibir todas las demás estaciones dentro de su alcance de radio, lo cual de hecho se puede considerar como inundación; algunos algoritmos usan esta propiedad.

En segundo lugar, la inundación es en extremo robusta. Incluso si grandes cantidades de enrutadores vuelan en pedazos (por ejemplo, en una red militar ubicada en una zona de guerra), la inundación encontrará una ruta, si es que existe, para transmitir un paquete a su destino. Además, la inundación requiere muy poca configuración. Los enrutadores sólo necesitan conocer a sus vecinos. Esto significa que la inundación se puede usar como bloque de construcción para otros algoritmos de enrutamiento que son más eficientes pero requieren una configuración un poco más complicada. La inundación también se puede usar como métrica con la que se pueden comparar otros algoritmos de enrutamiento. La inundación siempre selecciona la ruta más corta debido a que selecciona todas las rutas posibles en paralelo. En consecuencia, ningún otro algoritmo puede producir un retardo más corto (si ignoramos la sobrecarga generada por el proceso de inundación mismo).

Enrutamiento por vector de distancia

Por lo general, las redes de computadoras utilizan algoritmos de enrutamiento dinámico más complejos que la inundación, pero más eficientes debido a que encuentran las rutas más cortas para la topología actual. En particular hay dos algoritmos dinámicos que son los más populares: el enrutamiento por vector de distancia y el enrutamiento por estado del enlace.

El algoritmo de **enrutamiento por vector de distancia** opera haciendo que cada enrutador mantenga una tabla (es decir, un **vector**) que proporcione la mejor distancia conocida a cada destino y el enlace que se puede usar para llegar ahí. Para actualizar estas tablas se intercambia información con los vecinos. Eventualmente, todos los ruteadores conocen el mejor enlace para alcanzar cada destino.

En ocasiones, el algoritmo de enrutamiento por vector de distancia recibe otros nombres, de los cuales el más común es algoritmo de enrutamiento **Bellman-Ford distribuido**, en honor a los investigadores que lo desarrollaron. Éste fue el algoritmo original de enrutamiento de ARPANET y también se usó en Internet con el nombre de RIP.

En el enrutamiento por vector de distancia, cada enrutador mantiene una tabla de enrutamiento indizada por (y que contiene una entrada de) cada enrutador de la red. Esta entrada consta de dos partes: la línea preferida de salida a usar para ese destino y una estimación del tiempo o distancia a ese destino. La distancia se podría medir como la cantidad de saltos, o se podría usar otra métrica, como vimos al calcular las rutas más cortas.

Se supone que el enrutador conoce la "*distancia*" a cada uno de sus vecinos. Si la métrica es de saltos, la distancia sólo es un salto. Si la métrica es el retardo de propagación, el enrutador puede medirlo en forma directa con paquetes especiales de ECO que el receptor sólo marca con la hora y lo regresa tan rápido como puede.

Por ejemplo, suponga que el retardo se usa como métrica y que el enrutador conoce el retardo a cada uno de sus vecinos. Una vez cada *T* ms, cada enrutador envía a todos sus vecinos una lista de sus retardos estimados a cada destino. También recibe una lista similar de cada vecino. Imagine que una de estas tablas acaba de

llegar del vecino *X*, en donde X_i es la estimación respecto al tiempo que le toma llegar al enrutador *i*. Si el enrutador sabe que el retardo a *X* es de *m* ms, también sabe que puede llegar al enrutador *i* a través de *X* en X_i+m ms. Al efectuar este cálculo para cada vecino, un enrutador puede encontrar la estimación que parezca ser la mejor y usar esa estimación, así como el enlace correspondiente, en su nueva tabla de enrutamiento. Cabe mencionar que en este cálculo no se utiliza la antigua tabla de enrutamiento.

Este proceso de actualización se ilustra en la Ilustración 9. En la parte *(a)* se muestra una red. Las primeras cuatro columnas de la parte *(b)* muestran los vectores de retardo recibidos de los vecinos del enrutador *J*. *A* indica que tiene un retardo de 12 ms a *B*, un retardo de 25 ms a *C*, un retardo de 40 ms a *D*, etcétera. Suponga que *J* ha medido o estimado el retardo a sus vecinos *A*, *I*, *H* y *K* en 8, 10, 12 y 6 ms, respectivamente.

Considere la manera en que *J* calcula su nueva ruta al enrutador *G*. Sabe que puede llegar a *A* en 8 ms; además *A* indica ser capaz de llegar a *G* en 18 ms, por lo que *J* sabe que puede contar con un retardo de 26 ms a *G* si reenvía a *A* los paquetes destinados para *G*. Del mismo modo, calcula el retardo a *G* a través de *I*, *H* y *K* en 41 (31+10), 18 (6+12) y 37 (31+6) ms, respectivamente. El mejor de estos valores es el 18, por lo que escribe una entrada en su tabla de enrutamiento para indicar que el retardo a *G* es de 18 ms, y que la ruta que se utilizará es a través de *H*. Para los demás destinos se realiza el mismo cálculo; la nueva tabla de enrutamiento se muestra en la última columna de la figura.

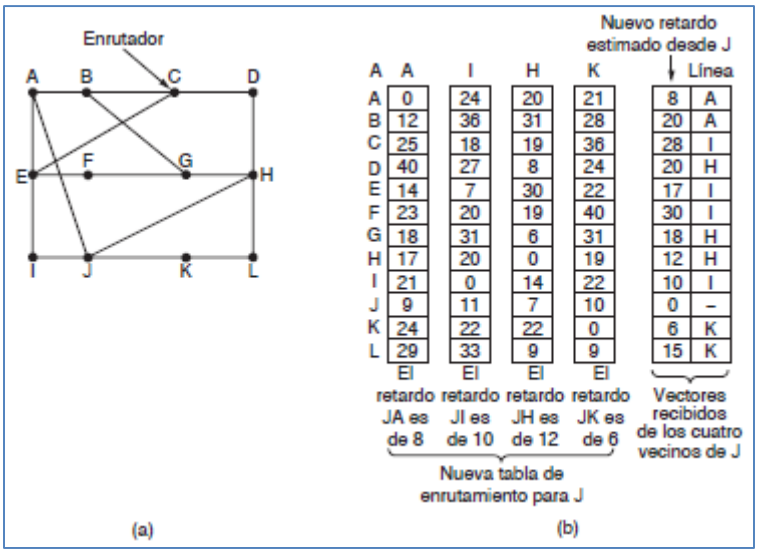


Ilustración 9 - (a) Una red. (b) Entrada de A, I, H, K y la nueva tabla de enrutamiento para J.

El problema del conteo al infinito

Al proceso de establecer los enrutamientos con base en las mejores rutas a través de la red se le conoce como **convergencia**. El enrutamiento por vector de distancia es útil como una simple técnica mediante la cual los enrutadores pueden calcular las rutas más cortas en forma colectiva, pero tiene una grave desventaja en la práctica: aunque converge hacia la respuesta correcta, puede llegar a hacerlo con lentitud. Básicamente reacciona rápido a las buenas noticias, pero es lento con las malas. Considere un enrutador cuya mejor ruta al destino *X* es larga. Si en el siguiente intercambio el vecino *A* informa de manera repentina un retardo corto a *X*, el enrutador simplemente cambia y usa la línea *A* para enviar tráfico a *X*. En un intercambio de vectores, se procesan las buenas noticias.

Para ver la rapidez de propagación de las buenas noticias, considere la red de cinco nodos (lineal) de la Ilustración 10, en donde la métrica de retardo es el número de saltos. Suponga que *A* está desactivado al principio y que los otros enrutadores lo saben. En otras palabras, todos registraron el retardo a *A* como infinito.

Al activarse *A*, los demás enrutadores saben de él gracias a los intercambios de vectores. Por sencillez, supondremos que hay un indicador periódico, que marca el compás para iniciar de manera simultánea un intercambio de vectores entre todos los enrutadores. En el momento del primer intercambio, *B* se entera de que su vecino de la izquierda tiene un retardo de cero hacia *A*. *B* crea entonces una entrada en su tabla de enrutamiento para indicar que *A* está a un salto de distancia hacia la izquierda. Los demás enrutadores aún

detectan que *A* está desactivado. En este punto, las entradas de la tabla de enrutamiento para *A* se muestran en la segunda fila de la Ilustración 10(a). Durante el siguiente intercambio, *C* se entera de que *B* tiene una ruta a *A* de longitud 1, por lo que actualiza su tabla de enrutamiento para indicar una ruta de longitud 2, pero *D* y *E* no se enteran de las buenas nuevas sino hasta después. Como es evidente, las buenas noticias se difunden a razón de un salto por intercambio. En una red cuya ruta mayor tiene una longitud de *N* saltos, en un lapso de *N* intercambios todo mundo sabrá sobre los enlaces y enrutadores que acaban de revivir.

Ahora consideremos la situación de la Ilustración 10(b), en la que todos los enlaces y enrutadores están inicialmente activos. Los enrutadores *B*, *C*, *D* y *E* tienen distancias a *A* de 1, 2, 3 y 4 saltos, respectivamente. De pronto, *A* se desactiva o bien se corta el enlace entre *A* y *B* (que de hecho es lo mismo desde el punto de vista de *B*).

En el primer intercambio de paquetes, *B* no escucha nada de *A*. Por fortuna, *C* dice: “No te preocupes. Tengo una ruta hacia *A* de longitud 2”. *B* no sabe que la ruta de *C* pasa a través de *B* mismo. Hasta donde *B* sabe, *C* puede tener 10 líneas, todas con rutas independientes hacia *A* de longitud 2. Como resultado, ahora *B* piensa que puede llegar a *A* por medio de *C*, con una longitud de ruta de 3. *D* y *E* no actualizan sus entradas para *A* en el primer intercambio.

En el segundo intercambio, *C* nota que cada uno de sus vecinos afirma tener una ruta de longitud 3 hacia *A*. *C* escoge una de ellas al azar y hace que su nueva distancia hacia *A* sea de 4, como se muestra en la tercera fila de la Ilustración 10(b). Los intercambios subsecuentes producen la historia que se muestra en el resto de la Ilustración 10(b).

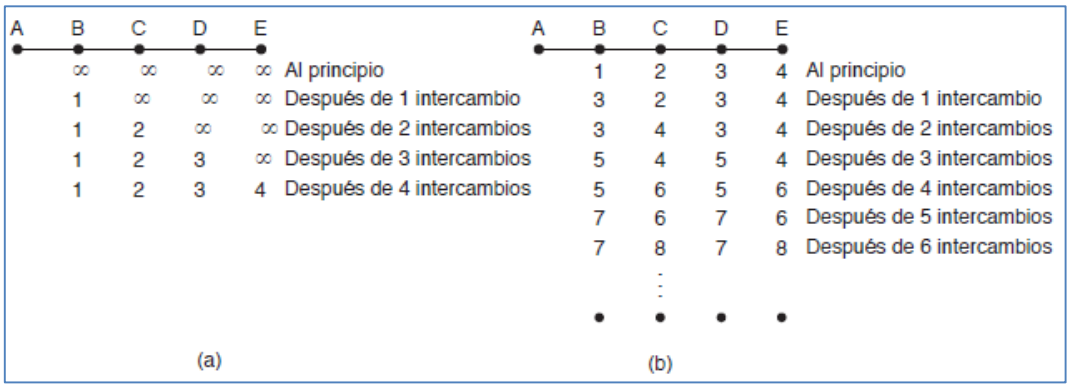


Ilustración 10 - El problema del conteo al infinito.

A partir de esta figura debe quedar clara la razón por la que las malas noticias viajan con lentitud: no hay ningún enrutador que tenga en algún momento un valor mayor en más de una unidad que el mínimo de todos sus vecinos. Gradualmente, todos los enrutadores elevan su cuenta hacia el infinito, pero el número de intercambios requerido depende del valor numérico usado para el infinito. Por esta razón, es prudente hacer que el infinito sea igual a la ruta más larga, más 1.

No es del todo sorprendente que a éste se le conozca como el **problema del conteo al infinito**. Se han dado muchos intentos por resolverlo; por ejemplo, evitar que los enrutadores anuncien sus mejores rutas de vuelta a los vecinos de quienes las escucharon mediante la regla del horizonte dividido con envenenamiento en reversa que se describe en el RFC 1058. Sin embargo, ninguna de estos métodos funciona bien en la práctica a pesar de los nombres tan coloridos. El núcleo del problema es que, cuando *X* dice a *Y* que tiene una ruta hacia alguna parte, *Y* no tiene forma de saber si él mismo está en esa ruta.

Enrutamiento por estado del enlace

El enrutamiento por vector de distancia se utilizó en ARPANET hasta 1979, cuando se reemplazó por el enrutamiento por estado del enlace. El principal problema que provocó su desaparición era que, con frecuencia, el algoritmo tardaba demasiado en converger una vez que cambiaba la topología de la red (debido al problema del conteo al infinito). En consecuencia, se reemplazó por un algoritmo totalmente nuevo, ahora conocido como **enrutamiento por estado del enlace**. Las variantes de este enrutamiento llamadas IS-IS y OSPF son los algoritmos de enrutamiento más utilizados dentro de las redes extensas y de Internet en la actualidad.

La idea detrás del enrutamiento por estado del enlace es bastante simple y se puede enunciar en cinco partes. Cada enrutador debe realizar lo siguiente para hacerlo funcionar:

1. Descubrir a sus vecinos y conocer sus direcciones de red.
2. Establecer la métrica de distancia o de costo para cada uno de sus vecinos.
3. Construir un paquete que indique todo lo que acaba de aprender.
4. Enviar este paquete a todos los demás enrutadores y recibir paquetes de ellos.
5. Calcular la ruta más corta a todos los demás enrutadores.

De hecho, se distribuye la topología completa a todos los enrutadores. Después se puede ejecutar el algoritmo de Dijkstra en cada enrutador para encontrar la ruta más corta a los demás enrutadores. A continuación veremos con mayor detalle cada uno de estos cinco pasos.

Aprender sobre los vecinos

Cuando un enrutador se pone en funcionamiento, su primera tarea es averiguar quiénes son sus vecinos. Para lograr esto envía un paquete especial *HELLO* en cada línea punto a punto. Se espera que el enrutador del otro extremo regrese una respuesta en la que indique su nombre. Estos nombres deben ser globalmente únicos puesto que, cuando un enrutador distante escucha después que hay tres enrutadores conectados a *F*, es indispensable que pueda determinar si los tres se refieren al mismo *F*.

Cuando se conectan dos o más enrutadores mediante un enlace de difusión (por ejemplo, un switch, anillo o Ethernet clásica), la situación es un poco más complicada. En la Ilustración 11(a) se ilustra una LAN de difusión a la que están conectados de forma directa tres enrutadores, *A*, *C* y *F*. Cada uno de estos enrutadores está conectado a uno o más enrutadores adicionales, como se muestra.

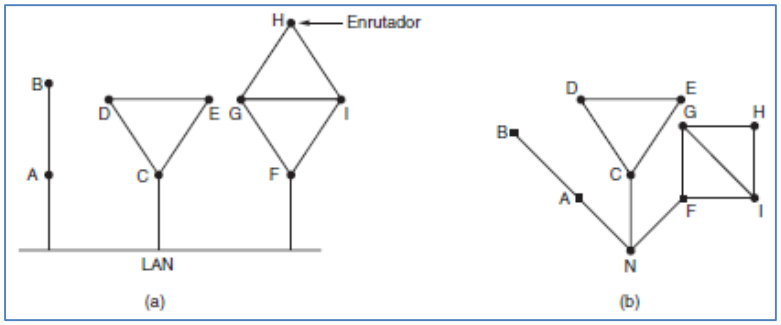


Ilustración 11 - (a) Nueve enrutadores y una LAN de difusión. (b) Modelo de grafo de (a).

La LAN de difusión proporciona conectividad entre cada par de enrutadores conectados. Sin embargo, al modelar la LAN como muchos enlaces punto a punto se incrementa el tamaño de la topología, lo cual conduce a mensajes desperdiciados. Una mejor manera de modelar la LAN, es considerarla como un nodo más, como se muestra en la Ilustración 11(b). Aquí hemos introducido un nodo artificial nuevo, *N*, al que están conectados *A*, *C* y *F*. Se selecciona un enrutador designado en la LAN para que desempeñe el papel de *N* en el protocolo de enrutamiento. El hecho de que sea posible ir de *A* a *C* en la LAN se representa aquí mediante la ruta ANC.

Establecimiento de los costos de los enlaces

El algoritmo de enrutamiento por estado del enlace requiere que cada enlace tenga una métrica de distancia o costo para encontrar las rutas más cortas. El costo para llegar a los vecinos se puede establecer de modo automático, o el operador de red lo puede configurar. Una elección común es hacer el costo inversamente proporcional al ancho de banda del enlace. Por ejemplo, una red Ethernet de 1 Gbps puede tener un costo de 1 y una red Ethernet de 100 Mbps un costo de 10. Esto hace que las rutas de mayor capacidad sean mejores opciones.

Si la red está geográficamente dispersa, el retardo de los enlaces se puede considerar en el costo, de modo que las rutas a través de enlaces más cortos sean mejores opciones. La manera más directa de determinar este retardo es enviar un paquete especial ECO a través de la línea, que el otro extremo tendrá que regresar de inmediato. Si se mide el tiempo de ida y vuelta, y se divide entre dos, el enrutador emisor puede obtener una estimación razonable del retardo.

Construcción de los paquetes de estado del enlace

Una vez que se ha recabado la información necesaria para el intercambio, el siguiente paso es que cada enrutador construya un paquete que contenga todos los datos. El paquete comienza con *la identidad del emisor*, seguida de un *número de secuencia*, una *edad* (que describiremos después) y una *lista de vecinos*. También se proporciona el *costo para cada vecino*. En la Ilustración 12(a) se muestra una red de ejemplo; los costos se muestran como etiquetas en las líneas. En la Ilustración 12(b) se muestran los paquetes de estado del enlace correspondientes para los seis enrutadores.

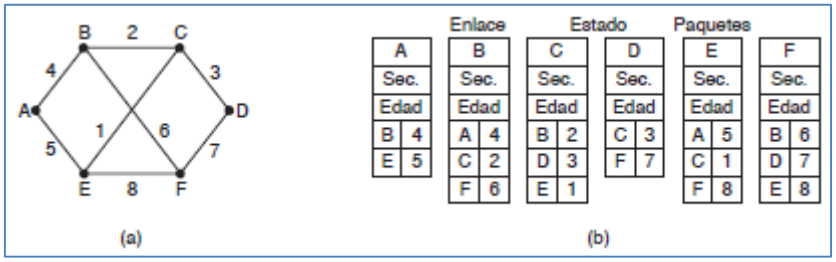


Ilustración 12 - (a) Una red. (b) Los paquetes de estado del enlace para esta red.

Es fácil construir los paquetes de estado del enlace. La parte difícil es determinar cuándo construirlos. Una posibilidad es construirlos de manera periódica; es decir, a intervalos regulares. Otra posibilidad es construirlos cuando ocurra un evento significativo, como la caída o la reactivación de una línea o de un vecino, o cuando sus propiedades cambien en forma considerable.

Distribución de los paquetes de estado del enlace

La parte más complicada del algoritmo es la distribución de los paquetes de estado del enlace. Todos los enrutadores deben recibir todos los paquetes de estado del enlace con rapidez y confiabilidad. Si se utilizan distintas versiones de la topología, las rutas que se calculen podrían tener inconsistencias como ciclos, máquinas inalcanzables y otros problemas.

Primero describiremos el algoritmo básico de distribución. Después le aplicaremos algunos refinamientos. La idea fundamental es utilizar inundación para distribuir los paquetes de estado del enlace a todos los enrutadores. Con el fin de mantener controlada la inundación, cada paquete contiene un número de secuencia que se incrementa con cada nuevo paquete enviado. Los enrutadores llevan el registro de todos los pares (enrutador de origen, secuencia) que ven. Cuando llega un nuevo paquete de estado del enlace, se verifica y compara con la lista de paquetes ya vistos. Si es nuevo, se reenvía a través de todas las líneas, excepto aquella por la que llegó. Si es un duplicado, se descarta. Si llega un paquete con número de secuencia menor que el mayor visto hasta el momento, se rechaza como obsoleto debido a que el enrutador tiene datos más recientes.

Este algoritmo tiene algunos problemas, pero son manejables. Primero, si los números de secuencia vuelven a comenzar, reinará la confusión. La solución aquí es utilizar un número de secuencia de 32 bits. Con un paquete de estado del enlace por segundo, el tiempo para volver a empezar será de 137 años, por lo que podemos ignorar esta posibilidad.

Segundo, si llega a fallar un enrutador, perderá el registro de su número de secuencia. Si comienza nuevamente en 0, se rechazará como duplicado el siguiente paquete que envíe.

Tercero, si llega a corromperse un número de secuencia y se recibe 65540 en vez de 4 (un error de 1 bit), los paquetes 5 a 65540 se rechazarán como obsoletos, dado que se piensa que el número de secuencia actual es 65540.

La solución a todos estos problemas es incluir la edad de cada paquete después del número de secuencia y disminuirla una vez cada segundo. Cuando la edad llega a cero, se descarta la información de ese enrutador. Por lo general, un paquete nuevo entra, por ejemplo, cada 10 segundos, por lo que la información de los enrutadores sólo expira cuando un enrutador está caído (o cuando se pierden seis paquetes consecutivos, un evento poco probable). Los enrutadores también decrementan el campo *Edad* durante el proceso inicial de

inundación para asegurar que no pueda perderse ningún paquete y sobrevivir durante un periodo de tiempo indefinido (se descarta el paquete cuya edad sea cero).

Algunos refinamientos a este algoritmo pueden hacerlo más robusto. Una vez que llega un paquete de estado del enlace a un enrutador para ser inundado, no se encola para su transmisión de inmediato, sino que se coloca en un área de almacenamiento para esperar un tiempo corto, en caso de que se activen o desactiven más enlaces. Si llega otro paquete de estado del enlace proveniente del mismo origen antes de que se transmita el primer paquete, se comparan sus números de secuencia. Si son iguales, se descarta el duplicado. Si son diferentes, se desecha el más antiguo. Como protección contra los errores en los enlaces, se confirma la recepción de todos los paquetes de estado del enlace.

En la Ilustración 13 se describe la estructura de datos que utiliza el enrutador *B* para la red de la Ilustración 12(a). Cada fila aquí corresponde a un paquete de estado del enlace recién llegado, pero que aún no se procesa por completo. La tabla registra dónde se originó el paquete, su número de secuencia y edad, así como los datos. Además, hay banderas de envío y de confirmación de recepción para cada uno de los tres enlaces de *B* (a *A*, *C* y *F*, respectivamente). Las banderas de envío significan que el paquete debe enviarse a través del enlace indicado. Las banderas de confirmación de recepción significan que ahí se debe confirmar su recepción.

En la Ilustración 13, el paquete de estado del enlace de *A* llega de manera directa, por lo que se debe enviar a *C* y *F*, y debe confirmarse la recepción a *A*, como lo indican los bits de bandera. De la misma forma, es necesario reenviar el paquete de *F* a *A* y a *C*, y debe enviarse a *F* la confirmación de su recepción.

Sin embargo, la situación del tercer paquete, que proviene de *E*, es diferente. Llega dos veces, la primera a través de *EAB* y la segunda por medio de *EFB*. En consecuencia, este paquete tiene que enviarse sólo a *C*, pero se debe confirmar su recepción tanto a *A* como a *F*, como lo indican los bits.

			Banderas de envío			Banderas de ACK			
Origen	Sec.	Edad	A	C	F	A	C	F	Datos
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Ilustración 13 - El búfer de paquetes para el enrutador B de la Ilustración 12(a).

Si llega un duplicado mientras el original aún está en el búfer, se deben cambiar los bits. Por ejemplo, si llega una copia del estado de *C* desde *F* antes de que la cuarta entrada en la tabla haya sido reenviada, cambiarán los 6 bits a 100011 para indicar que se debe enviar a *F* una confirmación de recepción del paquete, pero sin enviarle el paquete mismo.

Cálculo de las nuevas rutas

Una vez que un enrutador ha acumulado un conjunto completo de paquetes de estado del enlace, puede construir el grafo de toda la red debido a que todos los enlaces están simbolizados. De hecho, cada enlace se representa dos veces, una para cada dirección. Las distintas direcciones pueden tener incluso costos diferentes. Así, los cálculos de la ruta más corta pueden encontrar rutas del enrutador *A* a *B* que sean distintas a las del enrutador de *B* a *A*.

Ahora se puede ejecutar localmente el algoritmo de Dijkstra para construir las rutas más cortas a todos los destinos posibles. Los resultados de este algoritmo indican al enrutador qué enlace debe usar para llegar a cada destino. Esta información se instala en las tablas de enrutamiento y se puede reanudar la operación normal.

En comparación con el enrutamiento por vector de distancia, el enrutamiento por estado del enlace requiere más memoria y poder de cómputo. Para una red con *n* enrutadores, cada uno de los cuales tiene *k* vecinos, la memoria requerida para almacenar los datos de entrada es proporcional a *kn*, que es por lo menos tan grande

como una tabla de enrutamiento que lista todos los destinos. Además, el tiempo de cómputo también crece con más rapidez que *kn*, incluso con las estructuras de datos más eficientes; un problema en las grandes redes. Sin embargo, en muchas situaciones prácticas, el enrutamiento por estado del enlace funciona bien debido a que no sufre de los problemas de convergencia lenta.

El enrutamiento por estado del enlace se usa ampliamente en las redes actuales, por lo que son pertinentes unas pocas palabras sobre algunos protocolos que lo usan. Muchos ISP usan el protocolo de estado del enlace **IS-IS** (Sistema Intermedio a Sistema Intermedio, del inglés *Intermediate System-Intermediate System*). Este protocolo fue diseñado para una de las primeras redes llamada *DECnet*; más tarde lo adoptó la ISO para utilizarlo con los protocolos OSI y después se modificó para manejar otros protocolos también, siendo IP el más importante de éstos. El otro protocolo de estado es **OSPF** (Abrir primero la ruta más corta, del inglés *Open Shortest Path First*), diseñado por el IETF varios años después de IS-IS y adoptó muchas de las innovaciones diseñadas para este último. Estas innovaciones incluyen un método de estabilización automática para inundar las actualizaciones de estado del enlace, el concepto de un enrutador designado en una LAN y el método para calcular y soportar la división de rutas y múltiples métricas. Como consecuencia, hay muy poca diferencia entre IS-IS y OSPF. La diferencia más importante es que IS-IS puede transportar información sobre varios protocolos de capa de red al mismo tiempo (por ejemplo, IP, IPX y AppleTalk). OSPF no tiene esta característica y es una ventaja en los grandes entornos multiprotocolo.

También es pertinente hacer un comentario general sobre los algoritmos de enrutamiento. Los algoritmos de estado del enlace, de vector de distancia y otros más dependen del procesamiento en todos los enrutadores para calcular las rutas. Los problemas con el hardware o el software, incluso en una pequeña cantidad de enrutadores, pueden causar estragos en la red. Por ejemplo, si un enrutador afirma tener un enlace que no tiene u olvida un enlace que sí tiene, el grafo de la red será incorrecto. Si un enrutador deja de reenviar paquetes, o los corrompe al hacerlo, la ruta no funcionará como se esperaba. Por último, si al enrutador se le acaba la memoria o se ejecuta mal el cálculo de enrutamiento, surgirán problemas. A medida que la red crece en decenas o cientos de miles de nodos, la probabilidad de que un enrutador tenga fallas ocasionales deja de ser insignificante. El truco es tratar de limitar el daño cuando ocurra lo inevitable.

Enrutamiento jerárquico

A medida que crece el tamaño de las redes, también lo hacen en forma proporcional las tablas de enrutamiento del enrutador. Las tablas que están en crecimiento constante no sólo consumen memoria del enrutador, sino que también se necesita más tiempo de CPU para examinarlas y más ancho de banda para enviar informes de estado entre enrutadores. En cierto momento, la red puede crecer hasta el punto en que ya no sea viable que cada enrutador tenga una entrada para cada uno de los demás enrutadores, por lo que el enrutamiento tendrá que hacerse de manera jerárquica, como ocurre en la red telefónica.

Cuando se utiliza el enrutamiento jerárquico, los enrutadores se dividen en lo que llamaremos **regiones**. Cada enrutador conoce todos los detalles para enrutar paquetes a destinos dentro de su propia región, pero no sabe nada de la estructura interna de las otras regiones. Cuando se interconectan diferentes redes, es natural considerar cada una como región independiente con el fin de liberar a los enrutadores de una red de la necesidad de conocer la estructura topológica de las demás redes.

En las redes enormes, tal vez no sea suficiente una jerarquía de dos niveles; puede ser necesario agrupar las regiones en **clústeres**, los clústeres en **zonas**, las zonas en **grupos**, etc, hasta que se nos agoten los nombres para clasificarlos. Como ejemplo de jerarquía multinivel, considere una posible forma de enrutar un paquete de Berkeley, California, a Malindi, Kenia. El enrutador de Berkeley podría conocer la topología detallada de California, pero tendría que enviar todo el tráfico exterior al enrutador de Los Ángeles. El enrutador de Los Ángeles podría enrutar el tráfico directamente a otros enrutadores del país, pero tendría que enviar el tráfico internacional a Nueva York. El enrutador de Nueva York podría estar programado para dirigir todo el tráfico al enrutador del país de destino encargado del manejo de tráfico internacional, por decir, en Nairobi. Por último, el paquete encontraría su camino por el árbol de Kenia hasta llegar a Malindi.

En la Ilustración 14 aparece un ejemplo cuantitativo de enrutamiento en una jerarquía de dos niveles con cinco regiones. La tabla de enrutamiento completa para el enrutador *1A* tiene 17 entradas, como se muestra en la Ilustración 14(b). Si el enrutamiento se hace en forma jerárquica, como en la Ilustración 14(c), hay entradas

para todos los enrutadores locales, igual que antes, pero las demás regiones se condensan en un solo enrutador, de modo que todo el tráfico para la región 2 viaja a través de la línea *1B-2A*, pero el resto del tráfico remoto viaja por la línea *1C-3B*. El enrutamiento jerárquico redujo la tabla de 17 entradas a 7. A medida que crece la razón entre la cantidad de regiones y el número de enrutadores por región, aumentan los ahorros de espacio en la tabla.

Por desgracia, estas ganancias de espacio no son gratuitas. Hay que pagar un precio: un incremento en la longitud de la ruta. Por ejemplo, la mejor ruta de *1A* a *5C* es a través de la región 2, pero con el enrutamiento jerárquico, todo el tráfico a la región 5 pasa a través de la región 3, porque eso es mejor para la mayoría de los destinos de la región 5.

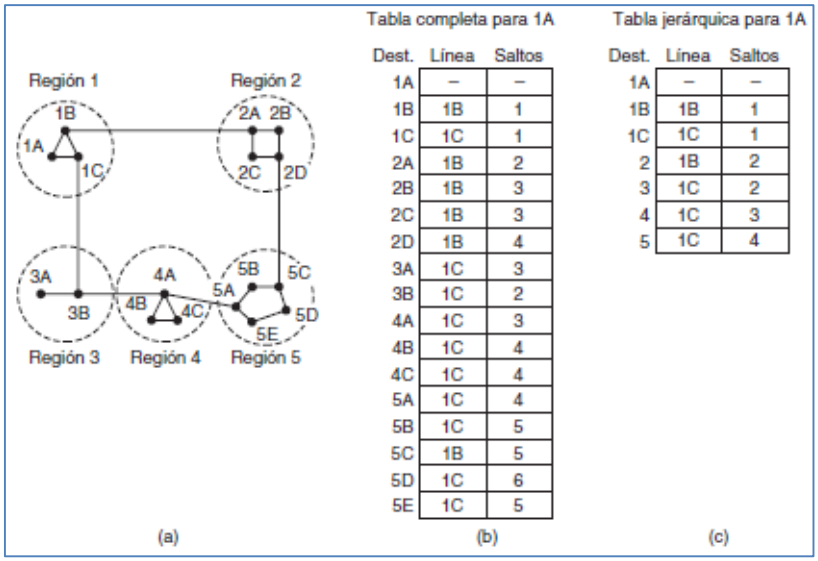


Ilustración 14 - Enrutamiento jerárquico.

Cuando una sola red se vuelve muy grande, surge una pregunta interesante: ¿cuántos niveles debe tener la jerarquía? Por ejemplo, considere una red con 720 enrutadores. Si no hay jerarquía, cada enrutador necesita 720 entradas en la tabla de enrutamiento. Si dividimos la red en 24 regiones de 30 enrutadores cada una, cada enrutador necesitará 30 entradas locales más 23 entradas remotas, lo que da un total de 53 entradas. Si elegimos una jerarquía de tres niveles con 8 clústeres, cada uno de los cuales contiene 9 regiones de 10 enrutadores, cada enrutador necesita 10 entradas para los enrutadores locales; 8, para el enrutamiento a otras regiones dentro de su propio clúster, y 7, para clústeres distantes, lo que da un total de 25 entradas. Kamoun y Kleinrock descubrieron que el número óptimo de niveles para una red de *N* enrutadores es de *ln(N)*, y se requieren un total de *eln(N)* entradas por enrutador. Ellos también demostraron que el aumento en la longitud promedio efectiva de la ruta causado por el enrutamiento jerárquico es tan pequeño que por lo general es aceptable.

Enrutamiento por difusión

En algunas aplicaciones, los hosts necesitan enviar mensajes a varios o a todos los hosts en la red. Por ejemplo, el servicio de distribución de informes sobre el clima o los programas de radio en vivo podrían funcionar mejor si se difunden a todas las máquinas para dejar que las personas interesadas lean los datos. El envío simultáneo de un paquete a todos los destinos se llama **difusión** (*broadcasting*). Se han propuesto varios métodos para llevarla a cabo.

Un método de difusión que no requiere características especiales de la red es que el origen sólo envíe un paquete distinto a cada destino. El método no sólo desperdicia ancho de banda y es lento, sino que también requiere que el origen tenga una lista completa de todos los destinos. En la práctica este método no es deseable, aun cuando tiene muchas aplicaciones.

Una mejora del algoritmo anterior es el **enrutamiento multidestino**, en donde cada paquete contiene una lista de destinos o un mapa de bits que indica los destinos deseados. Cuando un paquete llega al enrutador,

éste revisa todos los destinos para determinar el conjunto de líneas de salida que necesitará (se requiere una línea de salida si es la mejor ruta a cuando menos uno de los destinos). El enrutador genera una nueva copia del paquete para cada línea de salida que se utilizará, e incluye en cada paquete sólo aquellos destinos que utilizarán la línea. En efecto, el grupo de destinos se divide entre las líneas de salida. Después de una cantidad suficiente de saltos, cada paquete llevará sólo un destino, como un paquete normal. El enrutamiento multidestino es como los paquetes con direccionamiento individual, sólo que cuando varios paquetes deben seguir la misma ruta, uno de ellos paga la tarifa completa y los demás viajan gratis. Por lo tanto, el ancho de banda se utiliza con más eficiencia. Sin embargo, este esquema aún requiere que el origen conozca todos los destinos, además de que representa el mismo trabajo para un enrutador determinar a dónde debe enviar un paquete multidestino que varios paquetes distintos.

Ya hemos visto una mejor técnica de enrutamiento por difusión: la inundación. Cuando se implementa con un número de secuencia por cada origen, la inundación usa los enlaces de manera eficiente con una regla de decisión en los enrutadores que es relativamente simple. Aunque la inundación es poco adecuada para la comunicación punto a punto ordinaria, para difusión puede merecer que se le considere con seriedad. Sin embargo, resulta ser que podemos hacer algo todavía mejor una vez que se han calculado las rutas más cortas para los paquetes regulares.

La idea del **reenvío por ruta invertida** (*reverse path forwarding*) es elegante y sencilla una vez planteada. Cuando llega un paquete difundido a un enrutador, éste lo revisa para ver si llegó por el enlace que se usa por lo común para enviar paquetes hacia el origen de la difusión. De ser así, hay excelentes posibilidades de que el paquete difundido haya seguido la mejor ruta desde el enrutador y, por lo tanto, sea la primera copia en llegar al enrutador. Si éste es el caso, el enrutador reenvía copias del paquete a todos los enlaces, excepto a aquel por el que llegó. No obstante, si el paquete difundido llegó por un enlace diferente del preferido para llegar al origen, el paquete se descarta como probable duplicado.

En la Ilustración 15 se muestra un ejemplo del reenvío por ruta invertida. En la parte *(a)* se muestra una red; en la parte *(b)*, un árbol sumidero para el enrutador *I* de esa red y en la parte *(c)*, el funcionamiento del algoritmo de ruta invertida. En el primer salto, *I* envía paquetes a *F*, *H*, *J* y *N*, como lo indica la segunda fila del árbol. Cada uno de estos paquetes llega a *I* por la ruta preferida (suponiendo que la ruta preferida pasa a través del árbol sumidero), como lo indica el círculo alrededor de la letra. En el segundo salto se generan ocho paquetes, dos por cada uno de los enrutadores que recibieron un paquete en el primer salto. Como resultado los ocho llegan a enrutadores no visitados con anterioridad, y cinco llegan a través de la línea preferida. De los seis paquetes generados en el tercer salto, sólo tres llegan por la ruta preferida (a *C*, *E* y *K*); los otros son duplicados. Después de cinco saltos y 24 paquetes, termina la difusión, en comparación con cuatro saltos y 14 paquetes si se hubiera seguido exactamente el árbol sumidero.

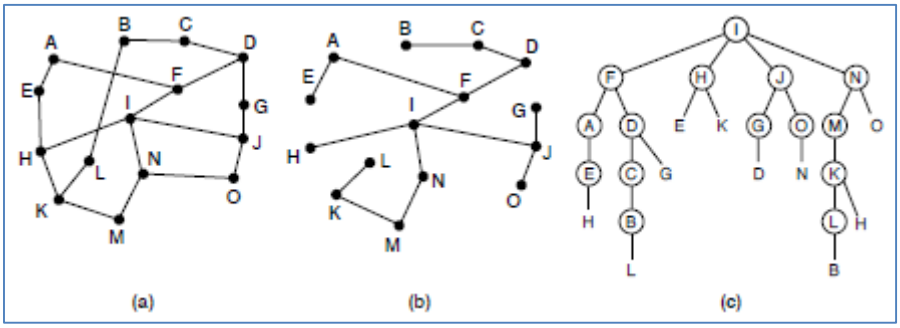


Ilustración 15 - Reenvío por ruta invertida. (a) Una red. (b) Un árbol sumidero. (c) El árbol construido mediante reenvío por ruta invertida.

La ventaja principal del reenvío por ruta invertida es que es tanto eficiente como fácil de implementar. Envía el paquete de difusión a través de cada enlace sólo una vez en cada dirección, como en la inundación, pero sólo requiere que los enrutadores sepan cómo llegar a todos los destinos, sin que necesiten recordar los números de secuencia (o usar otros mecanismos para detener la inundación) o listar todos los destinos en el paquete.

Nuestro último algoritmo de difusión mejora el comportamiento del reenvío por ruta invertida. Usa de manera explícita el árbol sumidero (o cualquier otro árbol de expansión conveniente) para el enrutador que inicia la difusión. Un **árbol de expansión** es un subconjunto de la red que incluye todos los enrutadores pero no contiene ciclos. Los árboles sumidero son árboles de expansión. Si cada enrutador sabe cuáles de sus líneas pertenecen al árbol de expansión, puede copiar un paquete de difusión entrante en todas las líneas del árbol de expansión, excepto en aquella por la que llegó. Este método utiliza de manera óptima el ancho de banda, ya que genera el número mínimo absoluto de paquetes necesarios para llevar a cabo el trabajo. En la Ilustración 15, por ejemplo, cuando el árbol de sumidero de la parte *(b)* es usado como el árbol de expansión, los paquetes de difusión son enviados con un mínimo de 14 paquetes. El único problema es que cada enrutador debe tener conocimiento de algún árbol de expansión para que este método pueda funcionar. Algunas veces esta información está disponible (por ejemplo, con el enrutamiento por estado del enlace, todos los enrutadores conocen la topología completa, por lo que pueden calcular un árbol de expansión), pero a veces no (por ejemplo, con el enrutamiento por vector de distancia).

Enrutamiento multidifusión

Algunas aplicaciones, como un juego multijugador o un video en vivo de un evento deportivo que se transmite por flujo continuo a muchas ubicaciones, envían paquetes a múltiples receptores. A menos que el grupo sea muy pequeño, es costoso enviar un paquete distinto a cada receptor. Por otro lado, sería un desperdicio difundir un paquete si el grupo consiste en, por decir, 1000 máquinas en una red con un millón de nodos, de tal forma que la mayoría de los receptores no están interesados en el mensaje (o peor aún, están en definitiva interesados pero se supone que no deben verlo). Por lo tanto, necesitamos una manera de enviar mensajes a grupos bien definidos que sean grandes en número, pero pequeños en comparación con la totalidad de la red.

El proceso de enviar un mensaje a uno de tales grupos se denomina **multidifusión** (*multicasting*); el algoritmo de enrutamiento que se utiliza es el **enrutamiento por multidifusión**. Todos los esquemas de multidifusión requieren alguna forma de crear y destruir grupos, además de identificar qué enrutadores son miembros de un grupo. La forma de realizar estas tareas no le concierne al algoritmo de enrutamiento. Por ahora vamos a suponer que cada grupo se identifica mediante una dirección de multidifusión y que los enrutadores conocen los grupos a los que pertenecen.

Los esquemas de enrutamiento por multidifusión se basan en los esquemas de enrutamiento por difusión que ya estudiamos; envían paquetes a través de árboles de expansión para entregar esos paquetes a los miembros del grupo, al tiempo que optimizan el uso del ancho de banda. Sin embargo, el mejor árbol de expansión a usar depende de si el grupo es denso, con receptores esparcidos por toda la red o disperso, en donde gran parte de la red no pertenece al grupo. En esta sección consideraremos ambos casos.

Si el grupo es denso, la difusión es un buen comienzo debido a que transmite de manera eficiente el paquete a todas las partes de la red. Pero la difusión llegará a algunos enrutadores que no sean miembros del grupo, lo cual es un desperdicio. Una solución explorada es recortar el árbol de expansión de difusión, mediante la eliminación de enlaces que no conducen a los miembros. El resultado es un **árbol de expansión de multidifusión** eficiente.

Como ejemplo, considere los dos grupos, 1 y 2, en la red que se muestra en la Ilustración 16(a). Algunos enrutadores están conectados a hosts que pertenecen a uno o ambos grupos, como se indica en la figura. En la Ilustración 16(b) se muestra un árbol de expansión para el enrutador de la izquierda. Este árbol se puede usar para difusión, pero es exagerado para la multidifusión, como podemos ver de las dos versiones recortadas que se muestran a continuación. En la figura Ilustración 16(c) se han eliminado todos los enlaces que no conducen a hosts que sean miembros del grupo 1. El resultado es el árbol de expansión de multidifusión para el enrutador de la izquierda. Los paquetes se reenvían sólo a través de este árbol de expansión, lo cual es más eficiente que el árbol de difusión debido a que hay 7 enlaces en vez de 10. En la Ilustración 16(d) se muestra el árbol de expansión de multidifusión después de hacer recortes para el grupo 2. También es eficiente, con sólo cinco enlaces esta vez. De igual forma muestra que los diferentes grupos de multidifusión tienen distintos árboles de expansión.

Hay varias maneras de recortar el árbol de expansión. Se puede utilizar la más sencilla si se maneja el enrutamiento por estado del enlace, y cada enrutador está consciente de la topología completa, incluyendo

qué hosts pertenecen a cuáles grupos. Así, cada enrutador puede construir su propio árbol de expansión recortado para cada emisor que envía datos al grupo en cuestión, para lo cual construye un árbol de expansión para el emisor de la manera usual y después elimina todos los enlaces que no conectan a los miembros del grupo con el nodo sumidero. **MOSPF** (OSPF de Multidifusión, del inglés *Multicast OSPF*) es un ejemplo de un protocolo de estado del enlace que funciona de esta manera.

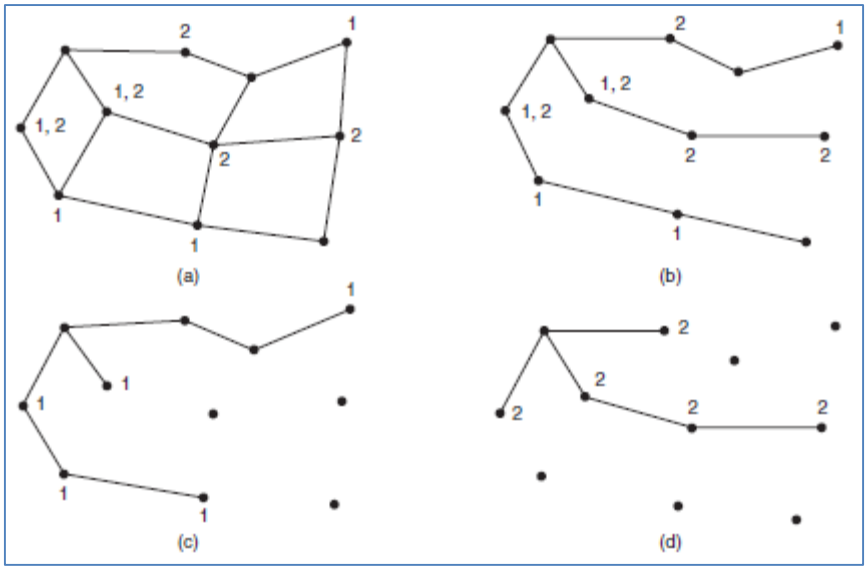


Ilustración 16 - (a) Una red. (b) Un árbol de expansión para el enrutador del extremo izquierdo. (c) Un árbol de multidifusión para el grupo 1. (d) Un árbol de multidifusión para el grupo 2.

Con el enrutamiento por vector de distancia se puede seguir una estrategia de recorte diferente. El algoritmo básico es el reenvío por ruta invertida. Sin embargo, cuando un enrutador sin hosts interesados en un grupo en particular y sin conexiones con otros enrutadores recibe un mensaje de multidifusión para ese grupo, responde con un mensaje *PRUNE* (de recorte) para indicar al vecino emisor que no envíe más multidifusiones para ese grupo. Cuando un enrutador que no tiene miembros del grupo entre sus propios hosts recibe uno de tales mensajes por todas las líneas a las que envía la multidifusión, también puede responder con un mensaje *PRUNE*. De esta forma, el árbol de expansión se recorta recursivamente. **DVMRP** (Protocolo de Enrutamiento Multidifusión de Vector de Distancia, del inglés *Distance Vector Multicast Routing Protocol*) es un ejemplo de un protocolo de enrutamiento multidifusión que funciona de esta manera.

El recorte produce árboles de expansión eficientes que sólo utilizan los enlaces realmente necesarios para llegar a los miembros del grupo. Una desventaja potencial de este algoritmo es que representa mucho trabajo para los enrutadores, en especial en redes grandes. Suponga que una red tiene *n* grupos, cada uno con un promedio de *m* nodos. En cada enrutador y por cada grupo se deben almacenar *m* árboles de expansión recortados, lo que da un total de *mn* árboles. Por ejemplo, la Ilustración 16(c) muestra el árbol de expansión para que el enrutador del extremo izquierdo envíe datos al grupo 1. El árbol de expansión para que el enrutador del extremo derecho envíe datos al grupo 1 (que no se muestra) tendrá una apariencia bastante distinta, ya que los paquetes se dirigirán de manera directa a los miembros del grupo en vez de hacerlo a través del lado izquierdo del grafo. Esto a su vez significa que los enrutadores deben reenviar los paquetes destinados al grupo 1 en distintas direcciones, dependiendo de cuál nodo envíe datos al grupo. Cuando existen muchos grupos grandes con varios emisores, se requiere un almacenamiento considerable para almacenar todos los árboles.

Un diseño alterno utiliza **árboles basados en núcleo** (*core-based trees*) para calcular un solo árbol de expansión por grupo. Todos los enrutadores coinciden en una raíz (el **núcleo** o **punto de reunión**) y para construir el árbol, envían un paquete de cada miembro a la raíz. El árbol es la unión de las rutas trazadas por estos paquetes. La Ilustración 17(a) muestra un árbol basado en núcleo para el grupo 1. Para enviar datos a este grupo, un emisor envía un paquete al núcleo. Cuando el paquete llega al núcleo, se reenvía hacia abajo por el árbol. Esto se muestra en la Ilustración 17(b) para el emisor del lado derecho de la red. Para optimizar el desempeño, los paquetes destinados para el grupo no necesitan llegar al núcleo antes de que se envíen por

multidifusión. Tan pronto como un paquete llega al árbol, se puede reenviar hacia la raíz y por todas las demás ramas. Éste es el caso para el emisor en la parte superior de la Ilustración 17(b).

Tener un árbol compartido no es óptimo para todas las fuentes. Por ejemplo, en la Ilustración 17(b) el paquete del emisor en el extremo derecho llega al miembro del grupo de la parte superior derecha a través del núcleo en tres saltos, en vez de hacerlo de manera directa. La ineficiencia depende de la ubicación del núcleo y los emisores, pero a menudo es razonable cuando el núcleo está en medio de ellos. Cuando sólo hay un emisor, como en un video que se transmite a un grupo por flujo continuo, es óptimo usar el emisor como el núcleo.

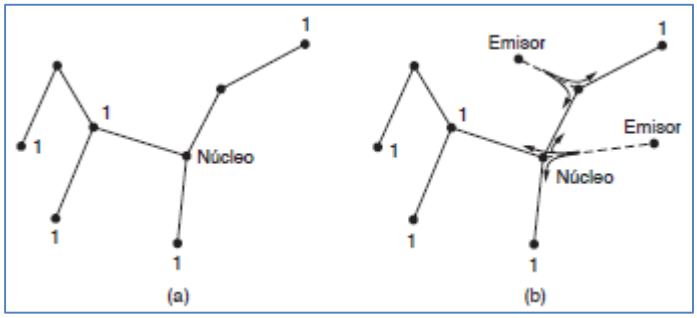


Ilustración 17 - (a) Árbol basado en núcleo para el grupo 1. (b) Envío de datos al grupo 1.

Cabe mencionar también que los árboles compartidos pueden representar un ahorro importante en los costos de almacenamiento, los mensajes enviados y el poder de cómputo. Cada enrutador sólo tiene que mantener un árbol por grupo, en vez de *m* árboles. Además, los enrutadores que no forman parte del árbol no hacen ningún esfuerzo por apoyar el grupo. Por esta razón, los métodos de árbol compartido como los árboles basados en núcleo se utilizan para la multidifusión hacia grupos dispersos en Internet, como parte de protocolos populares como **PIM** (Multidifusión Independiente del Protocolo, del inglés *Protocol Independent Multicast*).

Enrutamiento anycast

Hasta ahora hemos cubierto los modelos de distribución en los que un origen envía a un solo destino (**unidifusión** o *unicast*), a todos los destinos (**difusión** o *broadcast*) y a un grupo de destinos (**multidifusión** o *multicast*). Hay otro modelo de distribución llamado **anycast**, que algunas veces también es útil. En anycast, un paquete se entrega al miembro más cercano de un grupo. Los esquemas que encuentran estas rutas se denominan **enrutamiento anycast**.

¿Por qué querríamos usar anycast? En ocasiones los nodos proporcionan un servicio, como la hora del día o la distribución de contenido, en donde todo lo que importa es obtener la información correcta sin importar el nodo con el que se haya hecho contacto; cualquiera es igual. Por ejemplo, anycast se utiliza en Internet como parte del servicio DNS.

Por suerte no tenemos que idear nuevos esquemas de enrutamiento para anycast, ya que los enrutamientos por vector de distancia y de estado del enlace regulares pueden producir rutas anycast. Suponga que deseamos enviar datos mediante anycast a los miembros del grupo 1. Todos recibirán la dirección “1” en lugar de distintas direcciones. El enrutamiento por vector de distancia distribuirá los vectores en la forma usual y los nodos elegirán la ruta más corta hacia el destino 1. Como resultado, los nodos enviarán datos a la instancia más cercana del destino 1. Las rutas se muestran en la Ilustración 18(a). Este procedimiento funciona debido a que el protocolo de enrutamiento no se da cuenta de que hay varias instancias del destino 1. Es decir, cree que todas las instancias del nodo 1 son el mismo nodo, como en la topología que se muestra en la Ilustración 18(b).

Este procedimiento funciona también para el enrutamiento de estado del enlace, aunque con la consideración adicional de que el protocolo de enrutamiento no debe encontrar rutas que parezcan ser las más cortas y que pasen a través del nodo 1. Esto produciría saltos enormes, ya que las instancias del nodo 1 en realidad son nodos ubicados en distintas partes de la red. Sin embargo, los protocolos de estado del enlace ya pueden hacer esta distinción entre los enrutadores y los hosts. Pasamos por alto este hecho antes debido a que no era necesario para nuestro estudio.

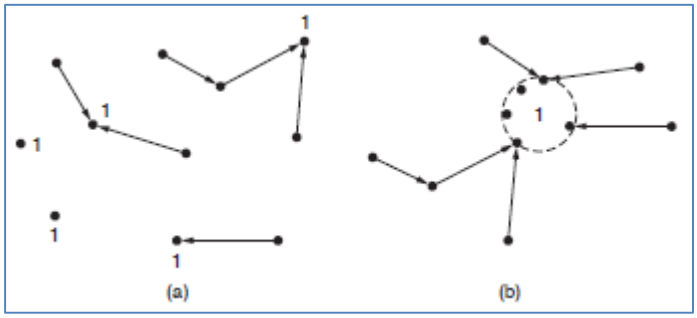


Ilustración 18 - (a) Rutas anycast al grupo 1. (b) Topología que ve el protocolo de enrutamiento.

Enrutamiento para hosts móviles

Millones de personas usan computadoras mientras se trasladan de un lado a otro, desde situaciones verdaderamente móviles con dispositivos inalámbricos en autos en movimiento, hasta situaciones nómadas en las que se utilizan computadoras portátiles en una serie de distintas localidades. Usaremos el término **hosts móviles** para referirnos a cualquiera de las dos categorías antes mencionadas, para diferenciarlas de los hosts fijos que nunca se mueven. Cada vez es más común que las personas quieran estar conectadas en cualquier parte del mundo en donde se encuentren, con la misma facilidad que si estuvieran en su hogar. Estos hosts móviles introducen una nueva complicación: para enrutar un paquete a un host móvil, la red tiene qué encontrarlo primero.

El modelo del mundo que consideraremos es uno en el que se supone que todos los hosts tienen una **ubicación base** (*home location*) permanente que nunca cambia. Además, cada host tiene una **dirección base** (*home address*) permanente que se puede usar para determinar su ubicación base. La meta de enrutamiento en los sistemas con hosts móviles es posibilitar el envío de paquetes a hosts móviles mediante el uso de su dirección base, y hacer que los paquetes lleguen de manera eficiente a ellos en cualquier lugar en el que puedan estar. Lo difícil, por supuesto, es encontrarlos.

Aquí es pertinente mencionar unos detalles sobre este modelo. Un modelo diferente tendría que recalcular rutas mientras el host móvil se desplaza y cambia la topología. Entonces, simplemente usaríamos los esquemas de enrutamiento que describimos antes en esta sección. Sin embargo, con un número creciente de hosts móviles, la red, con este modelo pronto terminaría calculando nuevas rutas sin parar. Al usar las direcciones locales se reduce de manera considerable esta carga.

Otra alternativa sería proporcionar movilidad por encima de la capa de red, algo que ocurre comúnmente con las computadoras portátiles en la actualidad. Cuando se desplazan a nuevas ubicaciones de Internet, las computadoras portátiles adquieren nuevas direcciones de red. No hay asociación entre la dirección antigua y la nueva; la red no sabe que pertenecían a la misma computadora portáltil. En este modelo, podemos usar una computadora portátil para navegar en web, pero otros hosts no le pueden enviar paquetes (por ejemplo, para una llamada entrante) sin construir un servicio de ubicación en la capa superior; por ejemplo, de nuevo iniciar sesión en *Skype* después de moverse. Además, las conexiones no se pueden mantener mientras el host está en movimiento; hay que iniciar nuevas conexiones. La movilidad en la capa de red es útil para corregir estos problemas.

La idea básica que se usa para el enrutamiento móvil en las redes de Internet y celulares es que el host móvil indique su posición actual a un host que se encuentre en la ubicación base. Este host, que actúa en beneficio del host móvil, se denomina **agente de base** (*home agent*). Una vez que conoce la ubicación actual del host móvil, puede reenviar paquetes para que sean entregados.

La Ilustración 19 muestra el enrutamiento móvil en acción. Un emisor al noroeste de la ciudad de Seattle desea enviar un paquete a un host que por lo general se encuentra al otro lado de Estados Unidos, en Nueva York. El caso de interés para nosotros es cuando el host móvil no se encuentra en su ubicación base; por ejemplo, que se encuentre temporalmente en San Diego.

El host móvil en San Diego debe adquirir una dirección de la red local antes de usar esta red. Esto ocurre de la manera usual en la que los hosts obtienen direcciones de red; más adelante veremos cómo se realiza esto en

Internet. La dirección local se denomina **dirección de custodia** (*care of address*). Una vez que el host móvil tiene esta dirección, puede indicar a su agente de base en dónde se encuentra en ese momento. Para ello envía un mensaje de registro al agente de base (paso 1) con la dirección de custodia. El mensaje se muestra con una línea punteada en la Ilustración 19 para indicar que es un mensaje de control y no de datos.

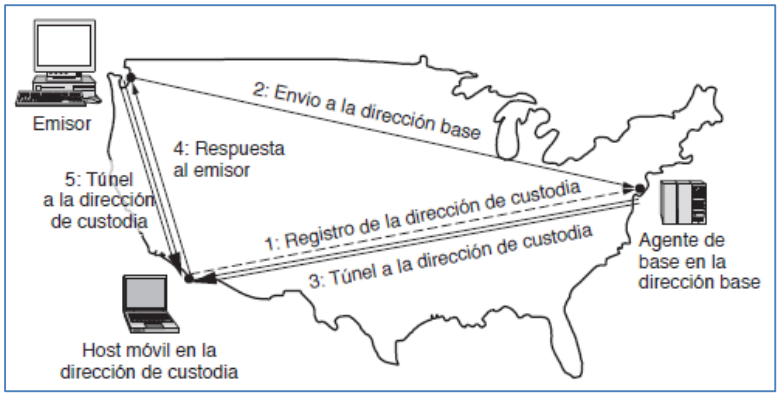


Ilustración 19 - Enrutamiento de paquetes para hosts móviles.

A continuación, el emisor envía un paquete de datos al host móvil mediante su dirección permanente (paso 2). Este paquete es enrutado por la red hacia la ubicación base del host, ya que es a donde pertenece la dirección base. En Nueva York, el agente de base intercepta este paquete, puesto que el host móvil está alejado de su ubicación base. Después envuelve o **encapsula** el paquete con un nuevo encabezado y lo envía a la dirección de custodia (paso 3). Este mecanismo se denomina **tunelización** (*tunneling*).

Cuando el paquete encapsulado llega a la dirección de custodia, el host móvil lo desenvuelve y recupera el paquete del emisor. Después, el host móvil envía su paquete de respuesta directamente al emisor (paso 4). La ruta general se denomina **enrutamiento triangular**, debido a que puede ser la menos directa si la ubicación remota está alejada de la ubicación base. Como parte del paso 4, el emisor puede memorizar la dirección de custodia actual. Los paquetes subsiguientes se pueden enrutar directamente al host móvil si se tunelizan a la dirección de custodia (paso 5), ignorando por completo la ubicación base. Si se pierde la conectividad por alguna razón mientras el móvil se traslada, siempre se puede usar la dirección base para localizarlo.

Un aspecto importante que hemos omitido de esta descripción es la seguridad. En general, cuando un host o enrutador recibe un mensaje de la forma "A *partir de este momento, envíame todo el correo de Estefanía*", podría tener algunas dudas acerca de con quién está hablando y si se trata de una buena idea. La información de seguridad se incluye en los mensajes de manera que se pueda verificar su validez mediante los protocolos criptográficos que veremos en las próximas unidades.

Existen muchas variaciones del enrutamiento móvil. El esquema anterior está modelado con base en la movilidad IPv6, la forma de movilidad que se utiliza en Internet y como parte de las redes celulares basadas en IP como UMTS. Aquí mostramos al emisor como un nodo estático por cuestión de sencillez, pero los diseños permiten que ambos nodos sean hosts móviles. Como alternativa, el host puede formar parte de una red móvil; por ejemplo, una computadora en un avión. Las extensiones del esquema básico soportan redes móviles sin trabajo por parte de los hosts.

Algunos esquemas utilizan un **agente foráneo** (es decir, remoto), algo similar al agente de base pero en la ubicación foránea, o análogo al **VLR** (Registro de Ubicación de Visitante, del inglés *Visitor Location Register*) en las redes celulares. Sin embargo, en esquemas más recientes no se necesita el agente foráneo; los hosts móviles actúan como sus propios agentes foráneos. En cualquier caso, el conocimiento de la ubicación temporal del host móvil se limita a un pequeño número de hosts (por ejemplo, el móvil, el agente de base y los emisores) de modo que muchos de los enrutadores en una red extensa no necesiten recalcular rutas.

Enrutamiento en redes ad hoc

Ya vimos cómo realizar el enrutamiento cuando los hosts son móviles pero los enrutadores son fijos. Un caso aún más extremo es cuando los enrutadores mismos son móviles.

Cada nodo se comunica en forma inalámbrica y actúa como host y como enrutador. A las redes de nodos que simplemente están cerca entre sí se les denomina redes ad hoc o **MANET** (Redes Ad hoc Móviles, del inglés *Mobile Ad hoc NETworks*).

Lo que distingue a las redes ad hoc de las redes cableadas es que en las primeras, de repente se eliminó la topología. Los nodos pueden ir y venir o aparecer en nuevos lugares en cualquier momento. En una red cableada, si un enrutador tiene una ruta válida a algún destino, esa ruta continúa siendo válida a menos que ocurran fallas, que con suerte son raras. En una red ad hoc, la topología podría cambiar todo el tiempo, por lo que la necesidad o la validez de las rutas puede cambiar en cualquier momento, sin previo aviso. No es necesario decir que estas circunstancias hacen del enrutamiento en redes ad hoc algo más desafiante que el enrutamiento en sus contrapartes fijas.

Se ha propuesto una gran variedad de algoritmos de enrutamiento para las redes ad hoc. No obstante, como se han usado muy poco en la práctica, en comparación con las redes móviles, no está claro cuál de estos protocolos es el más útil. Como ejemplo analizaremos uno de los algoritmos de enrutamiento más populares: **AODV** (Vector de Distancia Ad hoc bajo Demanda, del inglés *Ad hoc On-demand Distance Vector*). Es pariente del algoritmo de vector de distancia que se adaptó para funcionar en un entorno móvil, en el que con frecuencia los nodos tienen un ancho de banda y tiempos de batería limitados. Ahora veamos cómo descubre y mantiene las rutas.

Descubrimiento de ruta

En el AODV, las rutas a un destino se descubren bajo demanda; es decir, sólo cuando alguien desea enviar un paquete a ese destino. Esto ahorra mucho trabajo que se desperdiciaría de otra manera, cuando la topología cambia antes de usar la ruta. En cualquier instante podemos describir la topología de una red ad hoc mediante un grafo de nodos conectados. Dos nodos se conectan (es decir, tienen un arco entre ellos en el grafo) si se pueden comunicar de manera directa mediante sus radios. Un modelo básico pero adecuado, que basta para nuestros fines, es que cada nodo se puede comunicar con los demás nodos que se encuentren dentro de su círculo de cobertura. Las redes reales son más complicadas, con edificios, colinas y demás obstáculos que bloquean la comunicación, y nodos para los que *A* está conectado a *B*, pero *B* no está conectado a *A* debido a que *A* tiene un transmisor de mayor potencia que *B*. Sin embargo, por simplicidad asumiremos que todas las conexiones son simétricas.

Para describir el algoritmo, considere la red ad hoc recién formada de la Ilustración 20. Suponga que un proceso en el nodo *A* desea enviar un paquete al nodo *I*. El algoritmo AODV mantiene una tabla de vectores de distancia en cada nodo, codificada por destino, que proporciona información acerca de ese destino, incluyendo a cuál vecino hay que enviar los paquetes con el fin de llegar al destino. Primero, *A* busca en su tabla y no encuentra una entrada para *I*. Ahora tiene que descubrir una ruta a *I*. Debido a esta propiedad de descubrir rutas sólo cuando es necesario, este algoritmo se conoce como "*bajo demanda*".

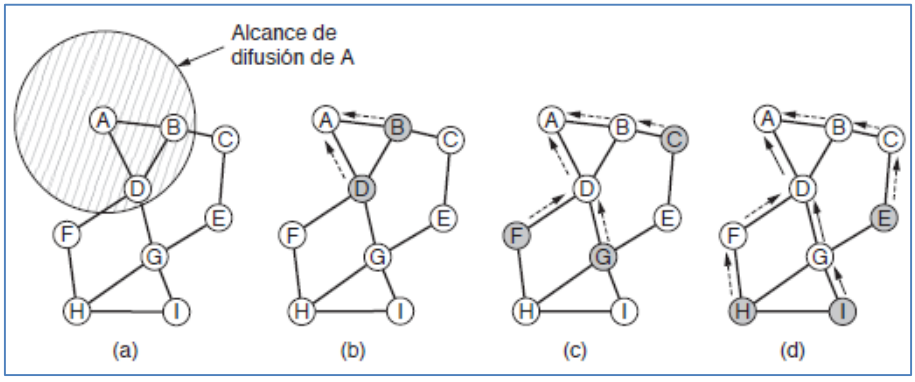


Ilustración 20 - a) Alcance de difusión de A. (b) Después de que B y D reciben la difusión de A. (c) Después de que C, F y G reciben la difusión de A. (d) Después de que E, H e I reciben la difusión de A. Los nodos sombreados son nuevos receptores. Las líneas punteadas muestran las posibles rutas invertidas. Las líneas sólidas muestran la ruta descubierta.

Para localizar a *I*, *A* construye un paquete especial de solicitud de ruta (*ROUTE REQUEST*) y lo difunde usando inundación. La transmisión de *A* llega a *B* y *D*, como se ilustra en la Ilustración 20(a). Cada nodo vuelve a

difundir la solicitud, que continúa para llegar a los nodos *F*, *G* y *C* en la Ilustración 20(c), y a los nodos *H*, *E* e *I* en la Ilustración 20(d). Se utiliza un número de secuencia establecido en la fuente para eliminar los duplicados durante la inundación. Por ejemplo, *D* descarta la transmisión proveniente de *B* en la Ilustración 20(c) debido a que ya reenvió la solicitud.

En un momento dado la solicitud llega al nodo *I*, que construye un paquete de respuesta de ruta (*ROUTE REPLY*). Este paquete se envía mediante unidifusión al emisor junto con la ruta invertida, seguida de la solicitud. Para que esto funcione, cada nodo intermedio debe recordar al nodo que le envió la solicitud. Las flechas en la Ilustración 20(b)-(d) muestran la información almacenada de la ruta invertida. Cada nodo intermedio también incrementa una cuenta de saltos al reenviar la respuesta. Esto indica a los nodos qué tan alejados están del destino. Las respuestas indican a cada nodo intermedio qué vecino debe usar para llegar al destino: es el nodo que les envió la respuesta. Los nodos intermedios *G* y *D* colocan la mejor ruta que hayan escuchado en sus tablas de enrutamiento, mientras procesan la respuesta. Cuando la respuesta llega a *A*, se ha creado una nueva ruta, *ADGI*.

En una red extensa, el algoritmo genera muchas difusiones, incluso para destinos cercanos. Para reducir la sobrecarga, el alcance de las difusiones se limita mediante el campo *Tiempo de vida* del paquete IP. Este campo es inicializado por el emisor y se decrementa en cada salto. Si llega a 0, el paquete se descarta en vez de difundirlo. Después, el proceso de descubrimiento de ruta se modifica de la siguiente manera. Para localizar un destino, el emisor difunde un paquete de solicitud de ruta (*ROUTE REQUEST*) con el campo *Tiempo de vida* establecido en 1. Si no regresa una respuesta dentro de un periodo razonable, se envía otro paquete, esta vez con el campo *Tiempo de vida* establecido en 2. Los intentos subsiguientes usan 3, 4, 5, etc. De esta forma la búsqueda se intenta primero de forma local, y después en anillos cada vez más amplios.

Mantenimiento de rutas

Debido a que es posible mover o apagar los nodos, la topología puede cambiar de manera espontánea. Por ejemplo, en la Ilustración 20, si *G* se apaga, *A* no se dará cuenta de que la ruta a *I* (*ADGI*) que estaba utilizando ya no es válida. El algoritmo necesita ser capaz de manejar esto. Cada nodo difunde de manera periódica un mensaje de saludo (*Hello*). Se espera que cada uno de sus vecinos responda a dicho mensaje. Si no se recibe ninguna respuesta, el difusor sabe que el vecino se ha movido del alcance o falló y ya no está conectado a él. De manera similar, si el difusor trata de enviar un paquete a un vecino que no responde, se da cuenta de que el vecino ya no está disponible.

Esta información se utiliza para eliminar rutas que ya no funcionan. Para cada destino posible, cada nodo, *N*, mantiene un registro de sus vecinos activos que le han proporcionado un paquete para ese destino durante los últimos ΔT segundos. Cuando alguno de los vecinos de *N* es inalcanzable, verifica su tabla de enrutamiento para ver cuáles destinos tienen rutas que utilicen el vecino que ya no se puede localizar. Para cada una de esas rutas, se informa a los vecinos activos que su ruta a través de *N* es ahora inválida y debe eliminarse de sus tablas de enrutamiento. En nuestro ejemplo, *D* elimina sus entradas para *G* e *I* de su tabla de enrutamiento y notifica a *A*, quien elimina su entrada para *I*. En el caso general, los vecinos activos avisan a sus vecinos activos y así en lo sucesivo, en forma recursiva hasta que se eliminan de todas las tablas de enrutamiento todas las rutas que dependen del nodo ahora ilocalizable.

En esta etapa, ya se eliminaron las rutas inválidas de la red y los emisores pueden encontrar nuevas rutas válidas mediante el mecanismo de descubrimiento que describimos. Sin embargo, existe una complicación. Recordemos que los protocolos de vector de distancia pueden sufrir de una convergencia lenta o de problemas de conteo al infinito después de un cambio en la topología, en donde pueden confundir las rutas antiguas e inválidas con las rutas nuevas y válidas.

Para asegurar una convergencia rápida, las rutas incluyen un número de secuencia controlado por el destino. El número de secuencia de destino es como un reloj lógico. El destino lo incrementa cada vez que envía un nuevo paquete de respuesta de ruta (*ROUTE REPLY*). Para preguntar por una ruta nueva, los emisores incluyen en el paquete *ROUTE REQUEST* el número de secuencia de destino de la última ruta que usaron, que puede ser el número de secuencia de la ruta que se acaba de eliminar, o 0 como un valor inicial. La petición se difundirá hasta encontrar una ruta con un número de secuencia más alto. Los nodos intermedios almacenan

las rutas que tienen un número de secuencia más alto, o la menor cantidad de saltos para el número de secuencia actual.

En aras de un protocolo bajo demanda, los nodos intermedios sólo almacenan las rutas en uso. La demás información sobre rutas aprendida durante las difusiones expira después de un retardo corto. Al descubrir y almacenar sólo las rutas que se utilizan es posible ahorrar ancho de banda y vida de la batería, en comparación con un protocolo de vector de distancia estándar, que actualiza las difusiones en forma periódica.

Hasta ahora sólo hemos considerado una sola ruta, de *A* a *I*. Para ahorrar más recursos, se comparte el descubrimiento de rutas y el mantenimiento cuando las rutas se traslapan. Por ejemplo, si *B* también desea enviar paquetes a *I*, llevará a cabo el descubrimiento de rutas. Sin embargo, en este caso la solicitud llegará primero a *D*, que ya tiene una ruta a *I*. El nodo *D* puede entonces generar una respuesta para indicar a *B* la ruta sin requerir ningún trabajo adicional.

Existen muchos más esquemas de enrutamiento ad hoc. Otro esquema bajo demanda muy conocido es **DSR** (Enrutamiento Dinámico de Origen, del inglés *Dynamic Source Routing*). El esquema **GPSR** (Enrutamiento sin Estado con Perímetro Codicioso, del inglés *Greedy Perimeter Stateless Routing*) explora una estrategia diferente, basada en la geografía. Si todos los nodos conocen sus posiciones geográficas, el reenvío a un destino puede proceder sin necesidad de calcular las rutas, con sólo dirigirse en el sentido correcto y regresar en círculos para escapar de cualquier camino sin salida. Los protocolos que sobrevivan dependerán de los tipos de redes ad hoc que demuestren ser útiles en la práctica.

Algoritmos de control de congestión

Cuando hay demasiados paquetes presentes en una red (o en una parte de ella), hay retardo o pérdida en los paquetes y se degrada el desempeño. Esta situación se llama **congestión**. Las capas de red y de transporte comparten la responsabilidad de manejar la congestión. Como ésta ocurre dentro de la red, la capa de red es quien la experimenta en forma directa y en última instancia debe determinar qué hacer con los paquetes sobrantes. Sin embargo, la manera más efectiva de controlar la congestión es reducir la carga que la capa de transporte coloca en la red. Para ello se requiere que las capas de red y de transporte trabajen en conjunto. En esta sección analizaremos los aspectos de la congestión relacionados con la red.

En la Ilustración 21 se muestra el comienzo de la congestión. Cuando la cantidad de paquetes que el host envía a la red está muy por debajo de su capacidad de transporte, la cantidad entregada es proporcional a la cantidad enviada. Si se envía el doble de paquetes, se entrega el doble de ellos. Sin embargo, a medida que la carga ofrecida se acerca a la capacidad de transporte, las ráfagas de tráfico llenan ocasionalmente los búferes dentro de los enrutadores y se pierden algunos paquetes. Estos paquetes perdidos consumen una parte de la capacidad, por lo que la cantidad de paquetes entregados cae por debajo de la curva ideal. Ahora la red está congestionada.

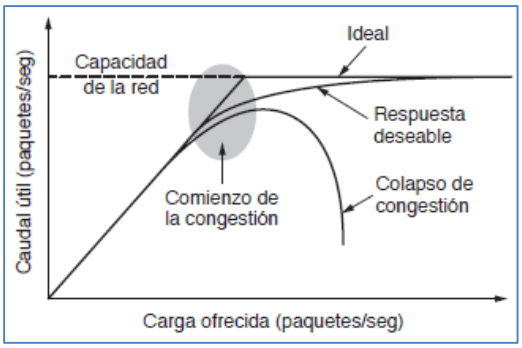


Ilustración 21 - Carga ofrecida (paquetes/s).

A menos que la red esté bien diseñada, puede experimentar un **colapso por congestión**, en donde el desempeño se desploma a medida que aumenta la carga ofrecida más allá de la capacidad. Esto puede ocurrir debido a que los paquetes se pueden retrasar el tiempo suficiente dentro de la red como para que ya no sean útiles cuando salgan de ella. Por ejemplo, en los primeros días de Internet, el tiempo que invertía un paquete en esperar a que se enviaran primero los paquetes acumulados a través de un enlace de 56 kbps, podía llegar

al máximo permitido para permanecer en la red. Entonces había que descartarlo. Un modo de falla distinto ocurre cuando los emisores retransmiten los paquetes que están muy atrasados, creyendo que se perdieron. En este caso la red entregará copias del mismo paquete, desperdiciando de nuevo su capacidad. Para capturar estos factores, al eje *y* de la Ilustración 21 se le nombró como **caudal útil** (*goodput*), que es la tasa a la que se entregan los paquetes útiles en la red.

Nos gustaría diseñar redes que evitaran la congestión siempre que fuera posible y que no sufrieran de un colapso por congestión en caso de que se atascaran. Por desgracia, no podemos evitar la congestión por completo. Si de repente empiezan a llegar flujos de paquetes en tres o cuatro líneas de entrada y todos necesitan la misma línea de salida, se acumulará una cola. Si no hay suficiente memoria para almacenarlos a todos, se perderán paquetes. Tal vez agregar memoria ayude hasta cierto punto, pero Nagle descubrió que si los enrutadores tienen una cantidad infinita de memoria, la congestión empeora en vez de mejorar. Esto se debe a que, para cuando los paquetes llegan a la parte frontal de la cola, ya expiraron (repetidas veces) y se enviaron duplicados. Esto empeora las cosas, no las mejora: conduce al colapso por congestión.

Los enlaces o enrutadores con poco ancho de banda, que procesan paquetes con más lentitud que la tasa de transmisión de la línea, también se pueden congestionar. En este caso, se puede mejorar la situación al dirigir parte del tráfico de manera que se aleje del embotellamiento y se vaya a otras partes de la red. Sin embargo, en un momento dado todas las regiones de la red estarán congestionadas. En esta situación, no hay otra alternativa más que deshacerse de una parte de la carga o construir una red más rápida.

Vale la pena recalcar la diferencia entre el *control de la congestión* y el *control de flujo*, pues la relación es muy sutil. El control de congestión se ocupa de asegurar que la red sea capaz de transportar el tráfico ofrecido. Es un asunto global, en el que interviene el comportamiento de todos los hosts y enrutadores. En contraste, el control de flujo se relaciona con el tráfico entre un emisor particular y un receptor particular. Su tarea es asegurar que un emisor rápido no pueda transmitir datos de manera continua a una velocidad mayor que la que puede absorber los paquetes el receptor.

Para ver la diferencia entre estos dos conceptos, considere una red compuesta de enlaces de fibra óptica de 100 Gbps en la que una supercomputadora está tratando de transferir por la fuerza un archivo extenso a una computadora personal que sólo puede manejar 1 Gbps. Aunque no hay congestión (la red misma no es el problema), se requiere control de flujo para obligar a la supercomputadora a detenerse con frecuencia para darle a la computadora personal un momento de respiro.

En el otro extremo, considere una red con líneas de 1 Mbps y 1000 computadoras grandes, la mitad de las cuales trata de transferir archivos a 100 kbps a la otra mitad. Aquí el problema no es que los emisores rápidos saturen a los receptores lentos, sino que el tráfico ofrecido total excede lo que la red puede manejar.

La razón por la que se confunden con frecuencia el control de congestión y el control de flujo es que la mejor manera de lidiar con ambos problemas es hacer que el host reduzca su velocidad. Así, un host puede recibir un mensaje de "*reducción de velocidad*" porque el receptor no puede manejar la carga o porque la red no la puede manejar.

Comenzaremos nuestro estudio del control de congestión mediante un análisis de los métodos que se pueden usar en distintas escalas de tiempo. Después analizaremos los métodos para prevenir que ocurra la congestión en primer lugar, seguidos de los métodos para lidiar con la congestión una vez que se ha establecido.

Métodos para el control de la congestión

La presencia de congestión significa que la carga es (temporalmente) mayor de la que los recursos (en una parte de la red) pueden manejar. Dos soluciones vienen a la mente: aumentar los recursos o reducir la carga. Como se muestra en la Ilustración 22, estas soluciones por lo general se aplican en distintas escalas de tiempo, para prevenir la congestión o reaccionar ante ella una vez que se presenta.

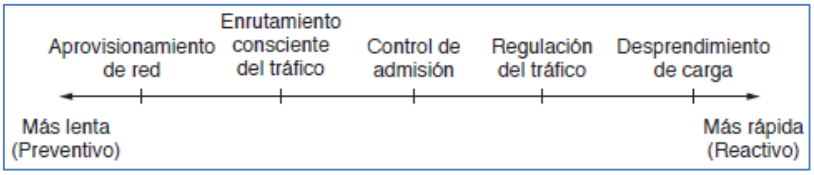


Ilustración 22 - Escalas de tiempo de los métodos para el control de la congestión.

La manera más básica de evitar la congestión es construir una red que coincida bien con el tráfico que transmita. Si hay un enlace con poco ancho de banda en la ruta a través de la cual se dirige la mayor parte del tráfico, es probable que haya congestión. Algunas veces se pueden agregar recursos en forma dinámica cuando hay un problema grave de congestión; por ejemplo, activar enrutadores de repuesto o habilitar líneas que por lo general se usan sólo como respaldos (para que el sistema sea tolerante a fallas), o comprar ancho de banda en el mercado abierto. Lo más frecuente es que los enlaces y enrutadores que se utilizan mucho en forma regular se actualicen a la primera oportunidad. A esto se le conoce como aprovisionamiento y ocurre en una escala de tiempo de meses, con base en las tendencias de tráfico de largo plazo.

Para aprovechar al máximo la capacidad existente de la red, las rutas se pueden ajustar a los patrones de tráfico que cambian durante el día, a medida que los usuarios de la red entran y salen en distintas zonas horarias. Por ejemplo, se pueden cambiar las rutas para desviar el tráfico de las mismas que se utilizan mucho si se cambian las ponderaciones de las rutas más cortas. Ciertas estaciones de radio locales tienen helicópteros que vuelan por sus ciudades para informar sobre la congestión en las carreteras, de modo que sus radioescuchas móviles puedan enrutar sus paquetes (autos) alrededor de los sitios con mucho tráfico. A esto se le conoce como **enrutamiento consciente del tráfico** (*traffic-aware routing*). También es útil dividir el tráfico entre varias rutas.

Sin embargo, en ocasiones no es posible aumentar la capacidad. La única forma entonces para combatir la congestión es reducir la carga. En una red de circuitos virtuales, se podrían rechazar las nuevas conexiones si provocaran el congestionamiento de la red. A esto se le conoce como **control de admisión**.

A un nivel más detallado, cuando la congestión es inminente la red puede distribuir retroalimentación a las fuentes cuyos flujos de tráfico sean responsables del problema. La red puede solicitar que estas fuentes regulen su tráfico, o puede reducir el tráfico por sí misma.

Dos dificultades con este método son: cómo identificar el comienzo de la congestión y cómo informar a la fuente que necesita reducir su velocidad. Para lidiar con la primera cuestión, los enrutadores pueden monitorear la carga promedio, el retardo de encolamiento o la pérdida de paquetes. En todos los casos, los números crecientes indican un aumento en la congestión.

Para lidiar con la segunda cuestión, los enrutadores deben participar en un lazo de retroalimentación con las fuentes. Para que un esquema funcione de manera correcta, es necesario ajustar la escala de tiempo con cuidado. Si un enrutador grita *ALTO* cada vez que llegan dos paquetes seguidos y grita *SIGA* cada vez que está inactivo por 20 µs, el sistema oscilará sin control y nunca convergerá. Por otra parte, si espera 30 minutos para asegurarse antes de decir algo, el mecanismo de control de congestión reaccionará tan despacio que no será de utilidad. Encontrar una retroalimentación oportuna no es un asunto trivial. Una cuestión adicional es tener enrutadores que envíen más mensajes cuando la red ya se encuentra congestionada.

Por último, cuando todo lo demás falla, la red se ve obligada a descartar los paquetes que no puede entregar. El nombre general para esto es **desprendimiento de carga** (*load shedding*). Una buena política para elegir qué paquetes descartar puede ayudar a evitar un colapso por congestión.

Enrutamiento consciente del tráfico

Los esquemas de enrutamiento que vimos anteriormente utilizaban ponderaciones de enlaces fijas. Estos esquemas se adaptaban a los cambios en la topología, pero no a los cambios en la carga. El objetivo al tener en cuenta la carga al calcular las rutas es desviar el tráfico de los puntos más activos que serán los primeros lugares en la red en experimentar congestión.

La manera más directa de hacer esto es establecer la ponderación de enlaces de manera que sea una función del ancho de banda del enlace (fijo) y el retardo de propagación más la carga medida (variable) o el retardo de encolamiento promedio. Así, las rutas de menor ponderación favorecerán a las rutas que tengan cargas más ligeras, siendo todo lo demás igual.

El enrutamiento consciente del tráfico se utilizó en los primeros días de Internet de acuerdo con este modelo. Sin embargo, existe un riesgo. Considere la red de la Ilustración 23, que está dividida en dos partes, Este y Oeste, conectada mediante dos enlaces, *CF* y *EI*. Suponga que la mayor parte del tráfico entre Este y Oeste utiliza el enlace *CF* y, como resultado, este enlace está muy cargado con retardos muy grandes. Si se incluye el retardo de encolamiento en la ponderación utilizada para el cálculo de la ruta más corta, el enlace *EI* será más atractivo. Una vez que se hayan instalado las nuevas tablas de enrutamiento, la mayor parte del tráfico Este-Oeste viajará ahora a través de *EI*, y se cargará este enlace. Como consecuencia, en la siguiente actualización *CF* parecerá ser la ruta más corta. De esta forma, las tablas de enrutamiento pueden oscilar mucho, lo cual conducirá a un enrutamiento errático y muchos problemas potenciales.

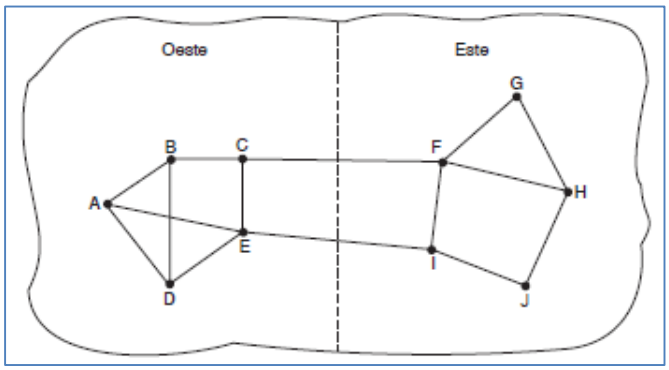


Ilustración 23 - Una red en la cual las partes Este y Oeste se conectan mediante dos enlaces.

Si se ignora la carga y sólo se consideran el ancho de banda y el retardo de propagación, este problema no ocurre. Los intentos de incluir la carga pero cambiar las ponderaciones dentro de un rango estrecho sólo reducen las oscilaciones de enrutamiento. Hay dos técnicas que pueden contribuir a una solución exitosa. La primera es el enrutamiento *multitrayectoria*, en donde puede haber múltiples rutas de un origen a un destino. En nuestro ejemplo, esto significa que el tráfico se puede esparcir a través de ambos enlaces de Este a Oeste. La segunda es que el esquema de enrutamiento desvíe el tráfico a través de las rutas con la suficiente lentitud como para que pueda converger.

Dadas estas dificultades, en Internet los protocolos de enrutamiento por lo general no ajustan sus rutas dependiendo de la carga, sino que los ajustes se realizan fuera del protocolo de enrutamiento, al cambiar lentamente sus entradas. A esto se le denomina **ingeniería de tráfico**.

Control de admisión

Una técnica que se utiliza mucho en las redes de circuitos virtuales para controlar la congestión es el **control de admisión**. La idea es simple: no se debe establecer un nuevo circuito virtual a menos que la red pueda transportar el tráfico adicional sin congestionarse. Por lo tanto, pueden fallar los intentos por establecer un circuito virtual. Esto es mejor que la alternativa, ya que dejar entrar más personas cuando la red está ocupada sólo empeora las cosas. Por analogía, en el sistema telefónico, cuando un conmutador se sobrecarga, practica el control de admisión al no dar tonos de marcado.

El truco con este método es averiguar cuándo puede un nuevo circuito virtual provocar una congestión. La tarea es simple en el sistema telefónico debido al ancho de banda fijo de las llamadas (64 kbps para audio no comprimido). Sin embargo, los circuitos virtuales en las redes de computadoras vienen en todas las formas y tamaños. Por ende, el circuito debe incluir alguna caracterización de su tráfico si se va a aplicar el control de admisión.

A menudo el tráfico se describe en términos de su tasa de transmisión y forma. El problema de cómo describirlo en una forma simple pero significativa es difícil, ya que por lo general el tráfico es de ráfagas; la

tasa promedio es sólo la mitad de la historia. Por ejemplo, el tráfico que varía mientras se navega por la web es más difícil de manejar que una película de flujo continuo con la misma velocidad real de transporte a largo plazo, pues es más probable que las ráfagas del tráfico web congestionen los enrutadores en la red. Un descriptor de uso común que captura este efecto es la **cubeta con goteo** (*leaky bucket*) o **cubeta con token** (*token bucket*). Una cubeta con goteo tiene dos parámetros que vinculan la tasa promedio y el tamaño de la ráfaga instantánea de tráfico.

Armada con las descripciones del tráfico, la red puede decidir si admite o no el nuevo circuito virtual. Una posibilidad es que la red reserve suficiente capacidad a lo largo de las rutas de cada uno de sus circuitos virtuales, de modo que no ocurra una congestión. En este caso, la descripción del tráfico es un acuerdo de servicio en cuanto a lo que la red garantizará a sus usuarios. Hemos prevenido la congestión, pero viramos hacia el tema relacionado de la calidad del servicio un poco antes de tiempo; regresaremos a este tema en la siguiente sección.

Aun sin hacer garantías, la red puede usar las descripciones del tráfico para el control de admisión. De esta forma, la tarea es estimar cuántos circuitos caben dentro de la capacidad de transporte de la red sin que haya congestión. Suponga que los circuitos virtuales que pueden enviar tráfico a tasas de hasta 10 Mbps pasan por el mismo enlace físico de 100 Mbps. ¿Cuántos circuitos se deben admitir? Sin duda, se pueden admitir 10 sin arriesgarse a una congestión, pero esto es un desperdicio en el caso normal puesto que sería muy raro que los 10 circuitos transmitieran a toda velocidad al mismo tiempo. En las redes reales, las mediciones del comportamiento en el pasado que capturan las estadísticas de las transmisiones, se pueden usar para estimar el número de circuitos que se pueden admitir, para intercambiar un mejor desempeño por un riesgo aceptable.

El control de admisión también se puede combinar con el enrutamiento consciente del tráfico si se consideran las rutas alrededor de los puntos con mayor tráfico como parte del procedimiento de establecimiento. Por ejemplo, considere la red que se muestra en la Ilustración 24(a), en donde hay dos enrutadores congestionados según se indica.

Suponga que un host conectado al enrutador *A* desea establecer una conexión a un host conectado al enrutador *B*. Por lo general, esta conexión pasaría a través de uno de los enrutadores congestionados. Para evitar esta situación, podemos redibujar la red como se muestra en la Ilustración 24(b), en donde se omiten los enrutadores congestionados y todas sus líneas. La línea punteada muestra una posible ruta para el circuito virtual que evita a los enrutadores congestionados.

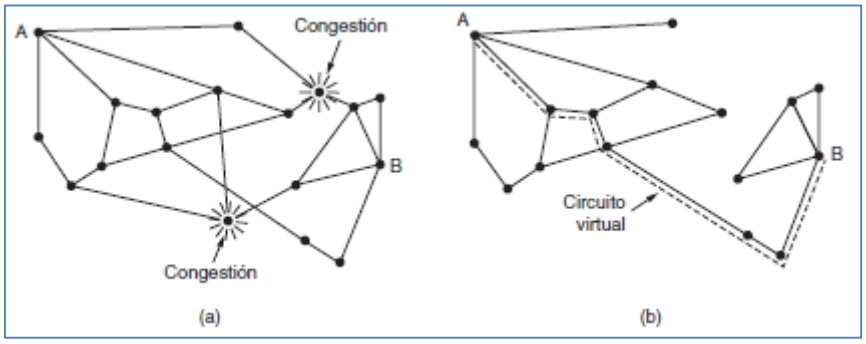


Ilustración 24 - (a) Una red congestionada. (b) La parte de la red que no está congestionada. También se muestra un circuito virtual de A a B.

Regulación de tráfico

En Internet y en muchas otras redes de computadoras, los emisores ajustan sus transmisiones para enviar tanto tráfico como la red pueda distribuir. En este escenario, la red aspira a operar justo antes de que comience la congestión. Cuando la congestión es inminente, debe pedir a los emisores que reduzcan sus transmisiones y su velocidad. Esta retroalimentación es algo común, en vez de una situación excepcional. En ocasiones se utiliza el término **evasión de congestión** para contrastar este punto de operación con el punto en que la red se ha congestionado (demasiado).

Ahora veamos algunos métodos para regular el tráfico que se pueden usar en las redes de datagramas y en las de circuitos virtuales. Cada método debe resolver dos problemas. En primer lugar, **los enrutadores deben determinar cuándo se acerca la congestión**, siendo lo ideal antes de que haya llegado. Para ello, cada enrutador puede monitorear en forma continua los recursos que utiliza. Tres posibilidades son: usar los enlaces de salida, el búfer para los paquetes puestos en cola dentro del enrutador y el número de paquetes que se pierden debido a una capacidad insuficiente. De estas posibilidades, la segunda es la más útil. Los promedios de uso no justifican directamente las ráfagas de la mayoría del tráfico; un uso del 50% puede ser bajo para un tráfico uniforme y demasiado alto para un tráfico muy variable. Las cuentas de los paquetes perdidos llegan demasiado tarde. La congestión ya ha comenzado para cuando se empiezan a perder los paquetes.

El retardo de encolamiento dentro de los enrutadores captura de manera directa cualquier congestión experimentada por los paquetes. Debe ser bajo la mayor parte del tiempo, pero se disparará cuando haya una ráfaga de tráfico que genere una acumulación de paquetes. Para mantener una buena estimación del retardo de encolamiento, *d*, se puede realizar un muestreo periódico de la longitud de cola instantánea, *s*, y se puede actualizar *d* de acuerdo con

$$d_{nueva} = \alpha d_{anterior} + (1 - \alpha)s$$

en donde la constante α determina qué tan rápido olvida el enrutador el historial reciente. A esto se le llama **EWMA** (Promedio Móvil Ponderado Exponencialmente, del inglés *Exponentially Weighted Moving Average*). Este promedio corrige las fluctuaciones y es equivalente a un filtro pasabajos. Cada vez que *d* se mueve por encima del umbral, el enrutador detecta el comienzo de la congestión.

El segundo problema es que **los enrutadores deben entregar una retroalimentación oportuna a los emisores que provocan la congestión**. Ésta se experimenta en la red, pero para aliviarla se requiere una acción de parte de los emisores que están usando la red. Para entregar la retroalimentación, el enrutador debe identificar a los emisores apropiados. Después, debe advertirles con cuidado, sin enviar más paquetes a la red que ya está congestionada. Los distintos esquemas usan mecanismos de retroalimentación diferentes, como veremos a continuación.

Paquetes reguladores

La manera más directa de notificar a un emisor sobre la congestión es decírselo directamente. En este método, el enrutador selecciona un paquete congestionado y envía un **paquete regulador** de vuelta al host de origen, proporcionándole el destino encontrado en el paquete. El paquete original se puede etiquetar (se activa un bit de encabezado) de modo que no genere más paquetes reguladores más adelante en la ruta y después se reenvía de la manera usual. Para evitar aumentar la carga en la red durante un momento de congestión, el enrutador tal vez sólo envíe paquetes reguladores a una tasa de transmisión baja.

Cuando el host de origen obtiene el paquete regulador, se le pide que reduzca el tráfico enviado al destino especificado; por ejemplo, un 50%. En una red de datagramas, con sólo elegir paquetes al azar cuando hay congestión es probable que los paquetes reguladores se envíen a los emisores rápidos, ya que tendrán la mayor parte de los paquetes en la cola. La retroalimentación implícita en este protocolo puede ayudar a evitar la congestión sin necesidad de regular a ninguno de los emisores, a menos que ocasione problemas. Por la misma razón, es probable que se envíen varios paquetes reguladores a un host y destino específicos. El host debe ignorar estos paquetes reguladores adicionales durante el intervalo fijo, hasta que tenga efecto su reducción del tráfico. Después de ese periodo, los demás paquetes reguladores indican que la red sigue estando congestionada.

Notificación explícita de congestión

En vez de generar paquetes adicionales para advertir sobre la congestión, un enrutador puede etiquetar cualquier paquete que reenvíe (para lo cual establece un bit en el encabezado de éste) para indicar que está experimentando una congestión. Cuando la red entrega el paquete, el destino puede observar que hay congestión e informa al emisor sobre ello cuando envíe un paquete de respuesta. En consecuencia, el emisor puede regular sus transmisiones como antes.

A este diseño se le conoce como **ECN** (Notificación Explícita de Congestión, del inglés *Explicit Congestion Notification*) y se utiliza en Internet. Es un refinamiento de los primeros protocolos de señalización de congestión, de los cuales el más notable fue el esquema de retroalimentación binaria de Ramakrishnan y Jain que se utilizó en la arquitectura de DECNet. Se utilizan dos bits en el encabezado del paquete IP para registrar el momento en que el paquete experimenta una congestión. Los paquetes no están marcados cuando se envían, como se observa en la Ilustración 25. Si alguno de los enrutadores por los que pasen está congestionado, entonces ese enrutador marcará el paquete para indicar que experimentó una congestión mientras lo reenvía. Después, el destino repetirá cualquier marca de vuelta al emisor, como una señal de congestión explícita en su siguiente paquete de respuesta. Esto se muestra con una línea punteada en la figura para indicar que ocurre por encima del nivel de IP (por ejemplo, en TCP). El emisor debe entonces regular sus transmisiones, como en el caso de los paquetes reguladores.

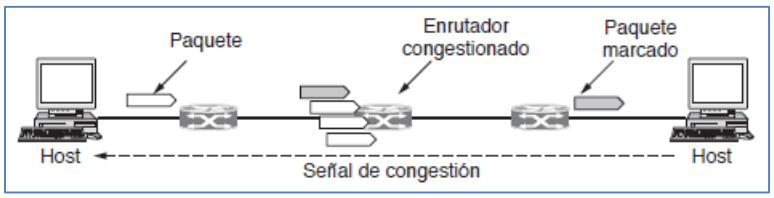


Ilustración 25 - Notificación explícita de congestión.

Contrapresión de salto por salto

A largas distancias o altas velocidades, muchos paquetes nuevos se pueden transmitir una vez que se haya señalado la congestión debido al retardo antes de que la señal haga efecto.

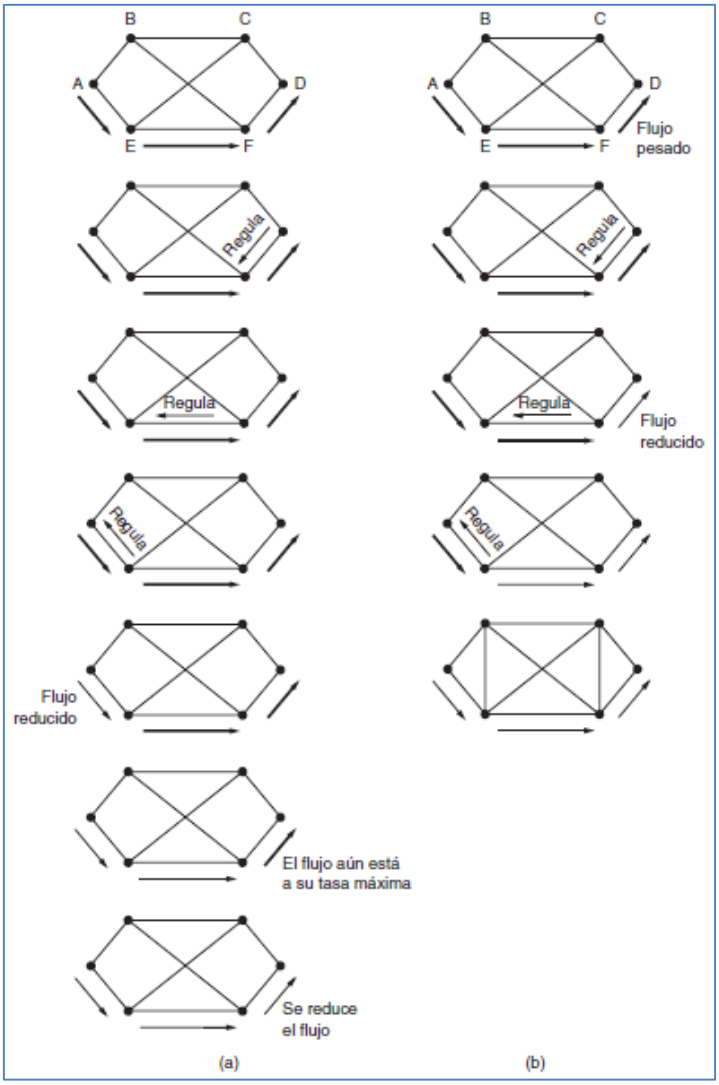


Ilustración 26 - (a) Un paquete regulador que afecta sólo al origen. (b) Un paquete regulador que afecta a cada salto por el que pasa.

Un método alternativo es hacer que el paquete regulador ejerza su efecto en cada salto por el que pase, como se muestra en la secuencia de la Ilustración 26(b). Aquí, tan pronto como el paquete regulador llega a *F*, se obliga a *F* a reducir el flujo que va hacia *D*. Para hacer esto, *F* tendría que destinar más búferes a la conexión, ya que la fuente aún está transmitiendo a toda velocidad, pero da a *D* un alivio inmediato. En el siguiente paso, el paquete regulador llega a *E*, e indica a éste que reduzca el flujo a *F*. Esta acción impone una mayor demanda sobre los búferes de *E* pero da un alivio inmediato a *F*. Por último, el paquete regulador llega a *A* y el flujo se reduce.

El efecto neto de este esquema de salto por salto es proporcionar un alivio rápido en el punto de congestión, a expensas de usar más búferes ascendentes. De esta manera se puede cortar la congestión de raíz sin que se pierdan paquetes.

Desprendimiento de carga

Cuando ninguno de los métodos anteriores elimina la congestión, los enrutadores pueden pasar al método más extremo: el **desprendimiento de carga**, que es una manera rebuscada de decir que cuando se inunda a los enrutadores con paquetes que no pueden manejar, simplemente se tiran. El término viene del mundo de la generación de energía eléctrica, donde se refiere a la práctica de instalaciones que producen apagones intencionales en ciertas áreas para salvar a la red completa de colapsarse en días calurosos de verano, en los que la demanda de energía eléctrica excede por mucho el suministro.

La pregunta clave para un enrutador abrumado por paquetes es cuáles paquetes tirar. La opción preferida puede depender del tipo de aplicaciones que utiliza la red. En una transferencia de archivos vale más un

paquete viejo que uno nuevo, puesto que si el enrutador se deshace del paquete 6 y mantiene los paquetes 7 a 10, por ejemplo, sólo obligará al receptor a esforzarse más por colocar en el búfer los datos que no puede usar todavía. En contraste, para los medios en tiempo real, un paquete nuevo vale más que uno viejo. Esto se debe a que los paquetes se vuelven inútiles si se retardan y se pasa el tiempo en el que se deben reproducir para el usuario.

A la primera política (más viejo es mejor que más nuevo) se le conoce comúnmente como **vino** (*wine*) y a la segunda (más nuevo es mejor que más viejo) con frecuencia se le llama **leche** (*milk*), ya que la mayoría de las personas preferirían beber leche nueva y por lo contrario vino viejo.

Un desprendimiento de carga más inteligente requiere la cooperación de los emisores. Un ejemplo es el de los paquetes que transmiten información de enrutamiento. Estos paquetes son más importantes que los paquetes de datos regulares, ya que establecen rutas; si se extravían, la red puede perder conectividad. Otro ejemplo es el de los algoritmos para comprimir video, como MPEG, que transmiten en forma periódica toda una trama y después envían tramas como diferencias respecto a la última trama completa. En este caso es preferible desprenderse de un paquete que forma parte de una diferencia, que desprenderse de uno que forme parte de una trama completa, ya que los futuros paquetes dependerán de la trama completa.

Para implementar una política inteligente de descarte, las aplicaciones deben marcar sus paquetes para indicar a la red qué tan importantes son. Así, al tener que descartar paquetes, los enrutadores pueden eliminar primero los paquetes de la clase menos importante, luego los de la siguiente clase más importante, y así en lo sucesivo.

Por supuesto, a menos que haya una razón poderosa para evitar marcar todos los paquetes como MUY IMPORTANTE–NUNCA, NUNCA DESCARTAR, nadie lo hará. A menudo se usan la contabilidad y el dinero para disuadir una marcación frívola. Por ejemplo, la red podría permitir a los emisores enviar con más rapidez de la permitida por el servicio que compraron si marcan los paquetes en exceso como de baja prioridad. En realidad, dicha estrategia no es una mala idea, ya que hace un uso más eficiente de los recursos inactivos al permitir que los hosts los utilicen mientras que nadie más esté interesado, pero sin establecer un derecho sobre ellos cuando los tiempos se vuelven difíciles.

Detección temprana aleatoria

Es más efectivo lidiar con la congestión cuando apenas empieza que dejar que dañe la red y luego tratar de solucionarlo. Esta observación conduce a un giro interesante sobre el desprendimiento de carga: descartar paquetes antes de que se agote realmente el espacio en el búfer.

La motivación para esta idea es que la mayoría de los hosts de Internet todavía no obtienen señales de congestión de los enrutadores en la forma de ECN; la única indicación confiable de congestión que reciben los hosts de la red es la pérdida de paquetes. Después de todo, es difícil construir un enrutador que no descarte paquetes cuando esté sobrecargado. Por ende, los protocolos de transporte (como TCP) están predeterminados a reaccionar a la pérdida como congestión, y en respuesta reducen la velocidad de la fuente. El razonamiento detrás de esta lógica es que TCP se diseñó para redes cableadas y éstas son muy confiables, así que los paquetes que se pierden se deben principalmente a desbordamientos del búfer y no a errores de transmisión. Los enlaces inalámbricos deben recuperar los errores de transmisión en la capa de enlace (de modo que no se vean en la capa de red) para funcionar bien con TCP.

Podemos explotar esta situación para ayudar a reducir la congestión. Al hacer que los enrutadores descarten los paquetes antes de que la situación se vuelva imposible, hay tiempo para que la fuente tome acción antes de que sea demasiado tarde. Un algoritmo popular para realizar esto se conoce como **RED** (Detección Temprana Aleatoria, del inglés *Random Early Detection*). Para determinar cuándo hay que empezar a descartar paquetes, los enrutadores mantienen un promedio acumulado de sus longitudes de cola. Cuando la longitud de cola promedio en algún enlace sobrepasa un umbral, se dice que el enlace está congestionado y se descarta una pequeña fracción de los paquetes al azar. Al elegir los paquetes al azar es más probable que los emisores más rápidos vean un desprendimiento de paquetes; ésta es la mejor opción, ya que el enrutador no puede saber cuál fuente está causando más problemas en una red de datagramas. El emisor afectado observará la pérdida cuando no haya confirmación de recepción, y entonces el protocolo de transporte reducirá su

velocidad. Así, el paquete perdido entrega el mismo mensaje que un paquete regulador, pero de manera implícita sin que el enrutador envíe ninguna señal explícita.

Los enrutadores RED mejoran el desempeño en comparación con los enrutadores que sólo descartan paquetes cuando sus búferes están llenos, aunque tal vez requieran de un ajuste para funcionar bien. Por ejemplo, el número ideal de paquetes a descartar depende de cuántos emisores necesiten ser notificados sobre la congestión. Sin embargo, ECN es la opción preferida si está disponible. Funciona exactamente igual, sólo que entrega una señal de congestión de manera explícita en vez de hacerlo como una pérdida; RED se utiliza cuando los hosts no pueden recibir señales explícitas.

Calidad de servicio

Las técnicas que analizamos en las secciones anteriores se diseñaron para reducir la congestión y mejorar el rendimiento de la red. Sin embargo, existen aplicaciones (y clientes) que exigen a la red garantías más sólidas de desempeño que "*lo mejor que se pueda hacer en base a las circunstancias*". En especial, las aplicaciones multimedia necesitan con frecuencia una tasa de transferencia real mínima y una latencia máxima para trabajar. En esta sección continuaremos con nuestro estudio del desempeño de la red, sólo que ahora con un enfoque más pronunciado en las formas de proporcionar una calidad de servicio que coincida con las necesidades de la aplicación. Ésta es un área en la que Internet pasa por un proceso continuo de actualización.

Una solución sencilla para proporcionar una buena calidad del servicio es construir una red con la suficiente capacidad para cualquier tráfico que maneje. El nombre de esta solución es **exceso de aprovisionamiento** (*overprovisioning*). La red resultante transportará el tráfico de la aplicación sin pérdidas considerables y, suponiendo que hay un esquema de enrutamiento decente, entregará los paquetes con una latencia baja. El desempeño no puede ser mejor que esto. Simplemente hay tanta capacidad disponible, que casi siempre se puede satisfacer la demanda.

El problema con esta solución es que tiene un costo elevado. Básicamente el problema se resuelve con dinero. Los mecanismos de calidad del servicio permiten que una red con menos capacidad cumpla con los requerimientos de la aplicación con la misma eficiencia, a un menor costo. Además, el exceso de aprovisionamiento se basa en el tráfico esperado. No se puede asegurar nada si el patrón de tráfico cambia demasiado. Con los mecanismos de calidad del servicio, la red puede honrar las garantías de desempeño que hace incluso cuando hay picos de tráfico, a costa de rechazar algunas solicitudes.

Es necesario considerar cuatro aspectos para asegurar la calidad del servicio:

1. Lo que las aplicaciones necesitan de la red.
2. Cómo regular el tráfico que entra a la red.
3. Cómo reservar recursos en los enrutadores para garantizar el desempeño.
4. Si la red puede aceptar o no más tráfico en forma segura.

Ninguna técnica maneja todos estos aspectos en forma eficiente. Sin embargo, se ha desarrollado una variedad de técnicas para usarlas en la capa de red (y de transporte). Las soluciones prácticas de calidad del servicio combinan varias técnicas. Para este fin, describiremos dos versiones de calidad del servicio para Internet, conocidas como Servicios integrados y Servicios diferenciados.

Requerimientos de la aplicación

A un conjunto de paquetes que van de un origen a un destino se le denomina **flujo**. En una red orientada a conexión, un flujo podría estar constituido por todos los paquetes de una conexión; o en una red sin conexión, un fujo serían todos los paquetes enviados de un proceso a otro. Podemos caracterizar las necesidades de cada flujo mediante cuatro parámetros principales: **ancho de banda**, **retardo**, **variación del retardo** (*jitter*) y **pérdida**. En conjunto, estos parámetros determinan la **QoS** (Calidad del Servicio, del inglés *Quality of Service*) que requiere el flujo.

En la Ilustración 27 se listan varias aplicaciones comunes y el nivel de sus requerimientos de red. Observe que los requerimientos de red son menos exigentes que los requerimientos de aplicación en los casos en que esta última puede mejorar con base en el servicio proporcionado por la red. En particular, las redes no necesitan tener cero pérdidas para una transferencia de archivos confiable, y no necesitan entregar paquetes con

retardos idénticos para reproducir audio y video. Cierta cantidad de pérdida se puede reparar con las retransmisiones, y cierta cantidad de variación del retardo se puede solucionar si se colocan paquetes en el búfer del receptor. Sin embargo, no hay nada que las aplicaciones puedan hacer para remediar la situación si la red proporciona muy poco ancho de banda o demasiado retardo.

Aplicación	Ancho de banda	Retardo	Variación del retardo	Pérdida
Correo electrónico.	Bajo	Bajo	Baja	Media
Compartir archivos.	Alto	Bajo	Baja	Media
Acceso a Web.	Medio	Medio	Baja	Media
Inicio de sesión remoto.	Bajo	Medio	Media	Media
Audio bajo demanda.	Bajo	Bajo	Alta	Baja
Video bajo demanda.	Alto	Bajo	Alta	Baja
Telefonía.	Bajo	Alto	Alta	Baja
Videoconferencias.	Alto	Alto	Alta	Baja

Ilustración 27 - Nivel de los requerimientos de calidad del servicio de la aplicación.

Las aplicaciones difieren en cuanto a las necesidades de ancho de banda; las aplicaciones de correo electrónico (e-mail), audio en todas las formas e inicio de sesión remoto no necesitan mucho, pero los servicios de compartición de archivos y el video en todas las formas necesitan bastante ancho de banda.

Lo más interesante son los requerimientos de retardo. Las aplicaciones para transferir archivos, incluyendo e-mail y video, no son sensibles al retardo. Si todos los paquetes se retardan lo mismo por unos cuantos segundos, no hay problema. Las aplicaciones interactivas, como la navegación web y el inicio de sesión remoto, son más sensibles al retardo. Las aplicaciones en tiempo real, como la telefonía y las videoconferencias, tienen requerimientos estrictos en cuanto al retardo. Si todas las palabras en una llamada telefónica se retardan demasiado, a los usuarios les parecerá inaceptable la conversación. Por otro lado, para reproducir archivos de audio o video de un servidor no se requiere un retardo bajo.

A la variación (es decir, desviación estándar) en el retardo o los tiempos de llegada de los paquetes se le conoce como **variación del retardo** (jitter). Las primeras tres aplicaciones en la Ilustración 27 no son sensibles a que los paquetes lleguen con intervalos de tiempo irregulares entre ellos. El inicio de sesión remoto es algo sensible a esto, ya que las actualizaciones en la pantalla aparecerán en pequeñas ráfagas si la conexión sufre demasiada variación del retardo. El video y en especial el audio son en extremo sensibles a la variación del retardo. Si un usuario está viendo un video a través de la red y las tramas se retardan exactamente 2 000 segundos, no hay problema. Pero si el tiempo de transmisión varía al azar entre 1 y 2 segundos, el resultado será terrible a menos que la aplicación oculte la variación del retardo. Para el audio, incluso una variación de unos cuantos milisegundos se puede escuchar con claridad.

Las primeras cuatro aplicaciones tienen requerimientos más exigentes sobre la pérdida que el audio y el video, ya que todos los bits se deben entregar correctamente. Por lo general, para lograr este objetivo se retransmiten los paquetes que se pierden en la red mediante la capa de transporte. Esto es trabajo desperdiciado; sería mejor si la red rechazara los paquetes que pudiera llegar a perder en primer lugar. Las aplicaciones de audio y video pueden tolerar unos cuantos paquetes perdidos sin necesidad de retransmitirlos, ya que las personas no detectan las pausas cortas o los saltos ocasionales en las tramas.

Para admitir varias aplicaciones, las redes pueden soportar distintas categorías de QoS. Un ejemplo influyente proviene de las redes ATM, que alguna vez formaron parte de una gran visión para el uso de redes, pero desde entonces se han convertido en una tecnología de nicho. Éstas tienen soporte para:

1. Tasa de bits constantes (por ejemplo, la telefonía).
2. Tasa de bits variable en tiempo real (por ejemplo, videoconferencias con compresión).
3. Tasa de bits variable no en tiempo real (por ejemplo, ver una película bajo demanda).
4. Tasa de bits disponible (por ejemplo, transferencia de archivos).

Estas categorías también son útiles para otros fines y otras redes. La tasa de bits constante es un intento por simular un cable, al proporcionar un ancho de banda y un retardo uniformes. La tasa de bits variable ocurre cuando el video está comprimido y algunas tramas tienen más compresión que otras. Para enviar una trama con muchos detalles en ella, tal vez sea necesario enviar muchos bits, mientras que una toma de una pared blanca se puede comprimir extremadamente bien. Las películas bajo demanda en realidad no son en tiempo real, ya que unos cuantos segundos de video se pueden colocar fácilmente en el búfer del receptor antes de que empiece la reproducción, por lo que la variación del retardo en la red sólo hace que varíe la cantidad de video almacenado que no se ha reproducido todavía. La tasa de bits disponible es para aplicaciones como el correo electrónico, que no son sensibles al retardo o a la variación del retardo, y que toman el ancho de banda que se les asigne.

Modelado de tráfico

Antes de que la red pueda hacer garantías de QoS, debe saber qué tráfico está garantizando. En la red telefónica esta caracterización es simple. Por ejemplo, una llamada de voz (en formato descomprimido) necesita 64 kbps y consiste en una muestra de 8 bits cada 125 µs. Sin embargo, el tráfico en las redes de datos es **en ráfagas**. Por lo general llega a tasas de transmisión no uniformes a medida que varía el tráfico, los usuarios interactúan con las aplicaciones y las computadoras cambian de una tarea a otra. Las ráfagas de tráfico son más difíciles de manejar que el tráfico a una tasa constante, ya que pueden llenar búferes y hacer que se pierdan paquetes.

El **modelado de tráfico** (*traffic shaping*) es una técnica para regular la tasa promedio y las ráfagas de un flujo de datos que entra a la red. El objetivo es permitir que las aplicaciones transmitan una amplia variedad de tráfico que se adapte a sus necesidades (incluyendo algunas ráfagas) y tener al mismo tiempo una manera simple y útil de describir los posibles patrones de tráfico a la red. Al establecer un flujo, el usuario y la red (es decir, el cliente y el proveedor) se ponen de acuerdo sobre cierto patrón de tráfico (es decir, la forma) para ese flujo. En efecto, el cliente dice al proveedor: “Mi patrón de transmisión se verá así; ¿puedes manejarlo?”

Algunas veces este acuerdo se denomina **SLA** (Acuerdo de Nivel de Servicio, del inglés *Service Level Agreement*), en especial cuando se realiza sobre flujos agregados y periodos extensos, como todo el tráfico para un cliente específico. Mientras que el cliente cumpla con su parte del contrato y sólo envíe los paquetes acordados, el proveedor promete entregarlos de manera oportuna.

El modelado de tráfico reduce la congestión y, por ende, ayuda a la red a cumplir con su promesa. Sin embargo, para que funcione también surge la cuestión de cómo puede saber el proveedor si el cliente está cumpliendo con el acuerdo y qué debe hacer en caso contrario. Los paquetes que excedan el patrón acordado podrían ser descartados por la red, o se podrían marcar con una prioridad más baja. Al monitoreo de un flujo de tráfico se le conoce como **supervisión de tráfico** (*traffic policing*).

El modelado y la supervisión no son tan importantes para las transferencias de igual a igual y otras transferencias similares que consumen todo el ancho de banda disponible, pero son de gran importancia para los datos en tiempo real, como las conexiones de audio y video, que tienen requerimientos exigentes en cuanto a la calidad del servicio.

Cubetas con goteo y con token

Ya vimos una forma de limitar los datos que envía una aplicación: la ventana deslizante, que usa un parámetro para limitar la cantidad de datos en tránsito en un momento dado, lo cual limita la tasa de transmisión en forma indirecta. Ahora veremos una manera más general de caracterizar el tráfico, con los algoritmos de cubeta con goteo y cubeta con token. Las formulaciones son un poco distintas, pero producen un resultado equivalente.

Trate de imaginar un recipiente (cubeta) con un pequeño orificio en el fondo, como se ilustra en la Ilustración 28(b). Sin importar la rapidez con que el agua entra a la cubeta, el flujo de salida tiene una tasa constante, *R*, cuando hay agua en la cubeta, y cero cuando la cubeta está vacía. Además, una vez que la cubeta se llena a toda su capacidad *B*, cualquier cantidad adicional de agua que entre se derramará por los costados y se perderá.

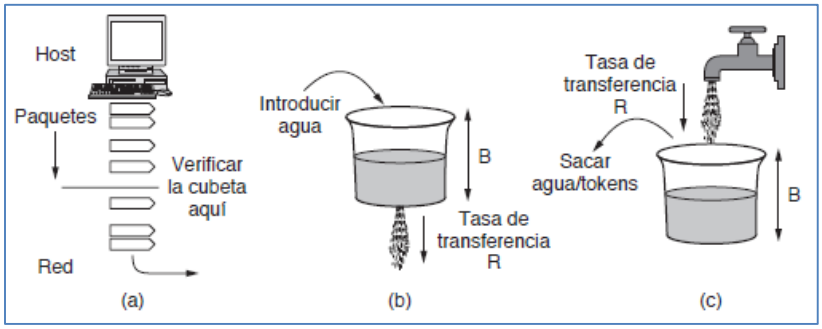


Ilustración 28 - (a) Modelado de paquetes. (b) Una cubeta con goteo. (c) Una cubeta con token.

Esta cubeta se puede usar para modelar o supervisar los paquetes que entran a la red, como se muestra en la Ilustración 28(a). Conceptualmente, cada host se conecta a la red mediante una interfaz que contiene una cubeta con goteo. Para enviar un paquete a la red, debe ser posible introducir más agua en la cubeta. Si un paquete llega cuando la cubeta está llena, debe ponerse en la cola hasta que se fugue la suficiente cantidad de agua como para poder contenerlo, o se debe descartar. Lo primero podría ocurrir en un host que modela su tráfico para la red como parte del sistema operativo. Lo segundo podría ocurrir en el hardware de una interfaz de red del proveedor, que supervise el tráfico que entra a la red. Turner propuso esta técnica, conocida como **algoritmo de la cubeta con goteo**.

Una formulación distinta pero equivalente es imaginar la interfaz de red como un recipiente (cubeta) que se está llenando, como se muestra en la Ilustración 28(c). El agua fluye por la llave a una tasa de transferencia *R* y la cubeta tiene una capacidad de *B*, como antes. Ahora, para enviar un paquete debemos tener la capacidad de sacar agua (o tokens, como se le dice normalmente al contenido) de la cubeta (en vez de meter agua en ella). En la cubeta no se puede acumular más de cierta cantidad fija de tokens, *B*, y si la cubeta está vacía, debemos esperar hasta que lleguen más tokens para poder enviar otro paquete. A este algoritmo se le conoce como **algoritmo de la cubeta con tokens**.

Las cubetas con goteo y con tokens limitan la tasa de transferencia a largo plazo de un flujo, pero dejan pasar ráfagas a corto plazo de hasta cierta longitud máxima regulada sin que se alteren ni sufran retardos artificiales. Las ráfagas extensas se controlarán mediante un modelador de tráfico de cubeta con goteo para reducir la congestión en la red. Como ejemplo, imagine que una computadora puede producir datos a una tasa de hasta 1000 Mbps (125 millones de bytes/s) y que el primer enlace de la red también opera a esta velocidad. El patrón de tráfico que genera el host se muestra en la Ilustración 29(a). Este patrón es en ráfagas. La tasa promedio durante un segundo es de 200 Mbps, aun cuando el host envía una ráfaga de 16000 KB a la máxima velocidad de 1000 Mbps (durante 1/8 de segundo).

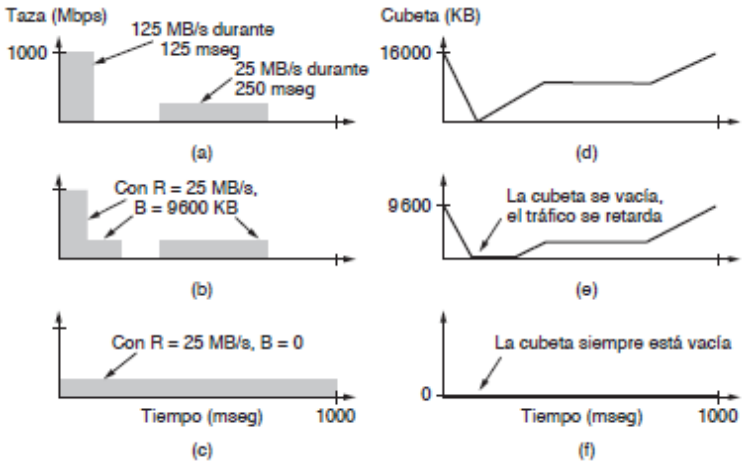


Ilustración 29 - (a) Tráfico proveniente de un host. La salida se modela mediante una cubeta de tokens con una tasa de 200 Mbps y con capacidades de (b) 9600 KB y (c) 0 KB. El nivel de la cubeta de tokens para modelar con una tasa de 200 Mbps y capacidades de (d) 16000 KB, (e) 9600 KB y (f) 0 KB.

Ahora suponga que los enrutadores pueden aceptar datos a la velocidad máxima sólo durante intervalos cortos, hasta que se llenen sus búferes. El tamaño del búfer es de 9600 KB, más pequeño que la ráfaga de tráfico. Durante intervalos largos, los enrutadores funcionan mejor con tasas que no se excedan de los 200 Mbps (por decir, ya que éste es todo el ancho de banda que recibe el cliente). La consecuencia es que si el tráfico se envía siguiendo este patrón; una parte se descartará en la red debido a que no cabe en los búferes de los enrutadores.

Para evitar esta pérdida de paquetes, podemos modelar el tráfico en el host mediante una cubeta con tokens. Si usamos una tasa *R*, de 200 Mbps y una capacidad *B* de 9600 KB, el tráfico se reducirá a una cantidad que la red pueda manejar. La salida de esta cubeta con tokens se muestra en la Ilustración 29(b). El host puede enviar con su máxima capacidad a 1000 Mbps durante un intervalo corto, hasta que haya drenado la cubeta. Después tiene que reducir la velocidad a 200 Mbps hasta que se haya enviado la ráfaga. El efecto es dispersar la ráfaga a través del tiempo, ya que era demasiado extensa como para manejarla toda a la vez. El nivel de la cubeta con tokens se muestra en la Ilustración 29(e). Empieza llena y se vacía a través de la ráfaga inicial. Cuando llega a cero, sólo se pueden enviar nuevos paquetes a la tasa a la que se está llenando el búfer; no puede haber más ráfagas sino hasta que la cubeta se haya recuperado. La cubeta se llena cuando no se envía tráfico y permanece sin cambios cuando se envía tráfico a la tasa de llenado.

También podemos modelar el tráfico para que tenga menos ráfagas. La Ilustración 29(c) muestra la salida de una cubeta con tokens con *R*=200 Mbps y una capacidad de 0. Éste es el caso extremo en el que el tráfico se regulariza por completo. No se permiten ráfagas y el tráfico entra a la red a una tasa estable. El nivel de cubeta correspondiente, que se muestra en la Ilustración 29(f), siempre está vacío. El tráfico se encola en el host para liberarlo a la red y siempre hay un paquete esperando ser enviado cuando sea posible.

Por último, la Ilustración 29(d) muestra el nivel para una cubeta con tokens con *R*=200 Mbps y una capacidad de *B*=16000 KB. Ésta es la cubeta con tokens más pequeñas por la que el tráfico puede pasar sin sufrir alteraciones. Se podría usar en un enrutador en la red para supervisar el tráfico que envía el host. Si el host envía tráfico que se conforma con base en la cubeta con tokens que acordó con la red, el tráfico pasará por esa misma cubeta de tokens en el enrutador del extremo de la red. Si el host envía a una tasa más rápida o con más ráfagas, la cubeta con tokens se quedará sin agua. Si esto ocurre, un supervisor de tráfico sabrá que el éste no era como se esperaba; entonces descartará los paquetes excesivos o reducirá su prioridad, dependiendo del diseño de la red. En nuestro ejemplo, la cubeta sólo se vacía unos momentos, al final de la ráfaga inicial, y después se recupera lo suficiente para la siguiente ráfaga.

Aun cuando hemos descrito cómo el agua fluye de manera continua hacia/desde la cubeta, las verdaderas implementaciones deben trabajar con cantidades discretas. Una cubeta con tokens se implementa mediante un contador para el nivel de la cubeta. El contador se incrementa *R/ΔT* unidades en cada pulso de reloj de *ΔT* segundos. En nuestro ejemplo anterior, serían 200 Kbits cada 1 ms. Cada vez que se envía una unidad de tráfico a la red se decrementa el contador, y se puede enviar tráfico hasta que el contador llegue a cero.

Cuando todos los paquetes son del mismo tamaño, el nivel de la cubeta se puede contar en paquetes (por ejemplo, 200 Mbit equivale a 20 paquetes de 1250 bytes). Sin embargo, es común que se utilicen paquetes de tamaños variables. En este caso, el nivel de la cubeta se cuenta en bytes. Si la cuenta de bytes restantes es demasiado baja como para enviar un paquete extenso, éste debe esperar hasta el siguiente pulso de reloj (o incluso más, si la tasa de llenado es baja).

Es un poco complicado calcular la longitud de la máxima ráfaga (hasta que se vacíe la cubeta). Es más larga que el resultado de dividir 9600 KB entre 125 MB/s, ya que mientras se está enviando la ráfaga llegan más tokens. Si designamos la longitud de la ráfaga como *S* segundos, la tasa máxima de salida como *M* bytes/s, la capacidad de la cubeta con tokens como *B* bytes y la tasa de llegada de tokens como *R* bytes/s, podremos ver que una ráfaga de salida contiene un máximo de *B+RS* bytes. También sabemos que el número de bytes en una ráfaga a máxima velocidad con longitud de *S* segundos es *MS*. Por ende, tenemos que

$$B + RS = MS$$

Podemos resolver esta ecuación para obtener *S=B/(M-R)*. Para nuestros parámetros de *B*=9600 KB, *M*=25 MB/s y *R*=25 MB/s, obtenemos un tiempo de ráfaga aproximado de 94 ms.

Un problema potencial con el algoritmo de cubeta con tokens es que reduce las ráfagas largas hasta la tasa *R* de largo plazo. Con frecuencia es deseable reducir la tasa pico, pero sin bajar hasta la tasa de largo plazo (y también sin elevar la tasa de largo plazo para permitir que entre más tráfico en la red). Una forma de obtener un tráfico más uniforme es insertar una segunda cubeta con tokens después de la primera. La tasa de la segunda cubeta deberá ser mucho mayor que la primera. Básicamente, la primera caracteriza el tráfico; corrige su tasa promedio pero permite algunas ráfagas. La segunda reduce la tasa pico a la que se envían las ráfagas a la red. Por ejemplo, si la tasa de la segunda cubeta con tokens se establece en 500 Mbps y la capacidad en 0, la ráfaga inicial entrará a la red a una tasa pico de 500 Mbps, lo cual es más bajo que la tasa de 1000 Mbps que teníamos antes.

Usar estas cubetas puede ser algo complicado. Cuando se usan con tokens para modelar el tráfico en los hosts, los paquetes se encolan y retardan hasta que las cubetas permitan que se envíen. Cuando se usan cubetas con tokens para supervisar tráfico en los enrutadores en la red, se simula el algoritmo para asegurar que no se envíen más paquetes de los permitidos. Sin embargo, estas herramientas ofrecen maneras de modelar el tráfico de la red en formas más manejables para ayudar a cumplir con los requerimientos de calidad del servicio.

Programación de paquetes

Tener la capacidad de regular la forma del tráfico ofrecido es un buen comienzo. Sin embargo, para ofrecer una garantía de desempeño debemos reservar suficientes recursos a lo largo de la ruta que toman los paquetes a través de la red. Para ello, supongamos que los paquetes de un flujo siguen la misma ruta. Si los dispersamos a través de rutas al azar, es difícil garantizar algo. Como consecuencia, es necesario establecer algo similar a un circuito virtual desde el origen hasta el destino, y todos los paquetes que pertenecen al flujo deben seguir esta ruta.

Los algoritmos que asignan recursos de enrutadores entre los paquetes de un flujo y entre los flujos competidores se llaman **algoritmos de programación de paquetes**. Se pueden reservar potencialmente tres tipos distintos de recursos para diversos flujos:

1. Ancho de banda.
2. Espacio de búfer.
3. Ciclos de CPU.

El primero, ancho de banda, es el más obvio. Si un flujo requiere 1 Mbps y la línea de salida tiene una capacidad de 2 Mbps, no va a ser posible tratar de dirigir tres flujos a través de esa línea. Por lo tanto, reservar ancho de banda significa no sobrecargar ninguna línea de salida.

Un segundo recurso que por lo general escasea es el espacio en búfer. Cuando llega un paquete, se pone en un búfer dentro del enrutador hasta que se pueda transmitir en la línea de salida elegida. El propósito del búfer es absorber pequeñas ráfagas de tráfico a medida que los flujos compiten entre sí. Si no hay búfer disponible, el paquete se tiene que descartar debido a que no hay lugar para colocarlo. Para una buena calidad de servicio, se deberían reservar algunos búferes para un flujo específico de manera que éste no tenga que competir con otros flujos para obtener el espacio del búfer. Siempre que ese flujo necesite un búfer, se le proporcionará uno hasta cierto valor máximo.

Por último, los ciclos de CPU también pueden ser un recurso escaso. Para procesar un paquete se necesita tiempo de CPU del enrutador, por lo que un enrutador sólo puede procesar cierta cantidad de paquetes por segundo. Aunque los enrutadores modernos pueden procesar la mayoría de los paquetes rápidamente, algunos tipos de paquetes requieren más procesamiento de la CPU, como los paquetes ICMP que veremos luego. Para asegurar el procesamiento oportuno de estos paquetes, es necesario asegurarse que la CPU no esté sobrecargada.

Para asignar ancho de banda y otros recursos del enrutador, los algoritmos de programación de paquetes determinan cuál de los paquetes en el búfer van a enviar en la línea de salida a continuación. Ya describimos el programador más simple al explicar cómo funcionan los enrutadores. Cada enrutador coloca los paquetes en una cola de búfer para cada línea de salida hasta que se puedan enviar, y se envían en el mismo orden en

el que llegaron. Este algoritmo se conoce como **FIFO** (Primero en Entrar, Primero en Salir, del inglés *First-In First-Out*) o su equivalente, **FCFS** (Primero en Llegar, Primero en Servir, del inglés *First-Come, First-Serve*).

Por lo general, los enrutadores FIFO descartan los paquetes recién llegados cuando la cola está llena. Como el paquete recién llegado se hubiera colocado al final de la cola, a este comportamiento se le conoce como **descarte trasero** (*tail drop*). Es intuitivo y tal vez se pregunte qué alternativas existen. De hecho, el algoritmo RED que ya describimos elegía un paquete recién llegado para descartarlo al azar cuando la longitud promedio de la cola aumentaba mucho. Los otros algoritmos de programación que describiremos también crean otras oportunidades para decidir qué paquete descartar cuando los búferes están llenos.

La programación FIFO es simple de implementar, pero no es adecuada para proporcionar una buena calidad del servicio, ya que cuando hay varios flujos, uno de ellos puede afectar fácilmente el desempeño de los demás. Si el primer flujo es agresivo y envía grandes ráfagas de paquetes, se depositarán en la cola. Procesar paquetes con base en su orden de llegada significa que el emisor agresivo puede acaparar la mayor parte de la capacidad de los enrutadores que sus paquetes recorren, privando a los demás flujos y reduciendo su calidad de servicio. Para empeorar las cosas, es probable que los paquetes de los otros flujos que logren pasar se retarden, ya que tuvieron que sentarse en la cola detrás de muchos paquetes provenientes del emisor agresivo.

Se han ideado muchos algoritmos de programación de paquetes que proporcionan un aislamiento más sólido entre los flujos y frustran los intentos de interferencia. Uno de los primeros fue el algoritmo de **encolamiento justo** ideado por Nagle. La esencia de este algoritmo es que los enrutadores tienen colas separadas, una por cada flujo para una línea de salida dada. Cuando la línea se vuelve inactiva, el enrutador explora de forma circular (*round-robin*) las colas, como se muestra en la Ilustración 30. Entonces toma el primer paquete de la siguiente cola. Así, con *n* hosts que compiten por la línea de salida, cada host tiene la oportunidad de enviar uno de cada *n* paquetes. Es justo en el sentido de que todos los flujos pueden enviar paquetes a la misma tasa. Enviar más paquetes no mejorará esta tasa.

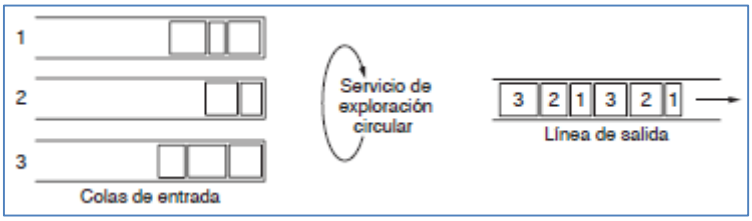
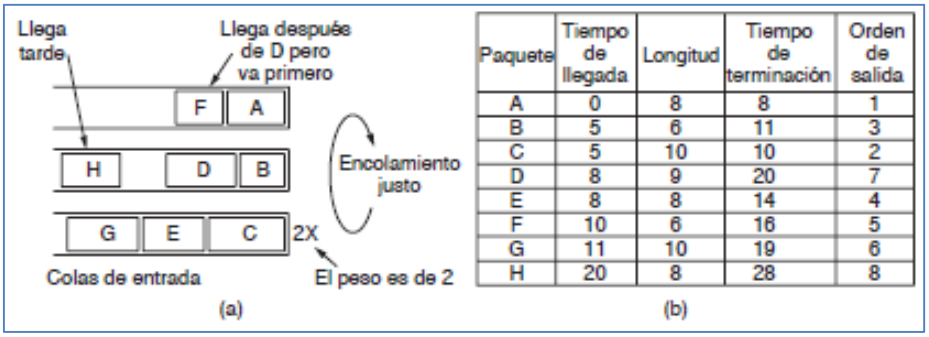


Ilustración 30 - Encolamiento justo por exploración circular (round-robin).

Aunque es un comienzo, este algoritmo tiene una falla: proporciona más ancho de banda a los hosts que utilizan paquetes grandes que a los que utilizan paquetes pequeños. Demers y sus colaboradores sugirieron una mejora en la que la exploración circular se realiza de tal forma que se simule una exploración circular byte por byte, en vez de paquete por paquete. El truco es calcular un tiempo virtual que sea el número de la ronda en la que cada paquete terminará de ser enviado. Cada ronda drena un byte de todas las colas que tienen datos por enviar. Entonces los paquetes se ordenan conforme a su tiempo de terminación y se envían en ese orden.

En la Ilustración 31 se muestra este algoritmo con un ejemplo de los tiempos de terminación para los paquetes que llegan en tres flujos. Si un paquete tiene una longitud *L*, la ronda en la que terminará es simplemente *L* rondas después del tiempo de inicio, que puede ser el tiempo de terminación del paquete anterior o el tiempo de llegada del paquete, si la cola está vacía cuando llega.



Paquete	Tiempo de llegada	Longitud	Tiempo de terminación	Orden de salida
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

Ilustración 31 - (a) Encolamiento justo ponderado. (b) Tiempos de terminación para los paquetes.

Si observamos la tabla de la Ilustración 31(b), y analizamos sólo los primeros dos paquetes en las dos colas superiores, nos daremos cuenta de que los paquetes llegan en el orden *A*, *B*, *D* y *F*. El paquete *A* llega en la ronda 0 y tiene 8 bytes de longitud, por lo que su tiempo de terminación es en la ronda 8. De manera similar, el tiempo de terminación para el paquete *B* es 11. El paquete *D* llega mientras se está enviando *B*. Su tiempo de terminación es 9 rondas de bytes después de que empieza cuando *B* termina, o 20. Asimismo, el tiempo de terminación para *F* es 16. Como no llegan paquetes nuevos, el orden de envío relativo es A, B, F, D, aun cuando *F* llegó después de *D*. Es posible que llegue otro pequeño paquete en el flujo superior y que obtenga un tiempo de terminación antes que *D*. Sólo se adelantará a *D* si no ha empezado la transmisión de ese paquete. El encolamiento justo no sustituye los paquetes que ya se están transmitiendo. Como los paquetes se envían en su totalidad, el encolamiento justo sólo es una aproximación del esquema ideal de byte por byte. Pero es una muy buena aproximación que en todo momento se mantiene con una diferencia de un paquete con respecto a la transmisión mediante el esquema ideal.

Una desventaja de este algoritmo en la práctica es que da la misma prioridad a todos los hosts. En muchas situaciones es conveniente dar a los servidores de video más ancho de banda que a los servidores de archivos, por dar un ejemplo. Para hacer esto sólo es necesario dar al servidor de video dos o más bytes por ronda. Este algoritmo modificado se llama **WFQ** (Encolamiento Justo Ponderado, del inglés *Weighted Fair Queueing*).

La cola inferior de la Ilustración 31(a) tiene un peso de 2, por lo que sus paquetes se envían con más rapidez como podemos ver en los tiempos de terminación que se muestran en la Ilustración 31(b).

También existen otros tipos de algoritmos de programación. La programación por prioridad es un ejemplo simple, en donde cada paquete se marca con una prioridad. Los paquetes de prioridad alta siempre se envían antes que los de prioridad baja que estén en el búfer. Si tienen la misma prioridad, los paquetes se envían en orden FIFO. Sin embargo, la programación por prioridad tiene la desventaja de que una ráfaga de paquetes de alta prioridad puede privar a los paquetes de baja prioridad, que tal vez tengan que esperar de manera indefinida. WFQ ofrece por lo general una mejor alternativa. Al asignar a la cola de alta prioridad un peso grande (por decir, 3), por lo general los paquetes de alta prioridad pasarán a través de una línea corta (puesto que relativamente debe haber pocos paquetes de alta prioridad) y se seguirá enviando cierta fracción de paquetes de baja prioridad, aun cuando haya tráfico de alta prioridad. Un sistema de baja y alta prioridad es en esencia un sistema WFQ de dos colas, en donde la prioridad alta tiene un peso infinito.

Como ejemplo final de un programador, los paquetes podrían incluir etiquetas de tiempo para enviarse en ese orden. Existe un diseño en el que la etiqueta de tiempo registra qué tan atrasado o adelantado está el paquete según la programación, a medida que se envía a través de una secuencia de enrutadores por la ruta designada. Los paquetes que se hayan puesto en cola detrás de otros paquetes en un enrutador tenderán a estar atrasados, y los paquetes que se hayan atendido primero tenderán a estar adelantados. El efecto benéfico de enviar paquetes con base en el orden de sus etiquetas de tiempo es que los paquetes lentos se agilizan, al tiempo que se reduce la velocidad de los paquetes rápidos. El resultado es que la red entrega todos los paquetes con un retardo más consistente.

Control de admisión

Ya hemos visto todos los elementos necesarios para la calidad del servicio (QoS); es tiempo de reunirlos para realmente proporcionarla. Las garantías de QoS se establecen por medio del proceso de control de admisión.

Primero vimos cómo se utilizaba el control de admisión para controlar la congestión, lo cual es una garantía de desempeño, aunque algo débil. Las garantías que consideraremos ahora son más sólidas, pero el modelo es el mismo. El usuario ofrece un flujo junto con un requerimiento de QoS para la red. Después la red decide si acepta o rechaza el flujo, con base en su capacidad y a los compromisos que tiene con otros flujos. Si acepta, reserva la capacidad por adelantado en los enrutadores para garantizar la QoS cuando se envíe el tráfico por el nuevo flujo.

Es necesario hacer las reservaciones en todos los enrutadores a lo largo de la ruta que toman los paquetes a través de la red. Cualquier enrutador que esté en la ruta sin reservaciones podría congestionarse, y un solo enrutador congestionado puede quebrantar la garantía de QoS. Muchos algoritmos de enrutamiento encuentran la mejor ruta entre cada origen y cada destino, y envían todo el tráfico a través de la mejor ruta. Esto puede ocasionar que se rechacen algunos flujos si es que no hay suficiente capacidad disponible a lo largo de la mejor ruta. De todas formas, tal vez sea posible adoptar las garantías de QoS para los nuevos flujos si se elige una ruta distinta para el flujo con la capacidad excesiva. Esto se llama **enrutamiento con QoS**. También es posible dividir el tráfico para cada destino a través de varias rutas, de modo que sea más fácil encontrar la capacidad adicional. Un método simple es que los enrutadores seleccionen rutas de igual costo y dividan el tráfico en forma equitativa, o en proporción a la capacidad de los enlaces de salida. Sin embargo, también existen algoritmos más sofisticados.

Dada una ruta, la decisión de aceptar o rechazar un flujo no es una simple cuestión de comparar los recursos (ancho de banda, búfer, ciclos) solicitados por el flujo con la capacidad excesiva del enrutador en esas tres dimensiones. Es un poco más complicado que eso. Para empezar, aunque tal vez ciertas aplicaciones conozcan sus requerimientos de ancho de banda, pocas conocen sobre los requerimientos de búfer o ciclos de CPU, por lo que como mínimo se requiere una manera distinta de describir los flujos y traducir esta descripción para los recursos del enrutador.

Ahora bien, algunas aplicaciones son más tolerantes que otras en cuanto a no cumplir con uno que otro plazo límite. Las aplicaciones deben elegir de entre el tipo de garantías que puede hacer la red, ya sean garantías rígidas o un comportamiento que sea suficiente la mayor parte del tiempo. Si todo lo demás fuera igual, a todos les gustaría trabajar con garantías rígidas, pero la dificultad es que son costosas debido a que restringen el comportamiento en el peor de los casos. Con frecuencia las garantías para la mayoría de los paquetes son suficientes para las aplicaciones, y se pueden soportar más flujos con esta clase de garantía para una capacidad fija.

Por último, tal vez algunas aplicaciones estén dispuestas a regatear en cuanto a los parámetros de los flujos, pero otras no. Por ejemplo, un cinéfilo que por lo general vea películas a 30 tramas/s podría estar dispuesto a cambiar a 25 tramas/s si no hay suficiente ancho de banda. De manera similar, tal vez se puedan ajustar el número de píxeles por trama, el ancho de banda de audio y otras propiedades.

Como puede haber muchas partes involucradas en la negociación del flujo (el emisor, el receptor y todos los enrutadores a lo largo de la ruta entre ellos), debemos describir los flujos con precisión en términos de los parámetros específicos que se pueden negociar. Un conjunto de tales parámetros se denomina **especificación de flujo**. Por lo general, el emisor (por ejemplo, el servidor de video) produce una especificación de flujo que propone los parámetros que le gustaría usar. A medida que se propaga la especificación a lo largo de la ruta, cada enrutador la examina y modifica los parámetros según sus necesidades. Las modificaciones sólo pueden reducir el flujo, no incrementarlo (por ejemplo, una tasa de datos más baja, no una más alta). Cuando la especificación llega al otro extremo, se pueden establecer los parámetros.

Como ejemplo de lo que puede haber en una especificación de flujo, considere el de la Ilustración 32, que se basa en los RFC 2210 y 2211 para los Servicios integrados, un diseño de QoS que cubriremos en la siguiente sección. Tiene cinco parámetros. Los primeros dos, la tasa de la cubeta con tokens y el tamaño de la cubeta con tokens, utilizan una cubeta con tokens para proporcionar la tasa máxima sostenida a la que puede transmitir el emisor, promediada con respecto a un intervalo largo y la ráfaga más grande que puede enviar durante un intervalo corto.

El tercer parámetro, la tasa pico de datos, es la tasa máxima de transmisión tolerada, incluso durante intervalos breves. El emisor nunca debe sobrepasar esta tasa, ni siquiera en ráfagas cortas.

Los últimos dos parámetros especifican los tamaños mínimo y máximo de paquetes, incluyendo los encabezados de la capa de transporte y de red (por ejemplo, TCP e IP). El tamaño mínimo es importante porque procesar cada paquete toma un tiempo fijo, aunque sea corto. Un enrutador podrá estar preparado para manejar 10000 paquetes/s de 1 KB cada uno, pero no para manejar 100000 paquetes/s de 50 bytes cada uno, aunque esto represente una tasa de datos menor. El tamaño máximo de paquete es importante debido a las limitaciones internas de la red que no deben sobrepasarse. Por ejemplo, si parte de la ruta pasa a través de una Ethernet, el tamaño máximo del paquete se restringirá a no más de 1500 bytes, sin importar lo que el resto de la red pueda manejar.

Parámetro	Unidad
Tasa de la cubeta con tokens.	Bytes/seg
Tamaño de la cubeta con tokens.	Bytes
Tasa pico de datos.	Bytes/seg
Tamaño mínimo de paquete.	Bytes
Tamaño máximo de paquete.	Bytes

Ilustración 32 - Ejemplo de una especificación de flujo.

Una pregunta interesante es cómo convierte un enrutador una especificación de flujo en un conjunto de reservaciones de recursos específicos. A primera instancia, podría parecer que si un enrutador tiene un enlace que opera a, por decir, 1 Gbps y el paquete promedio es de 1000 bits, puede procesar 1 millón de paquetes/s. Pero esta observación no es verdadera, ya que siempre habrá periodos inactivos en el enlace debido a las fluctuaciones estadísticas en la carga. Si el enlace necesita toda la capacidad disponible para realizar su trabajo, al estar inactivo incluso por unos cuantos bits se generará una acumulación de la que nunca se podrá deshacer.

Incluso con una carga ligeramente por debajo de la capacidad teórica, se pueden generar colas y pueden ocurrir retardos.

Un método para relacionar las especificaciones de flujo con los recursos de enrutador que correspondan con las garantías de desempeño de ancho de banda y retardo se basa en modelar las fuentes de tráfico mediante cubetas de tokens (*R*, *B*) y WFQ en los enrutadores. Cada flujo recibe un peso *W* de WFQ lo suficientemente grande como para drenar su cubeta de tokens a la tasa *R*, como se muestra en la Ilustración 33. Por ejemplo, si el flujo tiene una tasa de 1 Mbps y tanto el enrutador como el enlace de salida tienen una capacidad de 1 Gbps, el peso para el flujo debe ser mayor que 1/1000 parte del total de los pesos para todos los flujos en ese enrutador para el enlace de salida. Esto garantiza al flujo un ancho de banda mínimo. Si no puede proporcionar una tasa suficientemente grande, no se puede admitir el flujo.

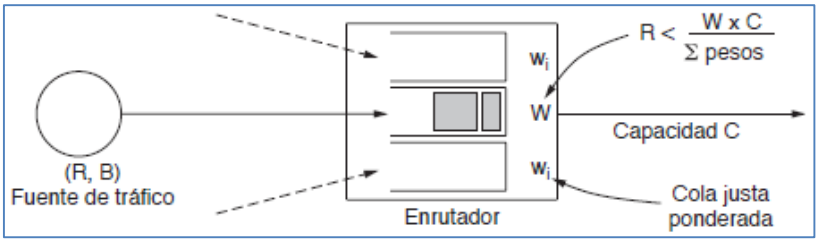


Ilustración 33 - Garantías de ancho de banda y retardo mediante cubetas con tokens y WFQ.

El retardo de encolamiento más grande que verá el flujo es en función del tamaño de ráfaga de la cubeta con tokens. Considere los dos casos extremos. Si el tráfico es uniforme y sin ráfagas, los paquetes se drenarán del enrutador con la misma rapidez con que llegan. No habrá retardo de encolamiento (ignorando los efectos de la paquetización). Por otro lado, si el tráfico se acumula en ráfagas, entonces puede llegar una ráfaga de tamaño máximo *B* al enrutador, toda a la vez. En este caso, el retardo de encolamiento máximo *D* será el tiempo requerido para drenar esta ráfaga con base en el ancho de banda garantizado, o *B/R* (de nuevo,

ignorando los efectos de la paquetización). Si este retardo es demasiado grande, el flujo debe solicitar más ancho de banda a la red.

Estas garantías son rígidas. Las cubetas con tokens limitan las ráfagas de la fuente y el encolamiento justo aísla el ancho de banda que se proporciona a los distintos flujos. Esto significa que el flujo cumplirá con sus garantías de ancho de banda y retardo sin importar cómo se comporten los demás flujos competidores en el enrutador. Esos otros flujos no podrán quebrantar la garantía, ni siquiera al acumular tráfico y enviarlo todo a la vez.

Además, el resultado es válido para una ruta a través de varios enrutadores en cualquier topología de red. Cada flujo obtiene un ancho de banda mínimo, ya que ese ancho de banda está garantizado en cada enrutador. La razón por la que cada flujo obtiene un retardo máximo es más sutil. En el peor caso en que una ráfaga de tráfico llegue al primer enrutador y compita con el tráfico de otros flujos, se retardará hasta un tiempo máximo de *D*. Sin embargo, este retardo también regulará la ráfaga. A su vez, esto significa que la ráfaga no incurrirá en más retardos de encolamiento en los siguientes enrutadores. A lo más, el retardo de encolamiento general será de *D*.

Servicios integrados

Entre 1995 y 1997, la IETF se esforzó mucho en diseñar una arquitectura para la multimedia de flujos continuos. Este trabajo generó cerca de dos docenas de RFC, empezando con los RFC 2205–2212. El nombre genérico para este trabajo es **servicios integrados**. Se diseñó tanto para aplicaciones de unidifusión como para multidifusión. Un ejemplo de la primera es un solo usuario recibiendo en flujo continuo un clip de video de un sitio de noticias. Un ejemplo de la segunda es una colección de estaciones de televisión digital difundiendo sus programas como flujos de paquetes IP a muchos receptores en distintas ubicaciones. A continuación nos concentraremos en la multidifusión, puesto que la unidifusión es un caso especial de la multidifusión.

En muchas aplicaciones de multidifusión, los grupos pueden cambiar su membresía en forma dinámica; por ejemplo, conforme las personas entran a una videoconferencia, se aburren y cambian a una telenovela o al canal del juego de deportes. Bajo estas condiciones, el método de hacer que los emisores reserven ancho de banda por adelantado no funciona bien, debido a que requeriría que cada emisor rastreara todas las entradas y salidas de su audiencia. En un sistema diseñado para transmitir televisión con millones de suscriptores, ni siquiera funcionaría.

RSVP: el protocolo de reservación de recursos

La parte principal de la arquitectura de servicios integrados visible para los usuarios de la red es **RSVP**. Se describe en los RFC 2205-2210. Este protocolo se utiliza para hacer las reservaciones; se usan otros protocolos para enviar los datos. RSVP brinda la posibilidad de que varios emisores transmitan a múltiples grupos de receptores, permite que receptores individuales cambien de canal libremente, optimiza el uso de ancho de banda y al mismo tiempo elimina la congestión.

En su forma más simple, el protocolo usa enrutamiento de multidifusión con árboles de expansión, como vimos antes. A cada grupo se le asigna una dirección. Para enviar a un grupo, un emisor pone la dirección asignada en sus paquetes. El algoritmo estándar de enrutamiento multidifusión construye entonces un árbol de expansión que cubre a todos los miembros del grupo. El algoritmo de enrutamiento no es parte de RSVP. La única diferencia con la multidifusión normal es un poco de información adicional que se transmite por multidifusión al grupo en forma periódica, para indicarle a los enrutadores a lo largo del árbol que mantengan ciertas estructuras de datos en sus memorias.

Como ejemplo, considere la red de la Ilustración 34(a). Los hosts 1 y 2 son emisores multidifusión, y los hosts 3, 4 y 5 son receptores multidifusión. En este ejemplo, los emisores y los receptores están separados pero, en general, los dos grupos se pueden traslapar. Los árboles de multidifusión de los hosts 1 y 2 se muestran en las Ilustraciones 34(b) y 34(c), respectivamente.

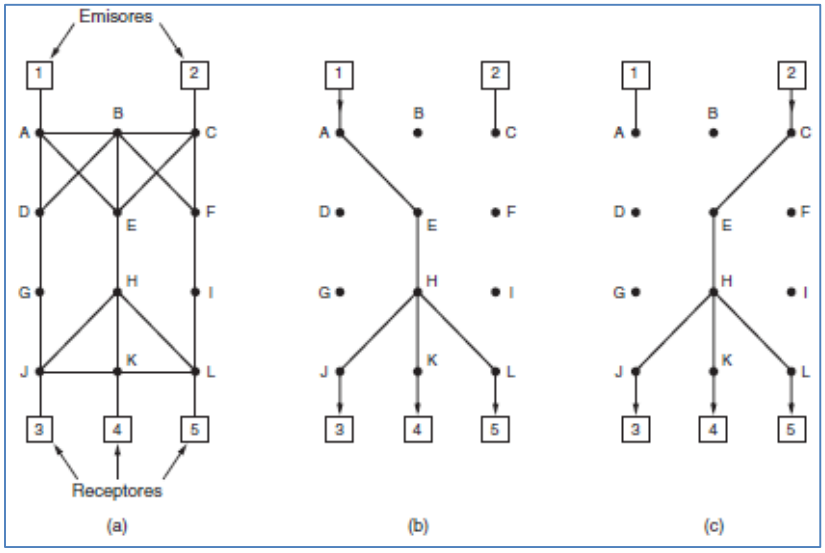


Ilustración 34 - (a) Una red. (b) El árbol de expansión de multidifusión para el host 1. (c) El árbol de expansión de multidifusión para el host 2.

Para obtener una mejor recepción y eliminar la congestión, cualquiera de los receptores en un grupo puede enviar un mensaje de reservación al emisor a través del árbol. El mensaje se propaga mediante el algoritmo de reenvío por ruta invertida. En cada salto, el enrutador nota la reservación y aparta el ancho de banda necesario. En la sección anterior vimos cómo se puede usar un programador de encolamiento justo ponderado para hacer esa reservación. Si no hay suficiente ancho de banda disponible, informa sobre una falla. Para cuando el mensaje regresa a la fuente, se ha reservado ancho de banda por toda la trayectoria desde el emisor hasta el receptor que hace la solicitud de reservación a lo largo del árbol de expansión.

En la Ilustración 35(a) se muestra un ejemplo de tales reservaciones. Aquí el host 3 ha solicitado un canal al host 1. Una vez establecido el canal, los paquetes pueden fluir de 1 a 3 sin congestiones. Ahora considere lo que ocurre si el host 3 reserva a continuación un canal hacia otro emisor, el host 2, para que el usuario pueda ver dos programas de televisión a la vez. Se reserva una segunda ruta, como se muestra en la Ilustración 35(b). Cabe mencionar que se requieren dos canales individuales del host 3 al enrutador *E* porque se están transmitiendo dos flujos independientes.

Por último, en la Ilustración 35(c) el host 5 decide observar el programa transmitido por el host 1 y también hace una reservación. Primero se reserva ancho de banda dedicado hasta el enrutador *H*. Sin embargo, éste ve que ya tiene una alimentación del host 1, por lo que si ya se ha reservado el ancho de banda necesario, no necesita reservar más. Observe que los hosts 3 y 5 podrían haber solicitado diferentes cantidades de ancho de banda (por ejemplo, el host 3 tiene una televisión pequeña y sólo quiere la información de baja resolución), así que la capacidad reservada debe ser lo bastante grande como para satisfacer hasta el receptor más demandante.

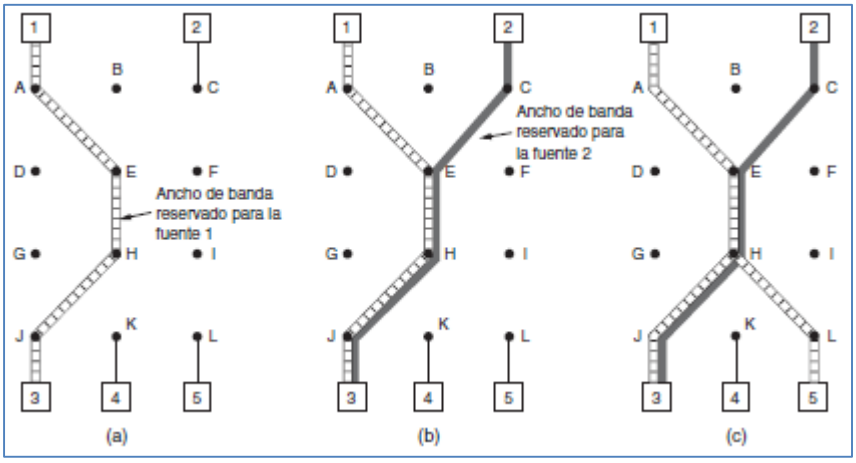


Ilustración 35 - a) El host 3 solicita un canal al host 1. (b) Después el host 3 solicita un segundo canal, al host 2. (c) El host 5 solicita un canal al host 1.

Al hacer una reservación, un receptor puede especificar (de manera opcional) una o más fuentes de las que desea recibir información. También puede especificar si estas selecciones quedarán fijas durante toda la reservación, o si el receptor quiere mantener abierta la opción de cambiar después las fuentes. Los enrutadores usan esta información para optimizar la planeación del ancho de banda. En particular, sólo se establecen dos receptores para compartir una ruta si ambos están de acuerdo con no cambiar las fuentes posteriormente.

La razón de esta estrategia en el caso totalmente dinámico es que el ancho de banda reservado está desacoplado de la selección de la fuente. Una vez que un receptor ha reservado ancho de banda, puede cambiar a otra fuente y conservar esa parte de la ruta existente que sea válida para la nueva fuente. Si el host 2 está transmitiendo varios flujos de video en tiempo real (como un difusor de TV con varios canales), el host 3 puede cambiar entre ellos a voluntad sin modificar su reservación: a los enrutadores no les importa el programa que está viendo el receptor.

Servicios diferenciados

Los algoritmos basados en flujo tienen el potencial de ofrecer buena calidad de servicio a uno o más flujos, debido a que reservan los recursos necesarios a lo largo de la ruta. Sin embargo, también tienen una desventaja. Requieren una configuración avanzada para establecer cada flujo, algo que no se escala bien cuando hay miles o millones de flujos. Además, mantienen el estado por flujo interno en los enrutadores, lo cual los hace vulnerables a las fallas de éstos. Por último, los cambios requeridos al código de enrutador son considerables e involucran intercambios complejos de enrutador a enrutador para establecer los flujos. Como consecuencia, aunque el trabajo continúa para mejorar los servicios integrados, existen pocas implementaciones de RSVP o algo parecido.

Por estas razones, la IETF también ha diseñado un método más simple para la calidad del servicio, uno que se pueda implementar ampliamente de manera local en cada enrutador sin una configuración avanzada y sin que toda la ruta esté involucrada. Este método se conoce como calidad de servicio **basada en clase** (en contraste a la basada en flujo). La IETF ha estandarizado una arquitectura para él, conocida como **servicios diferenciados**, que se describe en los RFC 2474, 2475 y varios más.

Un conjunto de enrutadores que forman un dominio administrativo (por ejemplo, un ISP o una compañía telefónica) puede ofrecer los servicios diferenciados. La administración define un conjunto de clases de servicios con sus correspondientes reglas de reenvío. Si un cliente se suscribe a los servicios diferenciados, los paquetes del cliente que entran en el dominio se marcan con la clase a la que pertenecen. Esta información se transporta en el campo Servicios diferenciados (*Differentiated services*) de los paquetes IPv4 e IPv6. Las clases se definen como **comportamientos por salto**, ya que corresponden al trato que recibirá el paquete en cada enrutador y no a una garantía a través de la red. Se proporciona un mejor servicio a los paquetes con ciertos comportamientos por salto (por ejemplo, *servicio premium*) que a otros paquetes (por ejemplo, *servicio regular*). Al tráfico dentro de una clase se le podría requerir que se apegue a algún modelo específico, como a

una cubeta con goteo y con cierta tasa especificada de drenado. Un operador con intuición para los negocios podría cobrar una cantidad adicional por cada paquete premium transportado, o podría permitir hasta *N* paquetes premium al mes por una mensualidad adicional fija. Cabe mencionar que este esquema no requiere una configuración avanzada o una reserva de recursos, ni tampoco una negociación extremo a extremo que consuma tiempo para cada flujo, como sucede con los servicios integrados. Esto hace que los servicios diferenciados sean relativamente sencillos de implementar.

El servicio basado en clase también ocurre en otras industrias. Por ejemplo, las compañías de envío de paquetes con frecuencia ofrecen servicio de tres días, de dos días, y servicio de un día para otro. Para los paquetes, las clases pueden diferir en términos de retardo, variación del retardo y probabilidad de que se descarten en caso de congestión, entre otras posibilidades.

Para hacer que la diferencia entre la calidad de servicio basada en flujo y la basada en clase sea más clara, considere un ejemplo: la telefonía de Internet. Con un esquema basado en flujo, cada llamada telefónica obtiene sus propios recursos y garantías. Con un esquema basado en clase, todas las llamadas telefónicas obtienen los recursos reservados para la clase telefonía. Los paquetes de clase de navegación web o de otras clases no pueden tomar estos recursos, pero ninguna llamada telefónica puede obtener ningún recurso privado reservado sólo para ella.

Reenvío expedito

Cada operador debe elegir las clases de servicios, pero ya que los paquetes con frecuencia se reenvían entre redes operadas por diferentes operadores, la IETF ha definido algunas clases de servicios independientes de la red. La clase más simple es el **reenvío expedito** y se describe en el RFC 3246.

La idea detrás del reenvío expedito es muy simple. Hay dos clases de servicios disponibles: regular y expedito. Se espera que la mayor parte del tráfico sea regular, pero una pequeña fracción de los paquetes son expeditos. Éstos deben tener la capacidad de transitar por la red como si no hubiera otros paquetes. De esta forma ellos tendrán un servicio con pocas pérdidas, bajo retardo y poca variación del retardo; justo lo que *VoIP* necesita. En la Ilustración 36 se muestra una representación simbólica de este sistema de “dos tubos”. Observe que todavía hay sólo una línea física. Las dos tuberías lógicas que se muestran en la figura representan una forma de reservar ancho de banda para las distintas clases de servicio, no una segunda línea física.

A continuación veremos una forma de implementar esta estrategia. Los paquetes se clasifican como expeditos o regulares y se marcan según corresponda. Este paso se puede realizar en el host emisor o en el enrutador de ingreso (el primero). La ventaja de realizar la clasificación en el host emisor es que hay más información disponible acerca de los paquetes que pertenecen a cada flujo. Esta tarea se puede llevar a cabo mediante software de red o incluso a través del sistema operativo, para no tener que cambiar las aplicaciones existentes. Por ejemplo, cada vez es más común que los hosts marquen los paquetes VoIP para un servicio expedito. Si los paquetes pasan a través de una red corporativa o un ISP que soporte el servicio expedito, recibirán un trato especial. Si la red no soporta el servicio expedito, no habrá ningún problema.

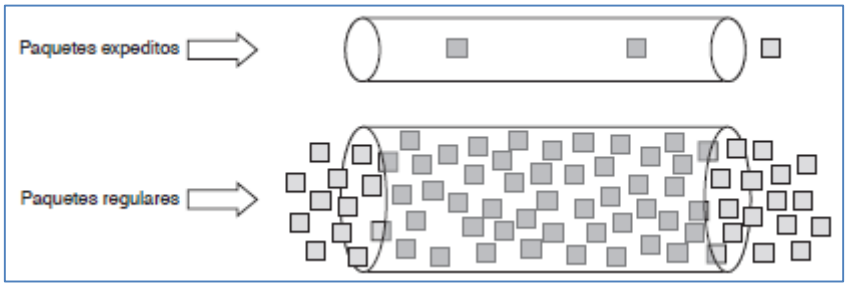


Ilustración 36 - Los paquetes expeditos experimentan una red sin tráfico.

Desde luego que si el host se encarga de marcar los paquetes, es probable que el enrutador de ingreso supervise el tráfico para asegurarse que los clientes no envíen más tráfico expedito del que pagaron. Dentro de la red, los enrutadores pueden tener dos colas de salida para cada línea de salida, una para los paquetes expeditos y otra para los paquetes regulares. Al llegar un paquete se pone en la cola correspondiente. La cola

expedita tiene prioridad sobre la regular; por ejemplo, mediante el uso de un programador de prioridades. De esta forma, los paquetes expeditos ven una red sin carga, aun cuando, de hecho, haya una carga pesada de tráfico regular.

Reenvío asegurado

Hay un esquema un poco más elaborado para administrar las clases de servicios, el cual se conoce como **reenvío asegurado**. Se describe en el RFC 2597 y especifica que debe haber cuatro clases de prioridades, cada una con sus propios recursos. Las primeras tres clases se deben llamar oro, plata y bronce. Además, define tres probabilidades de descarte para los paquetes que están en congestión: baja, media y alta. En conjunto, estos dos factores definen 12 clases de servicios.

La Ilustración 37 muestra una forma en que se pueden procesar los paquetes bajo el esquema de reenvío asegurado. El primer paso es clasificar los paquetes en una de las cuatro clases de prioridades. Como en el reenvío expedito, este paso se podría realizar en el host emisor (como se muestra en la figura) o en el enrutador de ingreso; el operador puede limitar la tasa de paquetes de prioridad más alta como parte del servicio ofrecido.

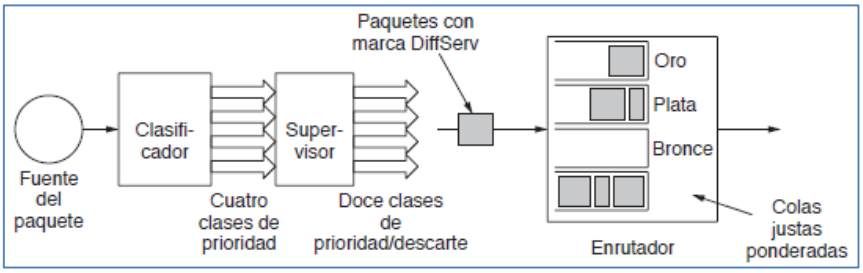


Ilustración 37 - Una posible implementación del reenvío asegurado.

El siguiente paso es determinar la clase de descarte para cada paquete. Para ello es necesario pasar los paquetes de cada clase de prioridad por un supervisor de tráfico como una cubeta con tokens. El supervisor deja pasar todo el tráfico, pero identifica los paquetes que caben dentro de pequeñas ráfagas con un nivel de descarte bajo, a los paquetes que exceden las pequeñas ráfagas con un nivel de descarte medio y a los paquetes que exceden las ráfagas grandes con un nivel de descarte alto. Entonces, la combinación de las clases de prioridad y de descarte se codifica en cada paquete.

Por último, los enrutadores en la red procesan los paquetes mediante un programador de paquetes que distingue las diversas clases. Una elección común es usar el encolamiento justo ponderado para las cuatro clases de prioridad, en donde las clases superiores reciben pesos más altos. De esta forma, las clases superiores obtendrán la mayor parte del ancho de banda pero no se privará a las clases inferiores de todo el ancho de banda. Por ejemplo, si los pesos se duplican de una clase a la siguiente clase más alta, ésta tendrá el doble de ancho de banda. Dentro de una clase de prioridad, se puede dar preferencia a los paquetes con una clase de descarte más alta para descartarlos mediante la ejecución de un algoritmo tal como RED (Detección Temprana Aleatoria). RED empezará a descartar paquetes a medida que se acumule la congestión, pero antes de que el enrutador se quede sin espacio en el búfer. En esta etapa aún hay espacio de búfer para aceptar los paquetes con bajo nivel de descarte mientras se descartan los paquetes con alto nivel de descarte.

Interconexión de redes

Hasta ahora hemos supuesto de manera implícita que hay una sola red homogénea, en donde cada máquina usa el mismo protocolo en cada capa. Por desgracia, este supuesto es demasiado optimista. Existen muchas redes distintas, entre ellas PAN, LAN, MAN y WAN. Ya describimos Ethernet, Internet por cable, 802.11 y muchas más. Hay numerosos protocolos con un uso muy difundido a través de estas redes, en cada capa. En las siguientes secciones estudiaremos con cuidado los aspectos que surgen cuando se conectan dos o más redes para formar una **interred** o simplemente una **internet**.

Sería más simple unir redes si todos usaran una sola tecnología de red; a menudo se da el caso de que hay un tipo dominante de red, como Ethernet. Si siempre habrá una variedad de redes, sería más simple que no necesitáramos interconectarlas. Esto también es muy poco probable.

Internet es el ejemplo primordial de esta interconexión (vamos a escribir Internet con una "I" mayúscula para diferenciarla de otras interred o redes interconectadas). El propósito de unir todas estas redes es permitir que los usuarios en cualquiera de ellas se comuniquen con los usuarios de las otras redes. Cuando usted paga a un ISP por el servicio de Internet, el costo dependerá del ancho de banda de su línea, pero lo que en realidad está pagando es la capacidad de intercambiar paquetes con cualquier otro host que también esté conectado a Internet.

Como por lo general las redes difieren en formas importantes, no siempre es tan sencillo enviar paquetes de una red a otra. Debemos lidiar con problemas de heterogeneidad y también con problemas de escala a medida que la interred resultante aumenta su tamaño en forma considerable. Empezaremos por analizar la forma en que pueden diferir las redes, para ver a qué nos enfrentamos. Después veremos el método tan exitoso utilizado por **IP** (Protocolo de Internet), el protocolo de la capa de red de Internet, incluyendo las técnicas de tunelización a través de las redes, el enrutamiento en las interredes y la fragmentación de paquetes.

Cómo difieren las redes

Las redes pueden diferir de muchas maneras. Algunas de las diferencias, como técnicas de modulación o formatos de tramas diferentes, se encuentran en la capa física y en la de enlace de datos. No trataremos esas diferencias aquí. En su lugar, en la Ilustración 38 listamos algunas diferencias que pueden ocurrir en la capa de red. La conciliación de estas diferencias es lo que hace más difícil la interconexión de redes que la operación con una sola red.

Aspecto	Algunas posibilidades
Servicio ofrecido.	Sin conexión *vs.* orientado a conexión.
Direccionamiento.	Distintos tamaños, plano o jerárquico.
Difusión.	Presente o ausente (también multidifusión).
Tamaño de paquete.	Cada red tiene su propio valor máximo.
Ordenamiento.	Entrega ordenada y desordenada.
Calidad del servicio.	Presente o ausente; muchos tipos distintos.
Confiabilidad.	Distintos niveles de pérdida.
Seguridad.	Reglas de privacidad, cifrado, etcétera.
Parámetros.	Distintos tiempos de expiración, especificaciones de flujo, etcétera..
Contabilidad.	Por tiempo de conexión, paquete, byte o ninguna.

Ilustración 38 - Algunas de las diversas formas en que pueden diferir las redes.

Cuando los paquetes enviados por una fuente en una red deben transitar a través de una o más redes foráneas antes de llegar a la red de destino, pueden ocurrir muchos problemas en las interfaces entre las redes. Para empezar, la fuente necesita dirigirse al destino. ¿Qué hacemos si la fuente está en una red Ethernet y el destino en una red WiMAX? Suponiendo que sea posible especificar un destino WiMAX desde una red Ethernet, los paquetes pasarían de una red sin conexión a una orientada a conexión. Para ello tal vez sea necesario establecer una conexión improvisada, lo cual introduce un retardo y mucha sobrecarga si la conexión no se utiliza para muchos paquetes más.

También debemos tener en cuenta muchas diferencias específicas. ¿Cómo enviamos un paquete mediante multidifusión a un grupo con ciertos miembros en una red que no soporta multidifusión? Los diferentes tamaños máximos de paquete usados por las diferentes redes también pueden ser una gran molestia. ¿Cómo se pasa un paquete de 8000 bytes a través de una red cuyo tamaño máximo de paquete es de 1500 bytes? Si los paquetes en una red orientada a conexión transitan en una red sin conexión, pueden llegar en un orden distinto al que se enviaron. Esto es algo que el emisor probablemente no esperaba, y podría ser también una sorpresa para el receptor.

Estos tipos de diferencias se pueden enmendar, con cierto esfuerzo. Por ejemplo, una puerta de enlace que une dos redes podría generar paquetes separados para cada destino en vez de un mejor soporte para la multidifusión. Un paquete extenso se podría dividir, enviar en piezas y unir de vuelta. Los receptores podrían colocar los paquetes en un búfer y entregarlos en orden.

Las redes también pueden diferir en sentidos más importantes que son más difíciles de conciliar. El ejemplo más claro es la calidad del servicio. Si una red tiene una QoS sólida y la otra ofrece un servicio del mejor esfuerzo, será imposible hacer garantías de ancho de banda y retardo para el tráfico en tiempo real de un extremo al otro. De hecho, es probable que sólo se puedan hacer mientras la red del mejor esfuerzo opere con poco uso, o cuando se utilice en raras ocasiones, lo cual no es muy probable que sea el objetivo de la mayoría de los ISP. Los mecanismos de seguridad son problemáticos, pero por lo menos el cifrado para la confiabilidad e integridad de los datos se puede poner encima de las redes que no lo incluyen de antemano.

Cómo se pueden conectar las redes

Existen dos opciones básicas para conectar distintas redes: podemos construir dispositivos que traduzcan o conviertan los paquetes de cada tipo de red en paquetes para otra red o podemos tratar de resolver el problema al agregar una capa de **indirección** (referencia indirecta) y construir una capa común encima de las distintas redes. En cualquier caso, los dispositivos se colocan en los límites entre las redes.

Algunos autores abogaron por una capa común para ocultar las diferencias de las redes existentes. Esta metodología ha tenido un gran éxito, y la capa que propusieron se separó eventualmente en los protocolos TCP e IP. IP proporciona un formato de paquete universal que todos los enrutadores reconocen y que se puede pasar casi por cualquier red. IP ha extendido su alcance de las redes de computadoras para apoderarse de la red telefónica. También opera en redes de sensores y en otros dispositivos pequeños que alguna vez se consideraron con recursos demasiado restringidos como para soportarlo.

Hemos analizado varios dispositivos diferentes que conectan redes, incluyendo repetidores, hubs, switches, puentes, enrutadores y puertas de enlace. Los repetidores y hubs sólo desplazan bits de un cable a otro. En su mayoría son dispositivos analógicos y no comprenden nada sobre los protocolos de las capas superiores. Los puentes y switches operan en la capa de enlace. Se pueden usar para construir redes, pero sólo con una pequeña traducción de protocolos en el proceso; por ejemplo, entre los switches Ethernet de 10, 100 y 1000 Mbps. Nuestro enfoque en esta sección es en los dispositivos de interconexión que operan en la capa de red, es decir, los enrutadores. Dejaremos las puertas de enlace, que son dispositivos de interconexión de las capas superiores, para después.

Primero exploraremos a un alto nivel la forma en que se puede usar la interconexión con una capa de red común para interconectar redes distintas. En la Ilustración 39(a) se muestra una interred compuesta por redes 802.11, MPLS y Ethernet. Suponga que la máquina fuente en la red 802.11 desea enviar un paquete a la máquina de destino en la red Ethernet. Como estas tecnologías son distintas y además están separadas por otro tipo de red (MPLS), se requiere un procesamiento adicional en los límites entre las redes.

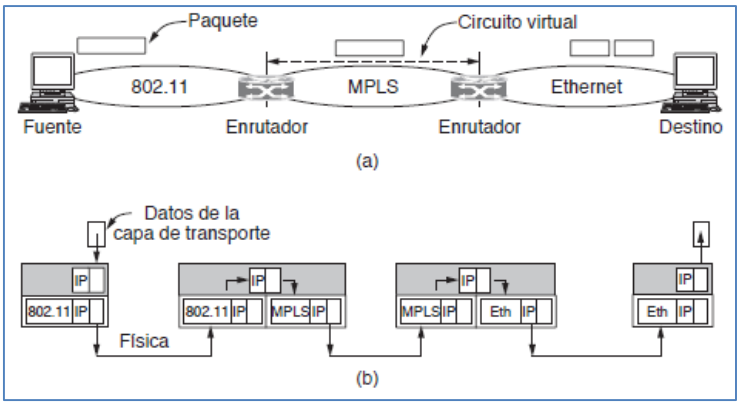


Ilustración 39 - (a) Un paquete que cruza distintas redes. (b) Procesamiento de protocolos de las capas de red y de enlace.

En general, como diferentes redes pueden tener distintas formas de direccionamiento, el paquete transporta una dirección de capa de red que puede identificar cualquier host a través de las tres redes. El primer límite al que llega el paquete es cuando cambia de una red 802.11 a una red MPLS. 802.11 proporciona un servicio sin conexión, pero MPLS provee un servicio orientado a conexión. Esto significa que se debe establecer un circuito virtual para cruzar esa red. Una vez que el paquete viaje por el circuito virtual, llegará a la red Ethernet. En este límite tal vez el paquete sea demasiado grande como para transportarlo, ya que 802.11 puede trabajar con tramas más grandes que Ethernet. Para manejar este problema, el paquete se divide en fragmentos y cada fragmento se envía por separado. Cuando los fragmentos llegan a su destino, se vuelven a ensamblar. Entonces es cuando el paquete ha completado su viaje.

En la Ilustración 39(b) se muestra el procesamiento de los protocolos para este viaje. La fuente acepta los datos de la capa de transporte y genera un paquete con el encabezado de la capa de red común, que en este ejemplo es IP. El encabezado de red contiene la dirección de destino final, que se utiliza para determinar que se debe enviar el paquete a través del primer enrutador. Así, el paquete se encapsula en una trama 802.11, cuyo destino es el primer enrutador, y se transmite. En el enrutador, el paquete se extrae del campo de datos de la trama y se descarta el encabezado de la trama 802.11. Ahora el enrutador examina la dirección IP en el paquete y busca esta dirección en su tabla de enrutamiento. Con base en esta dirección, decide enviar a continuación el paquete al segundo enrutador. Para esta parte de la ruta, es necesario establecer un circuito virtual MPLS hacia el segundo enrutador y encapsular el paquete con encabezados MPLS que viajen por este circuito. En el extremo lejano, se descarta el encabezado MPLS y de nuevo se consulta la dirección de red para buscar el siguiente salto en la capa de red, el destino en sí. Como el paquete es demasiado largo para enviarlo a través de Ethernet, se divide en dos partes. Cada una se coloca en el campo de datos de una trama Ethernet y se envía a la dirección Ethernet del destino. Ya en el destino, se elimina el encabezado Ethernet de cada una de las tramas y se vuelve a ensamblar el contenido. Finalmente el paquete ha llegado a su destino.

Cabe mencionar que hay una diferencia esencial entre el caso de enrutamiento y el caso de conmutación (o puenteo). Con un enrutador, el paquete se extrae de la trama y la dirección de red en el paquete se utiliza para decidir a dónde enviarlo. Con un switch (o puente), toda la trama se transporta con base en su dirección MAC. Los switches no tienen que entender el protocolo de capa de red que se utiliza para conmutar los paquetes. Los enrutadores sí tienen que hacerlo.

Por desgracia, la interconexión de redes no es tan fácil como suena. De hecho, cuando se introdujeron los puentes, la intención era que unieran distintos tipos de redes, o por lo menos distintos tipos de LAN. Para ello debían traducir tramas de una LAN en tramas de otra LAN. Sin embargo, esto no funcionó bien debido a la misma razón por la que es difícil la interconexión de redes: las diferencias en las características de las LAN (como los distintos tamaños máximos de los paquetes, y las LAN con y sin clases de prioridad) son difíciles de enmascarar. En la actualidad, el uso más común de los puentes es para conectar el mismo tipo de red en la capa de enlace, y los enrutadores conectan distintas redes en la capa de red.

La interconexión de redes ha tenido mucho éxito en la creación de redes extensas, pero sólo funciona cuando hay una capa de red común. De hecho, con el tiempo han surgido varios protocolos de red. Es difícil lograr que todos estén de acuerdo en cuanto a un solo formato cuando las empresas perciben como ventaja comercial el tener un formato propietario que puedan controlar. Algunos ejemplos además de IP, que ahora es el protocolo de red casi universal, son: IPX, SNA y AppleTalk. Ninguno de estos protocolos se utiliza mucho en la actualidad, pero siempre habrá otros protocolos. Probablemente ahora el ejemplo más relevante sea el de IPv4 e IPv6. Aunque ambas son versiones de IP, no son compatibles (o no hubiera sido necesario crear IPv6).

Un enrutador que puede manejar múltiples protocolos de red se denomina **enrutador multiprotocolo**. Éste debe traducir los protocolos o dejar una conexión para una capa de protocolo superior. Ninguna de las dos opciones es totalmente satisfactoria. Para una conexión en una capa superior (por decir, mediante el uso de TCP) se requiere que todas las redes implementen TCP (lo cual tal vez no sea posible). En consecuencia, se limita el uso en las redes a las aplicaciones que usan TCP (lo cual no incluye a muchas aplicaciones en tiempo real).

La alternativa es traducir los paquetes entre las redes. Sin embargo, a menos que los formatos de los paquetes sean muy similares y tengan los mismos campos de información, tales conversiones siempre serán incompletas

y con frecuencia estarán destinadas al fracaso. Por ejemplo, las direcciones IPv6 son de 128 bits de longitud. No cabrán en un campo de dirección IPv4 de 32 bits, sin importar qué tanto se esfuerce el enrutador. El hecho de lograr que IPv4 e IPv6 funcionen en la misma red, ha demostrado ser un gran obstáculo para la implementación de IPv6 (para ser justos, también ha sido muy difícil hacer que los clientes entiendan por qué les convendría usar IPv6 en primer lugar). Se pueden esperar mayores problemas al traducir entre protocolos básicamente distintos, como los protocolos de red sin conexión y los orientados a conexión. Dadas estas dificultades, la conversión sólo se intenta raras veces. Sin duda, la razón por la que IP ha funcionado tan bien se debe a que sirve como un tipo de mínimo común denominador. Requiere muy poco de las redes en las que opera, pero ofrece como resultado sólo un servicio del mejor esfuerzo.

Tunelización

Es en extremo difícil manejar el caso general de hacer que dos redes distintas se interconecten. Sin embargo, hay un caso especial que se puede manejar, incluso para distintos protocolos de red. Este caso es cuando el host de origen y el de destino están en el mismo tipo de red, pero hay una red diferente en medio. Como ejemplo, piense en un banco internacional con una red IPv6 en París, una en Londres y una conectividad entre las oficinas a través de la Internet IPv4. Esta situación se muestra en la Ilustración 40.

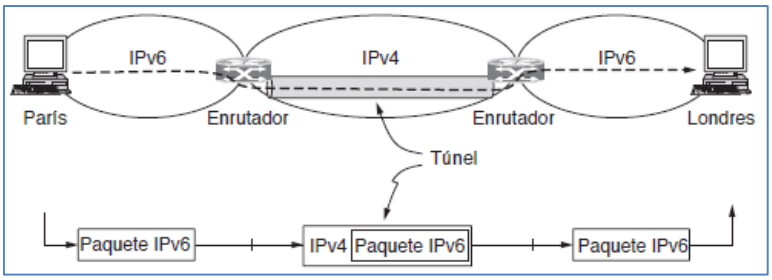


Ilustración 40 - Tunelización de un paquete de París a Londres.

La solución a este problema es una técnica llamada **tunelización** (*tunneling*). Para enviar un paquete IP a un host en la oficina de Londres, un host en la oficina de París construye el paquete que contiene una dirección IPv6 en Londres y la envía al enrutador multiprotocolo que conecta la red IPv6 de París con la Internet IPv4. Cuando este enrutador recibe el paquete IPv6, lo encapsula con un encabezado IPv4 dirigido al lado IPv4 del enrutador multiprotocolo que se conecta con la red IPv6 de Londres. Es decir, el enrutador coloca un paquete (IPv6) dentro de un paquete (IPv4). Cuando llega este paquete envuelto, el enrutador de Londres extrae el paquete IPv6 original y lo envía hacia el host de destino.

Podemos visualizar la ruta hacia la Internet IPv4 como un gran túnel que se extiende de un enrutador multiprotocolo al otro. El paquete IPv6 simplemente viaja de un extremo del túnel al otro, bien acomodado en una caja bonita. No tiene que preocuparse por lidiar con IPv4 para nada. Tampoco tienen que hacerlo los hosts en París o en Londres. Sólo los enrutadores multiprotocolo tienen que entender los paquetes IPv4 e IPv6. De hecho, el recorrido completo desde un enrutador multiprotocolo al otro es como un salto a través de un solo enlace.

La tunelización se utiliza mucho para conectar hosts y redes aisladas mediante el uso de otras redes. La red que resulta se denomina **red superpuesta** (*overlay*), ya que realmente está superpuesta sobre la red base. Implementar un protocolo de red con una nueva característica es un motivo común, como se indica en nuestro ejemplo de “IPv6 sobre IPv4”. La desventaja de la tunelización es que no se puede llegar a ninguno de los hosts en la red que se tuneliza debido a que los paquetes no pueden escapar a mitad del túnel. Sin embargo, esta limitación de los túneles se convierte en una ventaja gracias a las redes **VPN** (Redes Privadas Virtuales, del inglés *Virtual Private Network*). Una VPN es simplemente una red superpuesta que se utiliza para proporcionar una medida de seguridad.

Enrutamiento entre redes

El enrutamiento a través de una interred presenta el mismo problema que el enrutamiento en una sola red, pero con algunas complicaciones adicionales. Para empezar, las redes pueden usar internamente distintos algoritmos de enrutamiento. Por ejemplo, una red podría usar un enrutamiento por estado del enlace y otra

un enrutamiento por vector de distancia. Como los algoritmos de estado del enlace necesitan conocer la topología pero los algoritmos de vector de distancia no, tan sólo esta diferencia dificultaría mucho el proceso de encontrar las rutas más cortas a través de la internet.

Las redes manejadas por distintos operadores conducen a problemas más grandes. En primer lugar, tal vez los operadores tengan distintas ideas sobre lo que sería una buena ruta a través de la red. Posiblemente un operador quiera la ruta con el menor retardo, mientras que otro prefiera la ruta menos costosa. Esto obligará a los operadores a usar distintas cantidades para establecer los costos de la ruta más corta (por ejemplo, milisegundos de retardo en comparación con el costo monetario). Como no se compararán los pesos entre las redes, las rutas más cortas en la interred no estarán bien definidas.

O peor aún, tal vez un operador no quiera que otro conozca siquiera los detalles de las rutas en su red, quizá debido a que los pesos y las rutas puedan reflejar información delicada (como el costo monetario) que representa una ventaja comercial competitiva.

Por último, la interred puede ser mucho más grande que cualquiera de las redes que la conforman. Por lo tanto, puede requerir algoritmos de enrutamiento que escalen bien mediante el uso de una jerarquía, incluso si ninguna de las redes individuales necesita usar una jerarquía.

Todas estas consideraciones conducen a un algoritmo de enrutamiento de dos niveles. Dentro de cada red, se utiliza un **protocolo intradominio** o de **puerta de enlace interior** para el enrutamiento ("*puerta de enlace*" es un término antiguo para "*enrutador*"). Podría ser un protocolo de estado del enlace del tipo que ya hemos descrito. A través de las redes que conforman la interred se utiliza un **protocolo interdominio** o de **puerta de enlace exterior**. Todas las redes pueden usar distintos protocolos intradominio, pero deben usar el mismo protocolo interdominio. En Internet, el protocolo de enrutamiento interdominio se denomina **BGP** (Protocolo de Puerta de Enlace de Frontera, del inglés *Border Gateway Protocol*). Lo describiremos en la siguiente sección.

Hay un término importante más que debemos introducir. Como cada red se opera de manera independiente a las demás, se conoce comúnmente como un **AS** (Sistema Autónomo, del inglés *Autonomous System*). Una red de un ISP es un buen modelo mental de un AS. De hecho, una red de ISP puede estar compuesta por más de un AS en caso de ser administrada, o, si ha sido adquirida, por múltiples redes. Pero en general, la diferencia no es importante.

Por lo común, los dos niveles no son estrictamente jerárquicos, pues se podrían generar rutas muy subóptimas si una red internacional extensa y una red regional pequeña se abstrajeran para formar una sola red. Sin embargo, en realidad se expone muy poca información sobre las rutas dentro de la red para encontrar rutas a través de la interred. Esto ayuda a lidiar con todas las complicaciones. Mejora el escalamiento y permite a los operadores elegir libremente las rutas dentro de sus propias redes, mediante el uso de un protocolo de su elección. Además, tampoco se tienen que comparar los pesos a través de las redes ni exponer la información delicada fuera de ellas.

No obstante, hemos dicho muy poco hasta ahora sobre la forma en que se determinan las rutas a través de las redes de la interred. En Internet, un gran factor determinante consiste en los arreglos comerciales entre los ISP. Cada ISP puede cobrar o recibir dinero de los otros ISP por transportar tráfico. Otro factor es que, si el enrutamiento de la interred requiere cruzar límites internacionales, de repente pueden entrar en juego varias leyes, como las estrictas leyes de privacidad de Suecia con relación a la exportación de datos personales sobre los ciudadanos suecos. Todos estos factores no técnicos están envueltos en el concepto de una política de enrutamiento que gobierna la forma en que las redes autónomas seleccionan los enrutadores que van a usar. Volveremos a las políticas de enrutamiento cuando hablemos sobre el BGP.

Fragmentación de paquetes

Cada red o enlace impone un tamaño máximo a sus paquetes. Estos límites tienen varias razones, entre ellas:

1. El hardware (por ejemplo, el tamaño de una trama Ethernet).
2. El sistema operativo (por ejemplo, todos los búferes son de 512 bytes).
3. Los protocolos (por ejemplo, la cantidad de bits en el campo de longitud de paquete).
4. El cumplimiento de algún estándar (inter)nacional.

5. El deseo de reducir hasta cierto nivel las retransmisiones inducidas por errores.
6. El deseo de evitar que un paquete ocupe el canal demasiado tiempo.

El resultado de todos estos factores es que los diseñadores de redes no tienen la libertad de elegir cualquier tamaño máximo de paquetes que deseen. Las cargas útiles máximas para algunas tecnologías comunes son de 1500 bytes para Ethernet y 2272 para 802.11. El protocolo IP es más generoso, ya que permite paquetes de hasta 65515 bytes.

Por lo general, los hosts prefieren transmitir paquetes grandes, ya que esto reduce las sobrecargas de paquetes, como el ancho de banda desperdiciado en los bytes de encabezado. Surge un problema obvio de interconexión de redes cuando un paquete grande quiere viajar a través de una red cuyo tamaño máximo de paquete es muy pequeño. Esta molestia ha sido una cuestión persistente; y las soluciones han evolucionado junto con toda la experiencia que se ha obtenido gracias a Internet.

Una solución es asegurar que no ocurra el problema en primer lugar. Sin embargo, es más fácil decir esto que hacerlo. Por lo general, una fuente no conoce la ruta que tomará un paquete a través de la red hacia un destino, por lo que en definitiva no sabe qué tan pequeños deben ser los paquetes para llegar ahí. A este tamaño de paquete se le conoce como **MTU de la ruta** (Unidad de Transmisión Máxima de la ruta, del ingés *Path Maximum Transmission Unit*). Incluso si la fuente conociera el MTU de la ruta, los paquetes se enrutan de manera independiente en una red sin conexión como Internet. Este enrutamiento significa que las rutas pueden cambiar de repente, lo que a su vez puede cambiar de manera inesperada el MTU de la ruta.

La solución alternativa al problema es permitir que los enrutadores dividan los paquetes en fragmentos y envíen cada uno como un paquete de red separado. Sin embargo, es mucho más fácil convertir un objeto grande en pequeños fragmentos que el proceso inverso. Las redes de conmutación de paquetes también tienen problemas al unir nuevamente los fragmentos.

Existen dos estrategias opuestas para recombinar los fragmentos de vuelta en el paquete original. La primera es hacer que la fragmentación producida por una red de “pequeños paquetes” sea transparente para cualquier red subsecuente por la que deba pasar el paquete en su camino hacia el destino final. Esta opción se muestra en la Ilustración 41(a). En este método, cuando un paquete de tamaño excesivo llega a *G1*, el enrutador lo divide en fragmentos. Cada fragmento es dirigido al mismo enrutador de salida, *G2*, en donde se recombinan las piezas. De esta manera se ha hecho transparente el paso a través de la red de paquete pequeño. Las redes subsecuentes ni siquiera se enteran de que ha ocurrido una fragmentación.

La fragmentación transparente es sencilla, pero tiene algunos problemas. Por una parte, el enrutador de salida debe saber cuándo ha recibido todas las piezas, por lo que debe incluirse un campo de conteo o un bit de “fin de paquete”. Además, como todos los paquetes deben salir por el mismo enrutador para volver a ensamblarlos, las rutas están restringidas. Al no permitir que algunos fragmentos sigan una ruta al destino final, y otros fragmentos una ruta distinta, puede bajar un poco el desempeño. Lo más importante es la cantidad de trabajo que el enrutador tenga que llevar a cabo. Tal vez necesite colocar los fragmentos en un búfer a medida que vayan llegando, para después decidir cuándo debe descartarlos si no llegan todos los fragmentos. Parte de este trabajo también podría ser un desperdicio, ya que el paquete puede pasar a través de una serie de pequeñas redes de paquetes, y tal vez haya que fragmentarlo y recombinarlo repetidas veces.

La otra estrategia de fragmentación es abstenerse de recombinar los fragmentos en los enrutadores intermedios. Una vez que se ha fragmentado un paquete, cada fragmento se trata como si fuera un paquete original. Los enrutadores pasan los fragmentos, como se muestra en la Ilustración 41(b); la recombinación ocurre sólo en el host de destino.

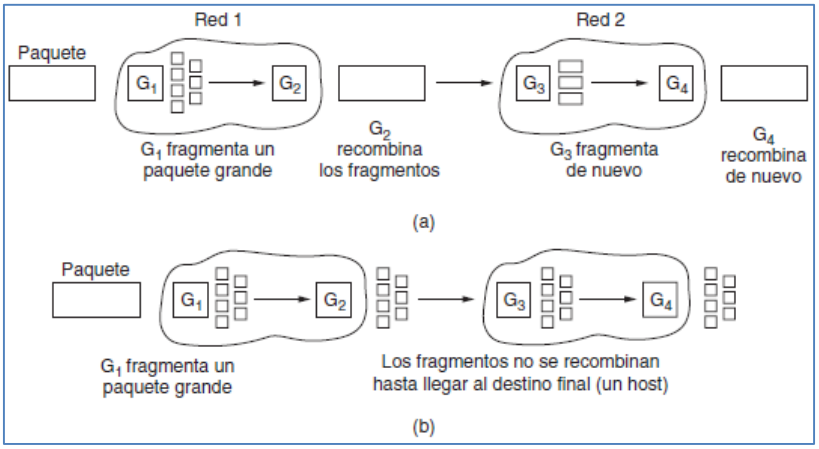


Ilustración 41 - (a) Fragmentación transparente. (b) Fragmentación no transparente.

La principal ventaja de la fragmentación no transparente es que los enrutadores tienen que trabajar menos. IP funciona de esta manera. Un diseño completo requiere que los fragmentos se enumeren de tal forma que se pueda reconstruir el flujo de datos original. El diseño utilizado por IP es proporcionar a cada fragmento un número de paquete (que todos los paquetes transportan), un desplazamiento de bytes absoluto dentro del paquete, y una bandera que indica si es el final del paquete. En la Ilustración 42 se muestra un ejemplo. Aunque es simple, este diseño tiene algunas propiedades atractivas. Los fragmentos se pueden colocar en un búfer en el destino, en el lugar apropiado para recombinarlos, aun cuando lleguen desordenados. Los fragmentos también se pueden fragmentar si pasan a través de una red con una MTU aún más pequeña. Esto se muestra en la Ilustración 42(c). Las retransmisiones del paquete (si no se recibieron todos los fragmentos) se pueden fragmentar en distintas piezas. Por último, los fragmentos pueden ser de un tamaño arbitrario, incluso hasta de un solo byte más el encabezado del paquete. En todos los casos, el destino simplemente usa el número de paquete y el desplazamiento del fragmento para colocar los datos en la posición correcta, y la bandera de fin de paquete para determinar cuándo tiene el paquete completo.

Por desgracia, este diseño aún tiene problemas. La sobrecarga puede ser mayor que con la fragmentación transparente, ya que los encabezados de los fragmentos ahora se transportan a través de algunos enlaces en donde tal vez no sean necesarios. Pero el verdadero problema es la existencia de los fragmentos en primera instancia. Según algunos autores, la fragmentación es perjudicial para el desempeño, ya que al igual que las sobrecargas de los encabezados, un paquete completo se extravía si cualquiera de sus fragmentos se pierde, y porque la fragmentación representa una carga para los hosts más grande de lo que se tenía pensado originalmente.

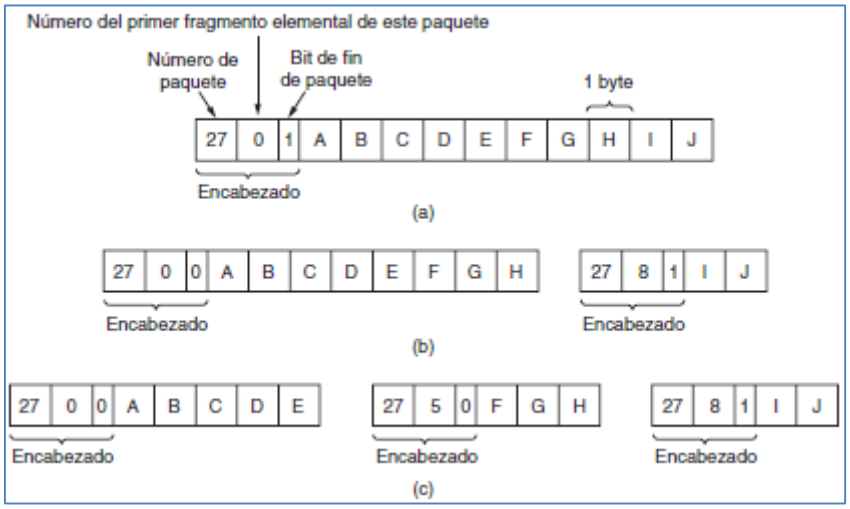


Ilustración 42 - La fragmentación cuando el tamaño de datos elemental es de 1 byte. (a) El paquete original que contiene 10 bytes de datos. (b) Los fragmentos después de pasar por una red con un tamaño máximo de paquete de 8 bytes de carga útil más encabezado. (c) Fragmentos después de pasar a través de una puerta de enlace de tamaño 5.

Esto nos conduce de vuelta a la solución original de deshacernos de la fragmentación en la red, la estrategia que se utiliza en la Internet moderna. Al proceso se le conoce como **descubrimiento de MTU de la ruta** y trabaja de la siguiente manera. Cada paquete IP se envía con sus bits de encabezado establecidos para indicar que no se puede realizar ningún tipo de fragmentación. Si un enrutador recibe un paquete demasiado grande, genera un paquete de error, lo devuelve a la fuente y descarta el paquete. Esto se muestra en la Ilustración 43. Cuando el origen recibe el paquete de error, usa la información interna para volver a fragmentar el paquete en piezas que sean lo bastante pequeñas como para que el enrutador las pueda manejar. Si un enrutador más adelante en la ruta tiene una MTU aún más pequeña, se repite el proceso.

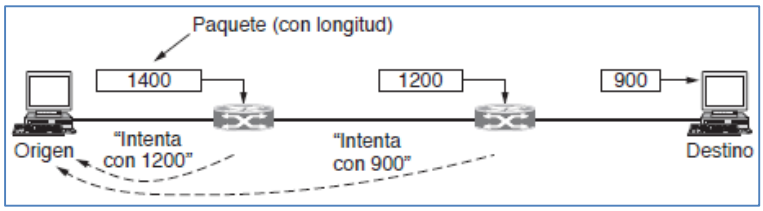


Ilustración 43 - Descubrimiento de MTU de la ruta.

La ventaja del descubrimiento de MTU de la ruta es que ahora la fuente sabe la longitud del paquete que puede enviar. Si las rutas y la MTU de la ruta cambian, se activarán nuevos paquetes de error y la fuente se adaptará a la nueva ruta. Sin embargo, la fragmentación sigue siendo necesaria entre la fuente y el destino, a menos que las capas superiores descubran la MTU de la ruta y pasen la cantidad correcta de datos al protocolo IP. Por lo general, TCP e IP se implementan en conjunto (como "TCP/IP") para pasar este tipo de información. Aun si esto no se hace con otros protocolos, de todas formas la fragmentación se sacó de la red y se pasó a los hosts.

La desventaja del descubrimiento de MTU de la ruta es que puede haber retardos iniciales adicionales sólo por enviar un paquete. Tal vez sea necesario más de un retardo de ida y vuelta para sondear la ruta y encontrar la MTU antes de entregar datos al destino. Aquí surge la pregunta acerca de si hay mejores diseños. Quizá la respuesta sea "sí". Considere el diseño en el que cada enrutador simplemente trunca los paquetes que exceden su MTU. Esto aseguraría que el destino conozca la MTU lo más rápido posible (con base en la cantidad de datos entregados) y reciba algunos de los datos.

La capa de red de Internet

Ahora es momento de examinar a conciencia la capa de red de Internet. Pero antes de entrar en detalles, vale la pena dar un vistazo a los principios que guiaron su diseño en el pasado y que hicieron posible el éxito que tiene hoy en día. En la actualidad, con frecuencia parece que la gente los ha olvidado. Estos principios se enumeran y analizan en el RFC 1958. Ahora sintetizaremos los 10 principios que pueden considerarse los más importantes.

1. **Asegurarse de que funciona**. No termine el diseño o estándar hasta que múltiples prototipos se hayan comunicado entre sí de manera exitosa. Con mucha frecuencia, los diseñadores primero escriben un estándar de 1000 páginas, logran que se apruebe y después descubren que tiene demasiadas fallas y no funciona. Entonces escriben una versión 1.1 del estándar. Ésta no es la manera correcta de proceder.
2. **Mantener la simplicidad**. Cuando tenga duda, utilice la solución más simple. William de Occam formuló este principio (la navaja de Occam) en el siglo XIV. Dicho en términos modernos: combata las características. Si una característica no es absolutamente esencial, descártela, especialmente si el mismo efecto se puede alcanzar mediante la combinación de otras características.
3. **Elegir opciones claras**. Si hay varias maneras para realizar la misma tarea, elija sólo una. Tener dos o más formas de hacer lo mismo es buscarse problemas. Con frecuencia, los estándares tienen múltiples opciones, modos o parámetros debido a que personas poderosas insisten en que su método es el mejor. Los diseñadores deben resistir con fuerza esta tendencia. Simplemente diga no.
4. **Explotar la modularidad**. Este principio lleva directamente a la idea de tener pilas de protocolos, cuyas capas sean independientes entre sí. De esta forma, si las circunstancias requieren que un módulo o capa cambie, los otros no se verán afectados.

5. **Prevenir la heterogeneidad**. En cualquier red grande habrán diferentes tipos de hardware, facilidades de transmisión y aplicaciones. Para manejarlos, el diseño de la red debe ser simple, general y flexible.
6. **Evitar las opciones y parámetros estáticos**. Si los parámetros son inevitables (por ejemplo, el tamaño máximo del paquete), es mejor hacer que el emisor y el receptor negocien un valor que definir opciones fijas.
7. **Buscar un buen diseño no es necesario que sea perfecto**. Con frecuencia, los diseñadores tienen un buen diseño pero éste no puede manejar algún caso especial. En lugar de desbaratar el diseño, los diseñadores deberían optar por el buen diseño y dejar que esa parte la resuelvan las personas con requisitos extraños.
8. **Ser estricto cuando envíe y tolerante cuando reciba**. En otras palabras, sólo envíe paquetes que cumplan rigurosamente con los estándares, pero espere paquetes que tal vez no cumplan del todo y trate de lidiar con ellos.
9. **Pensar en la escalabilidad**. Si el sistema debe manejar de manera efectiva millones de hosts y miles de millones de usuarios, no se toleran bases de datos centralizadas de ningún tipo y la carga se debe dispersar de la manera más equitativa posible entre los recursos disponibles.
10. **Considerar el desempeño y el costo**. Si una red tiene un desempeño pobre o un costo exagerado, nadie la utilizará.

Dejemos a un lado los principios generales y comencemos a ver los detalles de la capa de red de Internet. En la capa de red, Internet puede verse como un conjunto de redes, o **Sistemas Autónomos** (AS) interconectados. No hay una estructura real, pero existen varias redes troncales (*backbones*) principales. Éstas se construyen a partir de líneas de alto ancho de banda y enrutadores rápidos. Las más grandes de estas redes troncales, a la que se conectan todos los demás para llegar al resto de Internet, se llaman **redes de Nivel 1**. Conectadas a las redes troncales hay ISP (Proveedores de Servicio de Internet) que proporcionan acceso a Internet para los hogares y negocios, centros de datos e instalaciones de colocación llenas de máquinas servidores y redes regionales (de nivel medio). Los centros de datos sirven gran parte del contenido que se envía a través de Internet. A las redes regionales están conectados más ISP, LAN de muchas universidades y empresas, y otras redes de punta. En la Ilustración 44 se presenta un dibujo de esta organización cuasijerárquica.

El pegamento que mantiene unida a Internet es el protocolo de capa de red, IP (Protocolo de Internet, del inglés Internet Protocol). A diferencia de la mayoría de los protocolos de capa de red anteriores, IP se diseñó desde el principio con la interconexión de redes en mente. Una buena manera de visualizar la capa de red es la siguiente: su trabajo es proporcionar un medio de mejor esfuerzo (es decir, sin garantía) para transportar paquetes de la fuente al destino, sin importar si estas máquinas están en la misma red o si hay otras redes entre ellas.

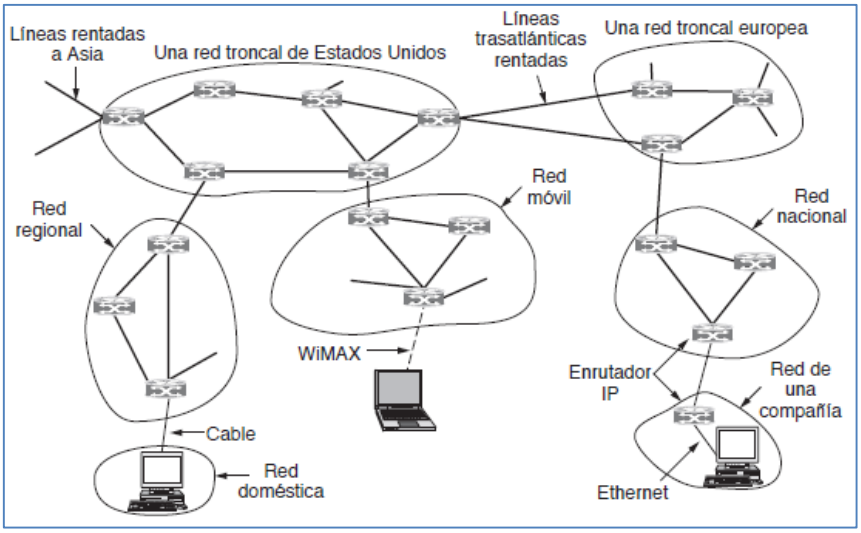


Ilustración 44 - Internet es una colección interconectada de muchas redes.

La comunicación en Internet funciona de la siguiente manera. La capa de transporte toma flujos de datos y los divide para poder enviarlos como paquetes IP. En teoría, los paquetes pueden ser de hasta 64 Kbytes cada uno, pero en la práctica por lo general no sobrepasan los 1500 bytes (ya que son colocados en una trama de Ethernet). Los enrutadores IP reenvían cada paquete a través de Internet, a lo largo de una ruta de un enrutador a otro, hasta llegar al destino. En el destino, la capa de red entrega los datos a la capa de transporte, que a su vez los entrega al proceso receptor. Cuando todas las piezas llegan finalmente a la máquina de destino, la capa de red las vuelve a ensamblar para formar el datagrama original. A continuación este datagrama se entrega a la capa de transporte.

En el ejemplo de la Ilustración 44, un paquete que se origina en la red doméstica tiene que atravesar cuatro redes y muchos enrutadores IP antes de llegar a la red de la compañía en la que se encuentra el host de destino. Esto es muy común en la práctica e incluso hay varias rutas más largas. También hay mucha conectividad redundante en Internet, con redes troncales e ISP conectados entre sí en varias ubicaciones. Esto significa que hay muchas rutas posibles entre dos hosts. Los protocolos de enrutamiento IP tienen la tarea de decidir qué rutas usar.

El protocolo IP versión 4

Un lugar adecuado para comenzar nuestro estudio de la capa de red de Internet es el formato de los datagramas de IP mismos. Un datagrama IPv4 consiste en dos partes: el encabezado y el cuerpo o carga útil. El encabezado tiene una parte fija de 20 bytes y una parte opcional de longitud variable. El formato del encabezado se muestra en la Ilustración 45. Los bits se transmiten en orden de izquierda a derecha y de arriba hacia abajo, comenzando por el bit de mayor orden del campo *Versión* (éste es un orden de bytes de red *big endian*. En las máquinas *little endian*, como las computadoras Intel x86, se requiere una conversión por software tanto para la transmisión como para la recepción). En retrospectiva, el formato *little endian* hubiera sido una mejor opción, pero al momento de diseñar IP nadie sabía que llegaría a dominar la computación.

El campo *Versión* lleva el registro de la versión del protocolo al que pertenece el datagrama. La versión 4 es la que domina Internet en la actualidad, y ahí es donde empezamos nuestro estudio. Al incluir la versión al inicio de cada datagrama, es posible tener una transición entre versiones a través de un largo periodo de tiempo. De hecho, IPv6 (la siguiente versión de IP) se definió hace más de una década y apenas se está empezando a implementar. Lo describiremos más adelante en esta sección. En un momento dado nos veremos obligados a usarlo cuando cada uno de los casi 2^{31} habitantes de China tenga una PC de escritorio, una computadora portátil y un teléfono IP. Como observación adicional en cuanto a la numeración, IPv5 fue un protocolo de flujo experimental en tiempo real que nunca fue muy popular.

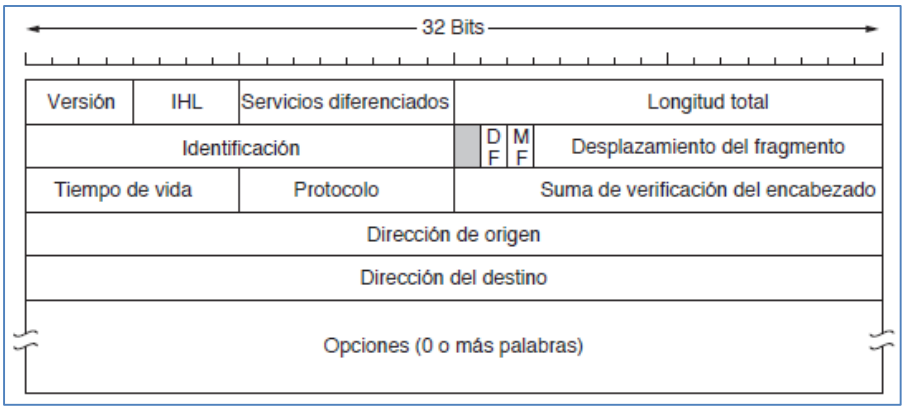


Ilustración 45 - El encabezado de IPv4 (Protocolo de Internet).

Dado que la longitud del encabezado no es constante, se incluye un campo en el encabezado (*IHL*) para indicar su longitud en palabras de 32 bits. El valor mínimo es de 5, cifra que se aplica cuando no hay opciones. El valor máximo de este campo de 4 bits es 15, lo que limita el encabezado a 60 bytes y por lo tanto, el campo *Opciones* a 40 bytes. Para algunas opciones, por ejemplo una que registre la ruta que ha seguido un paquete, 40 bytes es muy poco, lo que hace inútiles estas opciones.

El campo de *Servicio diferenciado* es uno de los pocos campos que ha cambiado su significado con el transcurso de los años. En un principio el nombre de este campo era *Tipo de servicio*. Su propósito era, y aún es, distinguir entre las diferentes clases de servicios. Son posibles varias combinaciones de confiabilidad y velocidad. Para voz digitalizada, la entrega rápida le gana a la entrega precisa. Para la transferencia de archivos, es más importante la transmisión libre de errores que la transmisión rápida. El campo *Tipo de servicio* contaba con 3 bits para indicar la prioridad y 3 bits para indicar si a un host le preocupaba más el retardo, la velocidad de transmisión real o la confiabilidad. Sin embargo, nadie sabía realmente qué hacer con estos bits en los enrutadores, por lo que se quedaron sin uso durante muchos años. Cuando se diseñaron los servicios diferenciados, la IETF tiró la toalla y reutilizó este campo. Ahora, los 6 bits superiores se utilizan para marcar el paquete con su clase de servicio; anteriormente en este capítulo describimos los servicios expedito y asegurado. Los 2 bits inferiores se utilizan para transportar información sobre la notificación de congestión; por ejemplo, si el paquete ha experimentado congestión o no. Asimismo, también describimos la notificación explícita de congestión como parte del control de la congestión.

El campo *Longitud total* incluye todo en el datagrama: tanto el encabezado como los datos. La longitud máxima es de 65535 bytes. En la actualidad este límite es tolerable, pero con las redes futuras tal vez se requieran datagramas más grandes.

El campo *Identificación* es necesario para que el host de destino determine a qué paquete pertenece un fragmento recién llegado. Todos los fragmentos de un paquete contienen el mismo valor de Identificación.

A continuación viene un bit sin uso, lo cual es sorprendente, ya que el espacio en el encabezado IP es muy escaso.

Después vienen dos campos de 1 bit relacionados con la fragmentación. *DF* significa no fragmentar (*Don't Fragment*), es una orden para que los enrutadores no fragmenten el paquete. En un principio estaban destinados para soportar a los hosts incapaces de reunir las piezas otra vez. Ahora se utiliza como parte del proceso para descubrir la MTU de la ruta. Al marcar el datagrama con el bit DF, el emisor sabe que llegará en una pieza o recibirá de vuelta un mensaje de error.

MF significa más fragmentos (*More Fragments*). Todos los fragmentos excepto el último tienen establecido este bit, que es necesario para saber cuándo han llegado todos los fragmentos de un datagrama.

El *Desplazamiento del fragmento* indica a qué parte del paquete actual pertenece este fragmento. Todos los fragmentos excepto el último del datagrama deben ser un múltiplo de 8 bytes, que es la unidad de fragmentos elemental. Dado que se proporcionan 13 bits, puede haber un máximo de 8192 fragmentos por datagrama, para soportar una longitud máxima de paquete de hasta el límite del campo *Longitud total*. En conjunto, los campos Identificación, MF y Desplazamiento del fragmento se utilizan para implementar la fragmentación como se describió anteriormente.

El campo *TtL* (*Tiempo de vida*) es un contador que se utiliza para limitar el tiempo de vida de un paquete. En un principio se suponía que iba a contar el tiempo en segundos, lo cual permitía un periodo de vida máximo de 255 s. Hay que decrementarlo en cada salto y se supone que se decrementa muchas veces cuando un paquete se pone en cola durante un largo tiempo en un enrutador. En la práctica, simplemente cuenta los saltos. Cuando el contador llega a cero, el paquete se descarta y se envía de regreso un paquete de aviso al host de origen. Esta característica evita que los paquetes anden vagando eternamente, algo que de otra manera podría ocurrir si se llegaran a corromper las tablas de enrutamiento.

Una vez que la capa de red ha ensamblado un paquete completo, necesita saber qué hacer con él. El campo *Protocolo* le indica a cuál proceso de transporte debe entregar el paquete. TCP es una posibilidad, pero también están UDP y otros más. La numeración de los protocolos es global en toda la Internet. Anteriormente los protocolos y otros números asignados se listaban en el RFC 1700, pero en la actualidad están contenidos en una base de datos en línea localizada en www.iana.org.

Puesto que el encabezado transporta información vital, como las direcciones, estima su propia suma de verificación por protección, la *Suma de verificación del encabezado*. El algoritmo suma todas las medias palabras de 16 bits del encabezado a medida que vayan llegando, mediante el uso de la aritmética de complemento a uno, y después obtiene el complemento a uno del resultado. Para los fines de este algoritmo,

se supone que la Suma de verificación del encabezado es cero al momento de la llegada. Dicha suma de verificación es útil para detectar errores mientras el paquete viaja por la red. Tenga en cuenta que se debe recalcular en cada salto, ya que por lo menos hay un campo que siempre cambia (el campo Tiempo de vida), aunque se pueden usar trucos para agilizar ese cálculo.

Los campos *Dirección de origen* y *Dirección de destino* indican la dirección IP de las interfaces de red de la fuente y del destino. En la siguiente sección estudiaremos las direcciones de Internet.

El campo *Opciones* se diseñó para proporcionar un recurso que permitiera que las versiones subsiguientes del protocolo incluyeran información que no estuviera presente en el diseño original, para que los experimentadores puedan probar ideas nuevas y evitar la asignación de bits de encabezado a la información que se necesite muy poco. Las opciones son de longitud variable. Cada una empieza con un código de 1 byte que identifica la opción. Algunas opciones van seguidas de un campo de longitud de la opción de 1 byte, y luego de uno o más bytes de datos. El campo Opciones se rellena para completar múltiplos de 4 bytes. En un principio se definieron las cinco opciones que se listan en la Ilustración 46.

Opción	Descripción
Seguridad.	Especifica qué tan secreto es el datagrama.
Enrutamiento estricto desde el origen.	Proporciona la ruta completa a seguir.
Enrutamiento libre desde el origen.	Proporciona una lista de enrutadores que no se deben omitir.
Registrar ruta.	Hace que cada enrutador adjunte su dirección IP.
Estampa de tiempo.	Hace que cada enrutador adjunte su dirección y su etiqueta de tiempo.

Ilustración 46 - Algunas de las opciones del protocolo IP.

La opción *Seguridad* indica qué tan secreta es la información. En teoría, un enrutador militar podría usar este campo para especificar que no se enruten paquetes a través de ciertos países que los militares consideren "*malos*". En la práctica todos los enrutadores lo ignoran, por lo que su única función real es la de ayudar a los espías a encontrar la información importante con mayor facilidad.

La opción de *Enrutamiento estricto desde el origen* proporciona la ruta completa desde el origen hasta el destino como una secuencia de direcciones IP. Se requiere que el datagrama siga esa ruta exacta. Esta opción se usa sobre todo cuando los administradores de sistemas necesitan enviar paquetes de emergencia cuando se corrompen las tablas de enrutamiento, o para hacer mediciones de sincronización.

La opción de *Enrutamiento libre desde el origen* requiere que el paquete pase por los enrutadores indicados en la lista, y en el orden especificado, pero se le permite pasar a través de otros enrutadores en el camino. Por lo general esta opción sólo indicará unos cuantos enrutadores para forzar una ruta específica. Por ejemplo, si se desea obligar a un paquete de Londres a Sydney a ir hacia el Oeste en lugar de hacia el Este, esta opción podría especificar enrutadores en Nueva York, Los Ángeles y Honolulú. Esta opción es de mucha utilidad cuando las consideraciones políticas o económicas obligan a pasar a través de, o evitar, ciertos países.

La opción de *Registrar ruta* indica a cada enrutador a lo largo de la ruta que adjunte su dirección IP al campo Opciones. Esto permite a los administradores del sistema buscar fallas en los algoritmos de enrutamiento. Cuando se estableció ARPANET en un principio, ningún paquete pasaba nunca por más de nueve enrutadores, por lo que 40 bytes de opciones eran más que suficientes. Pero como dijimos antes, ahora son muy pocos.

Por último, la opción *Estampa de tiempo* es como la opción Registrar ruta, excepto que además de registrar su dirección IP de 32 bits, cada enrutador también registra una estampa de tiempo de 32 bits. Esta opción también es principalmente útil para la medición en las redes.

Hoy en día se han dejado de usar las opciones IP. La mayoría de los enrutadores las ignoran o no las procesan en forma eficiente, haciéndolas a un lado como un caso poco común. Es decir, sólo cuentan con soporte parcial y se usan muy raras veces.

Direcciones IP

Una característica que define a IPv4 consiste en sus direcciones de 32 bits. Cada host y enrutador de Internet tiene una dirección IP que se puede usar en los campos Dirección de origen y Dirección de destino de los paquetes IP. Es importante tener en cuenta que una dirección IP en realidad no se refiere a un host, sino a una interfaz de red, por lo que si un host está en dos redes, debe tener dos direcciones IP. Sin embargo, en la práctica la mayoría de los hosts están en una red y, por ende, tienen una dirección IP. En contraste, los enrutadores tienen varias interfaces y, por lo tanto, múltiples direcciones IP.

Prefijos

A diferencia de las direcciones Ethernet, las direcciones IP son jerárquicas. Cada dirección de 32 bits está compuesta de una porción de red de longitud variable en los bits superiores, y de una porción de host en los bits inferiores. La porción de red tiene el mismo valor para todos los hosts en una sola red, como una LAN Ethernet. Esto significa que una red corresponde a un bloque contiguo de espacio de direcciones IP. A este bloque se le llama **prefijo**.

Las direcciones IP se escriben en **notación decimal con puntos**. En este formato, cada uno de los 4 bytes se escribe en decimal, de 0 a 255. Por ejemplo, la dirección hexadecimal 80D00297 de 32 bits se escribe como 128.208.2.151. Para escribir los prefijos, se proporciona la dirección IP menor en el bloque y el tamaño del mismo. El tamaño se determina mediante el número de bits en la porción de red; el resto de los bits en la porción del host pueden variar. Esto significa que el tamaño debe ser una potencia de dos. Por convención, el prefijo se escribe después de la dirección IP como una barra diagonal seguida de la longitud en bits de la porción de red. En nuestro ejemplo, si el prefijo contiene 28 direcciones y, por lo tanto, deja 24 bits para la porción de red, se escribe como 128.208.0.0/24.

Como la longitud del prefijo no se puede inferir sólo a partir de la dirección IP, los protocolos de enrutamiento deben transportar los prefijos hasta los enrutadores. Algunas veces los prefijos se describen simplemente mediante su longitud, como en un "/16" que se pronuncia como "*slash 16*". La longitud del prefijo corresponde a una máscara binaria de 1s en la porción de red. Cuando se escribe de esta forma, se denomina máscara de subred. Se puede aplicar un AND a la máscara de subred con la dirección IP para extraer sólo la porción de la red. Para nuestro ejemplo, la máscara de subred es 255.255.255.0. La Ilustración 47 muestra un prefijo y una máscara de subred.

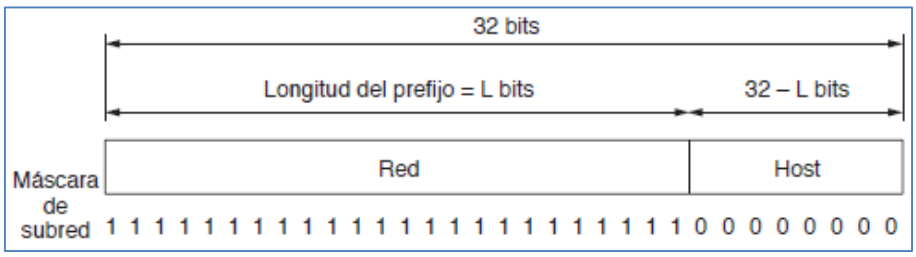


Ilustración 47 - Un prefijo y una máscara de subred del protocolo IP.

Las direcciones jerárquicas tienen ventajas y desventajas considerables. La ventaja clave de los prefijos es que los enrutadores pueden reenviar paquetes con base en la porción de red de la dirección, siempre y cuando cada una de las redes tenga un bloque de direcciones único. La porción del host no importa a los enrutadores, ya que enviarán en la misma dirección a todos los hosts de la misma red. Sólo hasta que los paquetes llegan a la red de destino es cuando se reenvían al host correcto. Esto hace que las tablas de enrutamiento sean mucho más pequeñas de lo que podrían ser con cualquier otro método.

Aunque el uso de una jerarquía permite a Internet escalar, tiene dos desventajas. En primer lugar, la dirección IP de un host depende de su ubicación en la red. Una dirección Ethernet se puede usar en cualquier parte del mundo, pero cada dirección IP pertenece a una red específica, por lo que los enrutadores sólo podrán entregar paquetes destinados a esa dirección en la red. Se necesitan diseños como IP móvil para soportar hosts que se desplacen de una red a otra, pero que deseen mantener las mismas direcciones IP.

La segunda desventaja es que la jerarquía desperdicia direcciones a menos que se administre con cuidado. Si se asignan direcciones a las redes en bloques (muy) grandes, habrá (muchas) direcciones que se asignen pero no se utilicen. Esta asignación no sería de gran importancia si hubiera muchas direcciones para repartir. Sin embargo, hace más de dos décadas se descubrió que el tremendo crecimiento de Internet estaba agotando con rapidez el espacio de direcciones libres. IPv6 es la solución a este agotamiento, pero hasta que no se implemente de manera amplia habrá mucha presión por asignar direcciones IP, de manera que se utilicen con mucha eficiencia.

Subredes

Para evitar conflictos, los números de red se administran a través de una corporación sin fines de lucro llamada **ICANN** (Corporación de Internet para la Asignación de Nombres y Números, del inglés *Internet Corporation for Assigned Names and Numbers*). Esta corporación ha delegado partes de este espacio de direcciones a varias autoridades regionales, las cuales reparten las direcciones IP a los ISP y otras compañías. Éste es el proceso por el cual se asigna un bloque de direcciones IP a una compañía.

Sin embargo, este proceso es sólo el principio de la historia, puesto que la asignación de direcciones IP es continua a medida que crecen las compañías. Ya dijimos antes que el enrutamiento por prefijo requiere que todos los hosts en una red tengan el mismo número de red. Esta propiedad puede provocar problemas a medida que las redes aumentan su tamaño. Por ejemplo, considere una universidad que empezó con nuestro prefijo de ejemplo /16, para que el Departamento de Ciencias Computacionales lo use en las computadoras de su red Ethernet. Un año después, el Departamento de Ingeniería eléctrica desea entrar a Internet. Pronto le sigue el Departamento de Arte. ¿Qué direcciones IP deben usar estos departamentos? Para obtener más bloques hay que ir fuera de la universidad y puede ser costoso o inconveniente. Además, el prefijo /16 que ya está asignado tiene suficientes direcciones para más de 60000 hosts. Tal vez se tenga pensado un crecimiento considerable, pero mientras esto no ocurra sería un desperdicio asignar más bloques de direcciones IP a la misma universidad. Se requiere una organización diferente.

La solución es dividir el bloque de direcciones en varias partes para uso interno en forma de múltiples redes, pero actuar como una sola red para el mundo exterior. A estas partes de la red se les llama subredes. Como mencionamos en la unidad 1, hay que estar conscientes de que este nuevo uso del término entra en conflicto con el uso anterior de "*subred*", cuyo significado es el conjunto de todos los enrutadores y líneas de comunicación en una red.

La Ilustración 48 muestra cómo pueden ayudar las subredes en nuestro ejemplo. El prefijo /16 se dividió en varias piezas. Esta división no necesita ser uniforme, pero hay que alinear cada pieza de manera que se pueda usar cualquier bit en la porción inferior del host. En este caso, la mitad del bloque (un /17) se asigna al Departamento de Ciencias Computacionales, una cuarta parte se asigna al Departamento de Ingeniería Eléctrica (un /18) y una octava parte (un /19) al Departamento de Arte. El octavo restante queda sin asignar. Una manera distinta de ver cómo se dividió el bloque es analizar los prefijos resultantes cuando se escriben en notación binaria:

Ciencias Computacionales:	10000000	11010000	1|xxxxxxx	xxxxxxxx
Ingeniería Eléctrica:	10000000	11010000	00|xxxxxx	xxxxxxxx
Arte:	10000000	11010000	011|xxxxx	xxxxxxxx

Aquí, la barra vertical (|) muestra el límite entre el número de la subred y la porción del host.

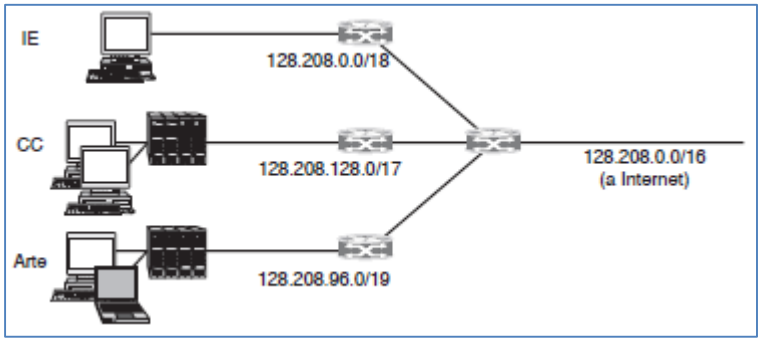


Ilustración 48 - División de un prefijo IP en redes separadas mediante el uso de subredes.

Cuando llega un paquete al enrutador principal, ¿cómo sabe a cuál subred entregarlo? Aquí es donde entran los detalles de nuestros prefijos. Una forma sería que cada enrutador tuviera una tabla con 65535 entradas que le indicaran qué línea de salida usar para cada host en el campus. Pero esto socavaría el principal beneficio de escalamiento que obtenemos al usar una jerarquía. En cambio, los enrutadores simplemente tienen que conocer las máscaras de subred para las redes en el campus.

Cuando llega un paquete, el enrutador analiza su dirección de destino y verifica a qué subred pertenece. Para ello, el enrutador aplica un AND a la dirección del destino con la máscara para cada subred y verifica que el resultado sea el prefijo correspondiente. Por ejemplo, considere un paquete destinado para la dirección IP 128.208.2.151. Para ver si está dirigido al Departamento de Ciencias Computacionales, le aplicamos un AND con 255.255.128.0 para tomar los primeros 17 bits (que son 128.208.0.0) y ver si coinciden con la dirección del prefijo (que es 128.208.128.0). En este caso no coinciden. Si verificamos los primeros 18 bits para el Departamento de Ingeniería Eléctrica, obtenemos 128.208.0.0 al aplicar un AND entre éstos y la máscara de subred. Este resultado coincide con la dirección del prefijo, por lo que el paquete se reenvía por la interfaz que conduce a la red de ingeniería eléctrica.

Las divisiones de las subredes se pueden cambiar más adelante si es necesario, para lo cual se actualizan todas las máscaras de subred en los enrutadores dentro de la universidad. Fuera de la red, las subredes no son visibles, por lo que para asignar una nueva subred no hay que ponerse en contacto con la ICANN ni cambiar bases de datos externas.

CIDR: Enrutamiento Interdominio sin Clases

Incluso si se asignan bloques de direcciones IP de manera que las direcciones se utilicen con eficiencia, de todas formas hay un problema presente: la explosión de las tablas de enrutamiento.

Los enrutadores en las organizaciones en el extremo de una red, como una universidad, necesitan tener una entrada para cada una de sus subredes, de manera que el enrutador pueda saber qué línea usar para llegar a esa red. Para las rutas a destinos fuera de la organización, sólo necesitan usar la regla simple predeterminada de enviar los paquetes en la línea hacia el ISP que conecta a la organización con el resto de Internet. Las otras direcciones de destino deben estar por ahí en alguna parte.

Los enrutadores en los ISP y las redes troncales en medio de Internet no se pueden dar esos lujos. Ellos deben saber qué ruta tomar hacia cada una de las redes; aquí no funciona la regla simple predeterminada. Se dice que estos enrutadores básicos están en la **zona libre predeterminada** de Internet. Nadie sabe en realidad cuántas redes están conectadas a Internet, pero es una gran cantidad. Esto puede generar una tabla muy grande. Tal vez no parezca muy grande según los estándares computacionales, pero hay que tener en cuenta que los enrutadores deben realizar una búsqueda en esta tabla para reenviar cada paquete, y los enrutadores en los ISP grandes pueden reenviar hasta millones de paquetes por segundo. Se requiere un hardware especializado y una memoria rápida para procesar paquetes a estas tasas de transmisión, no una computadora de propósito general.

Además, los algoritmos de enrutamiento requieren que cada enrutador intercambie información sobre las direcciones a las que puede llegar con otros enrutadores. Entre más grandes sean las tablas, más información habrá que comunicar y procesar. El procesamiento aumenta por lo menos en forma lineal con respecto al

tamaño de la tabla. Una mayor comunicación aumenta la probabilidad de que algunas partes se pierdan, por lo menos en forma temporal, lo que tal vez conduzca a inestabilidades en el enrutamiento.

El problema de las tablas de enrutamiento se podría haber resuelto mediante el uso de una jerarquía más profunda, como la red telefónica. Por ejemplo, hacer que cada dirección IP contenga campos de país, estado/provincia, ciudad, red y host podría funcionar. Así, cada enrutador sólo tendría que saber cómo llegar a cada país, a los estados o provincias en su propio país, a las ciudades en su estado o provincia, y a las redes en su ciudad. Por desgracia, esta solución requeriría mucho más de 32 bits para las direcciones IP y utilizaría las direcciones de una manera ineficiente.

Por fortuna, hay algo que podemos hacer para reducir los tamaños de las tablas de enrutamiento. Podemos aplicar la misma perspectiva que en las subredes: los enrutadores en distintas ubicaciones pueden saber acerca de una dirección IP dada que pertenece a prefijos de distintas zonas. Sin embargo, en vez de dividir un bloque de direcciones en subredes, aquí combinamos varios prefijos pequeños en un solo prefijo más grande. A este proceso se le conoce como **agregación de rutas**. Algunas veces al prefijo más grande resultante se le denomina **superred** para contrastar con las subredes como la división de bloques de direcciones.

Con la agregación, las direcciones IP están contenidas en prefijos de diversos tamaños. La misma dirección IP que un enrutador trata como parte de un prefijo /22 (un bloque que contiene 2^{10} direcciones), puede ser tratada por otro enrutador como parte de un prefijo /20 más grande (que contiene 2^{12} direcciones). Es responsabilidad de cada enrutador tener la información del prefijo correspondiente. Este diseño funciona con las subredes y se conoce como **CIDR** (Enrutamiento Interdominio sin Clases, del inglés *Classless InterDomain Routing*). El nombre resalta el contraste con las direcciones que codifican la jerarquía con las clases, lo cual describiremos en breve.

Para facilitar el entendimiento de CIDR, vamos a considerar un ejemplo en el que hay un bloque de 8192 direcciones IP disponible, empezando en 194.24.0.0. Suponga que la Universidad de Cambridge necesita 2048 direcciones y se le asignan las direcciones 194.24.0.0 a 194.24.7.255, junto con la máscara 255.255.248.0. Éste es un prefijo /21. A continuación, la Universidad de Oxford pide 4096 direcciones. Como un bloque de 4096 direcciones debe estar en un límite de 4096 bytes, a Oxford no se le pueden asignar direcciones que empiecen en 192.24.8.0. Sin embargo, recibe las direcciones 192.24.16.0 a 192.24.31.255, junto con la máscara de subred 255.255.240.0. Por último, la Universidad de Edimburgo pide 1024 direcciones y se le asignan las direcciones 192.24.8.0 a 194.24.11.255 junto con la máscara 255.255.252.0. En la Ilustración 49 se sintetizan estas asignaciones.

Universidad	Primera dirección	Última dirección	Cuántas	Prefijo
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edimburgo	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Disponible)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Ilustración 49 - Un conjunto de asignaciones de direcciones IP.

A todos los enrutadores en la zona libre predeterminada se les dice ahora sobre las direcciones IP en las tres redes. Los enrutadores cercanos a las universidades tal vez necesiten enviar por una línea de salida distinta para cada uno de los prefijos, así que ellos necesitan una entrada para cada uno de los prefijos en sus tablas de enrutamiento. Un ejemplo es el enrutador en Londres de la Ilustración 50.

Ahora veamos estas tres universidades desde el punto de vista de un enrutador distante en Nueva York. Todas las direcciones IP en los tres prefijos se deben enviar de Nueva York (o de Estados Unidos en general) a Londres. El proceso de enrutamiento en Londres detecta esto y combina los tres prefijos en una sola entrada agregada para el prefijo 194.24.0.0/19, que pasa al enrutador de Nueva York. Este prefijo contiene 8K direcciones y cubre las tres universidades, junto con las otras 1024 direcciones no asignadas. Mediante el uso de la agregación, estos prefijos se redujeron a uno, con lo cual se redujeron tanto los prefijos que se deben informar al enrutador de Nueva York, como las entradas en la tabla de enrutamiento de éste.

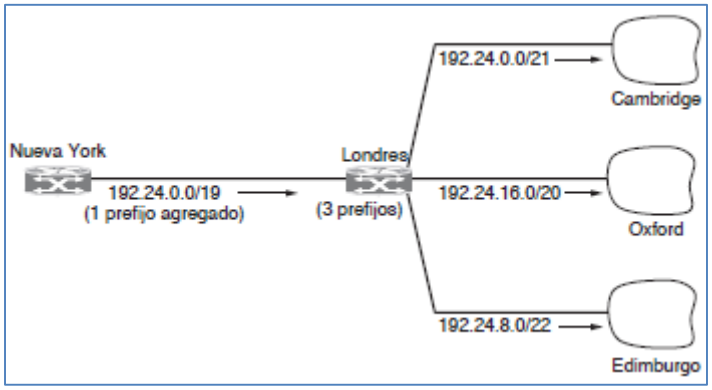


Ilustración 49 - Agregación de prefijos IP.

Una vez que se activa la agregación, es un proceso automático que depende de qué prefijos están ubicados en qué parte de Internet, no en las acciones de un administrador que asigna direcciones a las redes. La agregación se utiliza mucho en Internet y puede reducir el tamaño de las tablas de los enrutadores de manera considerable.

Como una sorpresa adicional, es posible traslapar a los prefijos. La regla es que los paquetes se envíen en la dirección de la ruta más específica, o el prefijo más largo coincidente que tenga la menor cantidad de direcciones IP. El enrutamiento del prefijo más largo coincidente ofrece un grado conveniente de flexibilidad, como podemos ver en el comportamiento del enrutador de Nueva York en la Ilustración 51. Este enrutador aún utiliza un solo prefijo agregado para enviar tráfico de las tres universidades a Londres. Sin embargo, el bloque previamente disponible de direcciones dentro de este prefijo se ha asignado ahora a una red en San Francisco. Una posibilidad es que el enrutador de Nueva York mantenga cuatro prefijos, y que envíe los paquetes de tres de ellos a Londres y los paquetes del cuarto prefijo a San Francisco. Sin embargo, el enrutamiento del prefijo más largo coincidente puede manejar este reenvío de paquetes con los dos prefijos que se muestran. Un prefijo general se utiliza para dirigir el tráfico de todo el bloque a Londres. También se utiliza uno más específico para dirigir una parte del prefijo más grande a San Francisco. Con la regla del prefijo más largo coincidente, las direcciones IP dentro de la red de San Francisco se enviarán en la línea de salida a San Francisco, y todas las demás direcciones IP en el prefijo más grande se enviarán a Londres.

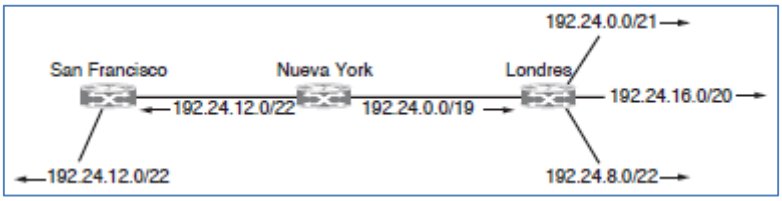


Ilustración 50 - Enrutamiento del prefijo más largo coincidente en el enrutador de Nueva York.

En concepto, el CIDR funciona de la siguiente manera. Cuando llega un paquete, se explora la tabla de enrutamiento para determinar si el destino está dentro del prefijo. Es posible que coincidan varias entradas con distintas longitudes de prefijos, en cuyo caso se utiliza la entrada con el prefijo más largo. Por ende, si hay una coincidencia para una máscara /20 y una máscara /24, se utiliza la entrada /24 para buscar la línea de salida para el paquete. Sin embargo, este proceso sería tedioso si la tabla realmente se explorara entrada por entrada. En cambio, se han ideado algoritmos complejos para agilizar el proceso de coincidencia de direcciones. Los enrutadores comerciales usan chips VLSI personalizados con estos algoritmos incrustados en el hardware.

Direccionamiento con clases y especial

Para ayudar al lector a que aprecie mejor la razón de por qué CIDR es tan útil, relataremos brevemente el diseño anterior a éste. Antes de 1993, las direcciones IP se dividían en las cinco categorías listadas en la Ilustración 52. Esta asignación se denominó **direccionamiento con clases**.

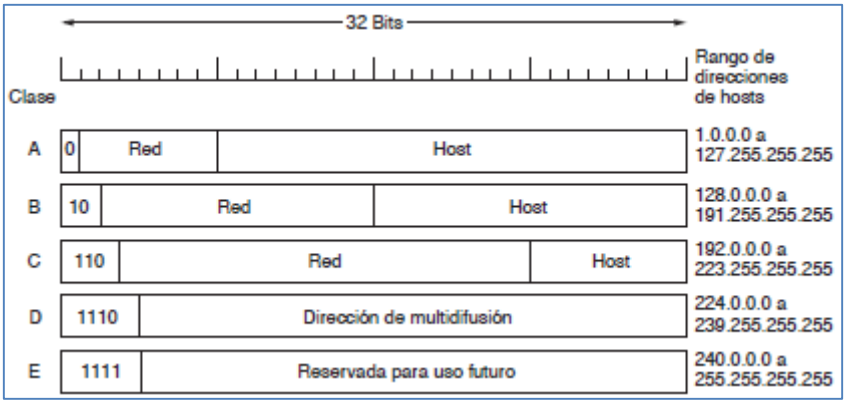


Ilustración 51 - Formatos de direcciones IP.

Los formatos de clase *A*, *B* y *C* permiten hasta 128 redes con 16 millones de hosts cada una, 16384 redes con hasta 65536 hosts cada una y 2 millones de redes (por ejemplo, LAN) con hasta 256 hosts cada una (aunque unas cuantas de éstas son especiales). También se soporta la multidifusión (el formato de clase *D*), en el cual un datagrama se dirige a múltiples hosts. Las direcciones que empiezan con 1111 se reservan para un uso futuro. Serían valiosas para usar ahora, dado que se está agotando el espacio de direcciones IPv4. Por desgracia, muchos hosts no aceptarán estas direcciones como válidas, puesto que han estado fuera de los límites por tanto tiempo y es difícil enseñar nuevos trucos a los hosts viejos.

Éste es un diseño jerárquico, pero a diferencia de CIDR, los tamaños de los bloques de direcciones son fijos. Existen más de 2000 millones de direcciones, pero al organizar el espacio de direcciones en clases se desperdician millones de ellas. En particular, el verdadero problema es la red clase *B*. Para la mayoría de las organizaciones, una red clase *A* con 16 millones de direcciones es demasiado grande, y una red clase *C* con 256 direcciones es muy pequeña. Una red clase *B* con 65536 direcciones es justo lo necesario. En el folclor de Internet, esta situación se conoce como el problema de los tres osos [del cuento Ricitos de oro y los tres osos].

En realidad, una dirección clase *B* es demasiado grande para la mayoría de las organizaciones. Hay estudios que demuestran que más de la mitad de todas las redes clase *B* tienen menos de 50 hosts. Una red clase *C* habría bastado, pero sin duda todas las organizaciones que solicitaron una dirección clase *B* pensaron que un día el campo de hosts de 8 bits les quedaría pequeño. En retrospectiva, podría haber sido mejor que las redes clase *C* usaran 10 bits en lugar de 8 para el número de host, permitiendo 1022 hosts por red. De haber sido éste el caso, la mayoría de las organizaciones probablemente se habría conformado con una red clase *C* y hubiera existido medio millón de ellas (en vez de sólo 16384 redes clase *B*).

Es duro culpar a los diseñadores de Internet por no haber proporcionado más (y más pequeñas) direcciones de clase *B*. En el momento en que se tomó la decisión de crear las tres clases, Internet era una red de investigación que conectaba las principales universidades de investigación en Estados Unidos (más un número muy pequeño de compañías y sitios del ejército que hacían investigación de redes). En esa época nadie percibía que Internet llegaría a ser un sistema de comunicación de mercado masivo que rivalizaría con la red telefónica.

Para lidiar con estos problemas se introdujeron las subredes para asignar de manera flexible bloques de direcciones dentro de una organización. Después se agregó el CIDR para reducir el tamaño de la tabla de enrutamiento global. Hoy en día, los bits que indican si una dirección IP pertenece a una red clase *A*, *B* o *C* ya no se utilizan, aunque las referencias a estas clases en la literatura siguen siendo comunes.

Para ver cómo el hecho de descartar las clases complicó aún más el reenvío de paquetes, considere lo simple que era en el viejo sistema con clases. Cuando llegaba un paquete en un enrutador, se desplazaba una copia de la dirección IP 28 bits a la derecha para producir un número de clase de 4 bits. Después una ramificación de 16 vías ordenaba los paquetes en las clases *A*, *B*, *C* (junto con *D* y *E*), con ocho casos para la clase *A*, cuatro casos para la clase *B* y dos casos para la clase *C*. Después se enmascaraba el código para cada clase del número de red de 8, 16 o 24 bits y se alineaba a la derecha en una palabra de 32 bits. Luego se buscaba el número de red en la tabla *A*, *B* o *C*, por lo general mediante el indexado para las redes *A* y *B*, y el *hashing* para las redes C. Una vez que se encontraba la entrada, se podía buscar la línea de salida y reenviar el paquete. Esto es mucho

más simple que la operación del prefijo más largo coincidente, el cual ya no puede usar una simple búsqueda en una tabla debido a que una dirección IP puede tener un prefijo de cualquier longitud.

Las direcciones de clase *D* se siguen utilizando en Internet para la multidifusión. En realidad podría ser más preciso decir que se están empezando a usar para multidifusión, puesto que en el pasado la multidifusión en Internet no se ha implementado mucho.

También existen otras direcciones con significados especiales, como se muestra en la Ilustración 53. La dirección IP 0.0.0.0, que es la dirección más baja, es utilizada por los hosts al momento de encenderlos. Significa "esta red" o "este host". Las direcciones IP con 0 como número de red se refieren a la red actual. Estas direcciones permiten que las máquinas hagan referencia a su propia red sin conocer su número (pero tienen que conocer la máscara de red para saber cuántos 0s incluir). La dirección que consiste sólo en 1s, o 255.255.255.255 (la dirección más alta), se utiliza para indicar a todos los hosts en la red especificada. Permite la difusión en la red local, por lo general una LAN. Las direcciones con un número de red apropiado y 1s en el campo de host permiten a las máquinas enviar paquetes de difusión a redes LAN distantes en cualquier parte de Internet. Sin embargo, muchos administradores de red deshabilitan esta característica, ya que es sobre todo un peligro de seguridad. Por último, todas las direcciones de la forma 127.xx.yy.zz se reservan para las pruebas de loopback. Los paquetes que se envían a esa dirección no se ponen en el cable; se procesan en forma local y se tratan como si fueran paquetes entrantes. Esto permite enviar paquetes al host sin que el emisor conozca su número, lo cual es útil para realizar pruebas.

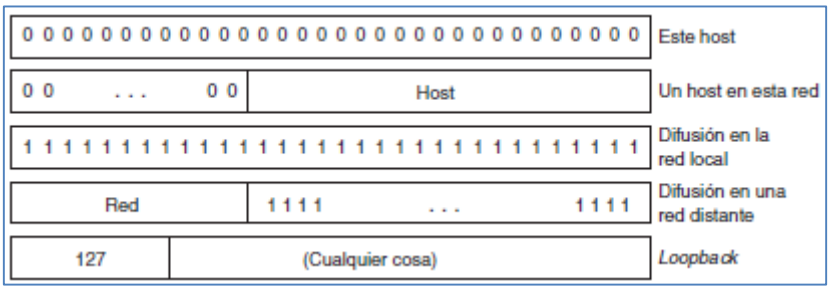


Ilustración 52 - Direcciones IP especiales.

NAT: Traducción de Dirección de Red

Las direcciones IP son escasas. Un ISP podría tener una dirección con prefijo de /16, lo cual le da 65534 números de host. Si tiene más clientes que esos, tiene un problema.

Esta escasez ha conducido a técnicas para usar las direcciones IP con moderación. Un método es asignar dinámicamente una dirección IP a una computadora cuando ésta se encuentra encendida y usa la red, y tomar de vuelta la dirección IP cuando el host se vuelve inactivo. Así, la dirección IP se puede asignar a otra computadora que se active en ese momento. De esta manera, una sola dirección de /16 puede manejar hasta 65534 usuarios activos.

Esta estrategia funciona bien en algunos casos, por ejemplo, para las redes de marcación, las computadoras móviles y otras computadoras que pueden estar temporalmente ausentes o apagadas. Sin embargo, no funciona muy bien para los clientes comerciales. Muchas PC en negocios deben estar prendidas en forma continua. Algunas son máquinas de los empleados, que se respaldan en la noche, y otras son servidores que tal vez necesiten atender una solicitud remota sin previo aviso. Estos negocios tienen una línea de acceso que siempre proporciona conectividad al resto de Internet.

Esta situación se aplica cada vez más a los usuarios domésticos que se suscriben a ADSL o Internet por cable, ya que no hay un cargo por conexión (sólo una tarifa mensual fija). Muchos de estos usuarios tienen dos o más computadoras en su hogar, a menudo una para cada miembro de la familia, y todos quieren estar en línea todo el tiempo. La solución es conectar todas las computadoras en una red doméstica a través de una LAN y colocar un enrutador (inalámbrico) en ella. Así, el enrutador se conecta con el ISP. Desde el punto de vista del ISP, la familia es ahora lo mismo que un pequeño negocio con un puñado de computadoras. Con las técnicas que hemos visto hasta ahora, cada computadora debe tener su propia dirección IP todo el tiempo. Para un ISP

con muchos miles de clientes, en especial clientes comerciales y familias que son casi como pequeños negocios, la demanda de direcciones IP puede exceder rápidamente el bloque disponible.

El problema de quedarse sin direcciones IP no es uno teórico que podría ocurrir en cierto momento en un futuro no muy distante. Está ocurriendo justo ahora. La solución a largo plazo es que toda la Internet migre a IPv6, que cuenta con direcciones de 128 bits. Esta transición está ocurriendo lentamente, pero pasarán años antes de que el proceso esté completo. Mientras tanto, para sobrevivir se requería una solución rápida. Esta solución, que se utiliza ampliamente en la actualidad, se conoce como **NAT** (Traducción de Dirección de Red, del inglés *Network Address Translation*) y se describe en el RFC 3022.

La idea básica detrás de NAT es que el ISP asigne a cada hogar o negocio una sola dirección IP (o a lo más, una pequeña cantidad de éstas) para el tráfico de Internet. Dentro de la red del cliente, cada computadora obtiene una dirección IP única, la cual se utiliza para enrutar el tráfico interno. Sin embargo, justo antes de que un paquete salga de la red del cliente y vaya al ISP, la dirección IP única interna se traduce a la dirección IP pública compartida. Esta traducción hace uso de los tres rangos de direcciones IP que se han declarado como privados. Las redes pueden utilizarlos de manera interna como deseen. La única regla es que no pueden aparecer paquetes que contengan estas mismas direcciones en Internet. Los tres rangos reservados son:

10.0.0.0 - 10.255.255.255/8	(16777216 hosts)
172.16.0.0 - 172.31.255.255/12	(1048576 hosts)
192.168.0.0 - 192.168.255.255/16	(65536 hosts)

En la Ilustración 54 se muestra la operación de NAT. Dentro de las premisas del cliente, cada máquina tiene una dirección única de la forma 10.x.y.z. Sin embargo, antes de que un paquete salga de las premisas del cliente, pasa a través de una **caja NAT** que convierte la dirección IP de origen interna, 10.0.0.1 en la figura, a la dirección IP verdadera del cliente, 198.60.42.12 en este ejemplo. A menudo, la caja NAT se combina en un solo dispositivo con un *firewall* (corta fuegos) que proporciona seguridad controlando cuidadosamente lo que entra y sale por la red de la compañía. También es posible integrar la caja NAT en un enrutador o módem ADSL.

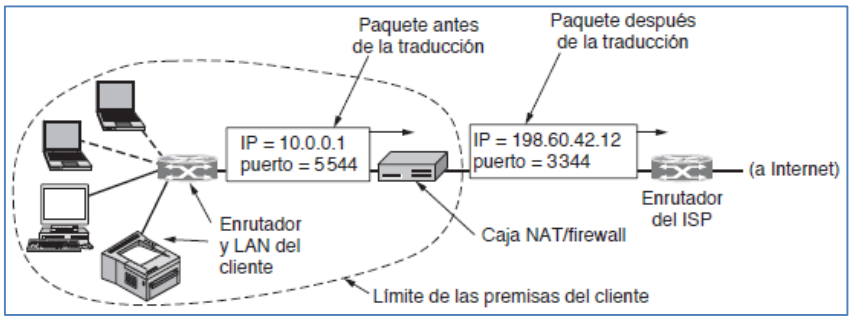


Ilustración 53 - Colocación y funcionamiento de una caja NAT.

Hasta ahora hemos ignorado un detalle pequeño pero crucial: cuando la respuesta vuelve (por ejemplo, de un servidor web), se dirige naturalmente a 198.60.42.12, por lo que, ¿cómo sabe ahora la caja NAT con qué dirección interna se reemplaza? Aquí está el problema con NAT. Si hubiera un campo extra en el encabezado IP, ese campo podría usarse para guardar el registro del emisor real, pero sólo queda 1 bit sin usar. En principio, podría crearse una nueva opción para mantener la verdadera dirección de origen, pero para ello habría que cambiar el código IP en todas las máquinas de todo Internet para manejar la nueva opción. Ésta no es una alternativa prometedora para un arreglo rápido.

Lo que realmente pasa es lo siguiente. Los diseñadores de NAT observaron que la mayoría de los paquetes IP llevan cargas útiles de TCP o UDP. En las próximas unidades veremos que los dos tienen encabezados que contienen un puerto de origen y un puerto de destino. Más adelante explicaremos sólo los puertos TCP, pero esa misma explicación es válida para los puertos UDP. Los puertos son enteros de 16 bits que indican dónde empieza y dónde acaba la conexión TCP. Estos puertos proporcionan el campo requerido para hacer que NAT funcione.

Cuando un proceso desea establecer una conexión TCP con un proceso remoto, se conecta a un puerto TCP sin usar en su propia máquina. Éste se conoce como **puerto de origen** y le indica al código TCP dónde enviar los paquetes entrantes que pertenecen a esta conexión. El proceso también proporciona un **puerto de destino** para indicar a quién se deben dar los paquetes en el lado remoto. Los puertos 0-1023 se reservan para los servicios conocidos. Por ejemplo, 80 es el puerto usado por los servidores web, para que los clientes remotos puedan localizarlos. Cada mensaje TCP de salida contiene un puerto de origen y uno de destino. En conjunto, estos puertos sirven para identificar los procesos que usan la conexión en ambos extremos.

Los puertos son efectivamente 16 bits adicionales de direccionamiento, que identifican qué proceso obtiene cuál paquete entrante.

Si utilizamos el campo *Puerto de origen*, podemos resolver nuestro problema de asignación. Siempre que un paquete de salida entra en la caja NAT, la dirección de origen 10.x.y.z se reemplaza por la verdadera dirección IP del cliente. Además, el campo *Puerto de origen* TCP se reemplaza por un índice en la tabla de traducción de 65536 entradas de la caja NAT. Esta entrada de la tabla contiene el puerto de origen y la dirección IP originales. Finalmente, las sumas de verificación de los encabezados IP y TCP se recalculan e insertan en el paquete. Es necesario reemplazar el *Puerto de origen*, porque podría ocurrir que ambas conexiones de las máquinas 10.0.0.1 y 10.0.0.2 usaran el puerto 5000, por ejemplo, así que el **Puerto de origen** no basta por sí solo para identificar el proceso de envío.

Cuando un paquete llega a la caja NAT desde el ISP, el *Puerto de origen* en el encabezado TCP se extrae y utiliza como un índice en la tabla de asignación de la caja NAT. Desde la entrada localizada, la dirección IP interna y el *Puerto de origen* TCP se extraen e insertan en el paquete. Luego, las sumas de verificación de IP y TCP se recalculan e insertan en el paquete. Entonces el paquete se pasa al enrutador del cliente para su entrega normal mediante el uso de la dirección 10.x.y.z.

Aunque este tipo de esquema resuelve el problema, muchos puristas en la comunidad de IP lo consideran a primera vista como una abominación. He aquí algunas de las objeciones sintetizadas brevemente. Primero, NAT viola el modelo arquitectónico de IP, el cual establece que cada dirección IP identifica a una sola máquina en forma única y a nivel mundial. Toda la estructura del software de Internet se basa en este hecho. Con NAT, miles de máquinas pueden usar (y lo hacen) la dirección 10.0.0.1.

Segundo, NAT quebranta el modelo de conectividad de extremo a extremo de Internet, que establece que cualquier host puede enviar un paquete a cualquier otro en cualquier momento dado. Como la asignación en la caja NAT se establece mediante los paquetes de salida, no se pueden aceptar paquetes entrantes sino hasta después de los paquetes de salida. En la práctica, esto significa que un usuario doméstico con NAT puede hacer conexiones TCP/IP a un servidor web remoto, pero un usuario remoto no puede hacer conexiones a un servidor de juego en la red doméstica. Se requiere una configuración especial o técnicas de **NAT transversal** para soportar este tipo de situación.

Tercero, NAT cambia a Internet de una red sin conexión a un tipo especial de red orientada a conexión. El problema es que la caja NAT debe mantener la información (es decir, la asignación) para cada conexión que pasa a través de ella. Hacer que la red mantenga el estado de la conexión es una propiedad de las redes orientadas a conexión, no de las redes sin conexión. Si la caja NAT falla y se pierde su tabla de asignación, se destruirán todas sus conexiones TCP. En ausencia de NAT, un enrutador puede fallar y reiniciarse sin un efecto a largo plazo sobre las conexiones TCP. El proceso de envío apenas queda fuera unos segundos y retransmite todos los paquetes cuya recepción no se haya confirmado. Con NAT, Internet es tan vulnerable como una red de circuitos conmutados.

Cuarto, NAT viola la regla más fundamental de los protocolos distribuidos en capas: la capa *k* no puede hacer ninguna suposición acerca de lo que la capa *k+1* ha puesto en el campo de carga útil. Este principio básico está ahí para que las capas sigan siendo independientes. Si TCP se actualiza después a TCP-2, con un diseño de encabezado diferente NAT fallará. Toda la idea de los protocolos distribuidos en capas es asegurar que los cambios en una capa no requieran cambios en otras capas. NAT destruye esta independencia.

Quinto, en Internet no se exige que los procesos utilicen TCP o UDP. Si un usuario en la máquina *A* decide usar algún nuevo protocolo de transporte para hablar con un usuario en la máquina *B* (por ejemplo, para una

aplicación multimedia), la introducción de una caja NAT hará que la aplicación falle, ya que la caja NAT no podrá localizar el Puerto de origen TCP de forma correcta.

Un sexto problema relacionado consiste en que algunas aplicaciones usan múltiples conexiones TCP/IP o puertos UDP en las formas prescritas. Por ejemplo, el **FTP** (Protocolo de Transferencia de Archivos estándar, del inglés *File Transfer Protocol*) inserta direcciones IP en el cuerpo del paquete para que el receptor las extraiga y utilice. Como NAT no sabe nada sobre estos arreglos, no puede reescribir las direcciones IP ni justificarlas de alguna otra forma. Esta falta de entendimiento significa que FTP y otras aplicaciones como el protocolo de telefonía de Internet **H.323** pueden fallar en presencia de NAT, a menos que se tomen precauciones especiales. A menudo es posible arreglar NAT para estos casos, pero no es una buena idea tener que arreglar el código en la caja NAT cada vez que surge una nueva aplicación.

Por último, debido a que el campo *Puerto de origen* de TCP es de 16 bits, a lo sumo se pueden asignar 65536 máquinas a una dirección IP. En realidad, el número es ligeramente menor porque los primeros 4096 puertos se reservan para usos especiales. Pero si hay varias direcciones IP disponibles, cada una puede manejar hasta 61440 máquinas.

En el RFC 2993 se explican éstos y otros problemas con NAT. A pesar de estas cuestiones, NAT se utiliza mucho en la práctica, en especial para las redes domésticas y de negocios pequeños, como la única técnica conveniente para lidiar con la escasez de direcciones IP. Se ha visto envuelta con los firewalls y la privacidad debido a que bloquea de manera predeterminada los paquetes entrantes no solicitados. Por esta razón es muy poco probable que desaparezca, incluso aunque el IPv6 se implemente de manera amplia.

IP versión 6

IP se ha utilizado intensivamente durante décadas. Ha trabajado muy bien, como lo demuestra el crecimiento exponencial de Internet. Por desgracia, IP se ha convertido en una víctima de su propia popularidad: el 3 de febrero de 2011, IANA asignó los últimos bloques libres. Este desastre se reconoció hace casi más de tres décadas y fue motivo de grandes discusiones y controversias dentro de la comunidad de Internet, para ver qué hacer al respecto.

En esta sección describiremos tanto el problema como varias soluciones propuestas. La única solución a largo plazo es cambiar a direcciones más grandes. **IPv6** (IP versión 6) es un diseño de reemplazo que hace precisamente eso. Puesto que utiliza direcciones de 128 bits, es muy poco probable que se vayan a agotar en un futuro previsible. Sin embargo, IPv6 ha demostrado ser muy difícil de implementar. Es un protocolo de capa de red diferente que en realidad no congenia internamente con IPv4, a pesar de tantas similitudes. Además, las empresas y los usuarios en realidad no están seguros de por qué querrían IPv6 en cualquier caso.

Además de los problemas de las direcciones, hay otras cuestiones que amenazan en segundo plano. En sus primeros años, Internet era utilizada por universidades, industrias de alta tecnología y el gobierno de Estados Unidos. (en especial, el Departamento de Defensa). Con la explosión de interés sobre Internet que empezó a mediados de la década de 1990, un grupo distinto de usuarios empezaron a utilizarla, a menudo con distintos requerimientos. En primer lugar, muchas personas con teléfonos inteligentes la utilizan para mantenerse en contacto con sus bases. En segundo lugar, con la convergencia de las industrias de las computadoras, la comunicación y el entretenimiento, los teléfonos y televisiones en el mundo se han convertido en un nodo de Internet. Bajo estas circunstancias, era evidente que IP tenía que evolucionar y hacerse más flexible.

Al ver estos problemas en el horizonte, en 1990 la IETF empezó a trabajar sobre una nueva versión de IP, una que nunca se quedara sin direcciones, que resolviera una variedad de problemas adicionales y también fuera más flexible y eficiente. Sus principales objetivos eran:

1. **Soportar miles de millones de hosts**, incluso con una asignación de direcciones ineficiente.
2. **Reducir el tamaño de las tablas de enrutamiento**.
3. **Simplificar el protocolo** para permitir a los enrutadores procesar los paquetes con más rapidez.
4. **Proporcionar mayor seguridad** (autentificación y privacidad).
5. **Poner más atención en cuanto al tipo de servicio**, en especial para los datos en tiempo real.
6. **Ayudar a la multidifusión** al permitir la especificación de alcances.
7. **Permitir que un host deambule libremente** sin tener que cambiar su dirección.

8. **Permitir que el protocolo evolucione** en el futuro.
9. **Permitir que el protocolo viejo y el nuevo coexistan** durante años.

El diseño de IPv6 presentaba una gran oportunidad para mejorar todas las características en IPv4 que estaban muy lejos de cumplir con lo que ahora se necesitaba. Para desarrollar un protocolo que cumpliera con todos esos requerimientos, la IETF emitió una llamada para propuestas y discusión en el RFC 1550. Al principio se recibieron 21 respuestas. Para diciembre de 1992, había siete propuestas serias en la mesa, que variaban desde hacer pequeñas correcciones a IP, hasta descartarlo por completo y reemplazarlo con un protocolo por completo distinto.

Una propuesta fue la de operar TCP sobre CLNP, el protocolo de capa de red diseñado para OSI. Con sus direcciones de 160 bits, CLNP hubiera proporcionado suficiente espacio de direcciones por un tiempo indefinido, ya que era capaz de dar a cada una de las moléculas de agua en los océanos suficientes direcciones (aproximadamente 2^5) para establecer una pequeña red. Esta opción también hubiera unificado dos de los principales protocolos de capa de red. Sin embargo, muchas personas sentían que esto hubiera sido como admitir que algo en el mundo de OSI se había hecho bien en realidad, una declaración considerada como políticamente incorrecta en los círculos de Internet. CLNP estaba modelado en forma muy parecida a IP, por lo que en realidad los dos no son tan diferentes. De hecho, el protocolo que se eligió en última instancia difiere más de IP que CLNP. Otro golpe contra CLNP era su defectuoso soporte para los tipos de servicios, algo que se requería para transmitir la multimedia con eficiencia.

Tres de las mejores propuestas se publicaron en *IEEE Network* (Deering, 1993; Francis, 1993; Katz y Ford, 1993). Después de mucha discusión, revisión y competencia por posición, se seleccionó una versión combinada modificada de las propuestas de Deering y Francis, ahora conocida como **SIPP** (Protocolo Simple de Internet Mejorado, del inglés *Simple Internet Protocol Plus*), designado también como IPv6.

IPv6 cumple bastante bien con los objetivos de la IETF. Mantiene las buenas características de IP, descarta o quita valor a las malas y agrega nuevas en donde se requiere. En general, IPv6 no es compatible con IPv4, pero es compatible con los otros protocolos auxiliares de Internet, incluyendo TCP, UDP, ICMP, IGMP, OSPF, BGP y DNS, con pequeñas modificaciones requeridas para lidiar con las direcciones más largas. A continuación describiremos las principales características de IPv6. Encontrará más información sobre este protocolo en los RFC 2460 al 2466.

Primero que nada, IPv6 tiene direcciones más largas que IPv4. Son de 128 bits, lo cual resuelve el problema deseado: proporcionar un suministro efectivamente ilimitado de direcciones de Internet. En breve hablaremos más sobre las direcciones.

La segunda mejora importante de IPv6 es la simplificación del encabezado. Sólo contiene siete campos (en comparación con los 13 de IPv4). Este cambio permite a los enrutadores procesar los paquetes con más rapidez y, por ende, mejora la velocidad de transmisión real y el retardo. También hablaremos en breve sobre el encabezado.

La tercera mejora importante es un soporte mejorado para las opciones. Este cambio era esencial con el nuevo encabezado, ya que los campos que antes eran requeridos ahora son opcionales (debido a que no se utilizan con tanta frecuencia). Además, la forma en que se representan las opciones es distinta y así es más fácil para los enrutadores omitir las porciones que no están destinadas para ellos. Esta característica agiliza el tiempo de procesamiento de los paquetes.

Una cuarta área en la que IPv6 representa un gran avance es la seguridad. La IETF tuvo suficiente con las historias en los periódicos sobre los niños precoces de 12 años que usaban sus computadoras personales para irrumpir en los bancos y las bases militares a través de Internet. Hubo una fuerte sensación de tener que hacer algo para mejorar la seguridad. La autentificación y la privacidad son características clave del nuevo IP. Sin embargo, después se modernizaron para el IPv4, por lo que en el área de la seguridad las diferencias ya no son tan grandes.

Por último, se puso más énfasis en la calidad del servicio. Se han hecho varios esfuerzos carentes de entusiasmo para mejorar la QoS en el pasado, pero ahora con el crecimiento de la multimedia en Internet, la sensación de urgencia es mayor.

El encabezado principal de IPv6

En la Ilustración 54 se muestra el encabezado de IPv6. El campo *Versión* siempre es 6 para IPv6 (y 4 para IPv4). Durante el periodo de transición de IPv4, los enrutadores podrán examinar este campo para saber qué tipo de paquete tienen. Como observación adicional, para hacer esta prueba se desperdician unas cuantas instrucciones en la ruta crítica, dado que el encabezado de enlace de datos indica por lo general el protocolo de red para demultiplexar, por lo que tal vez algunos enrutadores omitan la verificación. Por ejemplo, el campo Tipo de Ethernet tiene distintos valores para indicar una carga útil de IPv4 o IPv6. Las discusiones entre los campos "Hacerlo bien" y "Hacerlo rápido" sin duda serán extensas.

El campo *Servicios diferenciados* (originalmente conocido como *Clase de tráfico*) se utiliza para distinguir la clase de servicio para los paquetes con distintos requerimientos de entrega en tiempo real. Se utiliza con la arquitectura de servicio diferenciado para la calidad del servicio, de la misma forma que el campo con el mismo nombre en el paquete IPv4. Además, los 2 bits de menor orden se usan para señalar las indicaciones explícitas de congestión, de nuevo en la misma forma que IPv4.

El campo *Etiqueta de flujo* proporciona el medio para que un origen y un destino marquen grupos de paquetes que tengan los mismos requerimientos y que la red deba tratar de la misma forma, para formar una pseudoconexión. Por ejemplo, un flujo de paquetes de un proceso en cierto host de origen dirigido a cierto proceso en un host de destino puede tener requisitos muy estrictos de retardo y, por lo tanto, necesitar un ancho de banda reservado. El flujo se puede establecer por adelantado, dándole un identificador. Cuando aparece un paquete con una *Etiqueta de flujo* diferente de cero, todos los enrutadores pueden buscarla en sus tablas internas para ver el tipo de tratamiento especial que requiere. En efecto, los flujos son un intento de tener lo mejor de ambos mundos: la flexibilidad de una red de datagramas y las garantías de una red de circuitos virtuales.

Para fines de la calidad del servicio, cada flujo está designado con base en la dirección de origen, la dirección de destino y el número de flujo. Este diseño significa que pueden estar activos hasta 220 flujos al mismo tiempo entre un par dado de direcciones IP. También significa que, incluso si dos flujos provenientes de hosts diferentes pero con la misma etiqueta de flujo pasan por el mismo enrutador, éste será capaz de distinguirlos mediante las direcciones de origen y de destino. Lo ideal es que las etiquetas de flujo se escojan al azar, en vez de asignarlas de manera secuencial comenzando por el 1, de modo que los enrutadores tengan que usar *hashing* (conversión de valores mediante una función determinada).

El campo *Longitud de carga útil* indica cuántos bytes van después del encabezado de 40 bytes de la Ilustración 54. El nombre se cambió de *Longitud total* en el IPv4 porque el significado cambió ligeramente: los 40 bytes del encabezado ya no se cuentan como parte de la longitud (como antes). Este cambio significa que ahora la carga útil puede ser de 65 535 bytes en vez de sólo 65 515 bytes.

El campo *Siguiente encabezado* revela el secreto. La razón por la que pudo simplificarse el encabezado es que puede haber encabezados adicionales (opcionales) de extensión. Este campo indica cuál de los seis encabezados de extensión (en la actualidad), siguen de éste, en caso de que los haya. Si este encabezado es el último encabezado de IP, el campo *Siguiente encabezado* indica el protocolo de transporte (por ejemplo, TCP, UDP) al que se entregará el paquete.

El campo *Límite de saltos* se usa para evitar que los paquetes vivan eternamente. En la práctica es igual al campo *Tiempo de vida* del IPv4; esto es, un campo que se disminuye en cada salto. En teoría, en el IPv4 era un tiempo en segundos, pero ningún enrutador lo usaba de esa manera, por lo que se cambió el nombre para reflejar la manera en que se usa realmente.

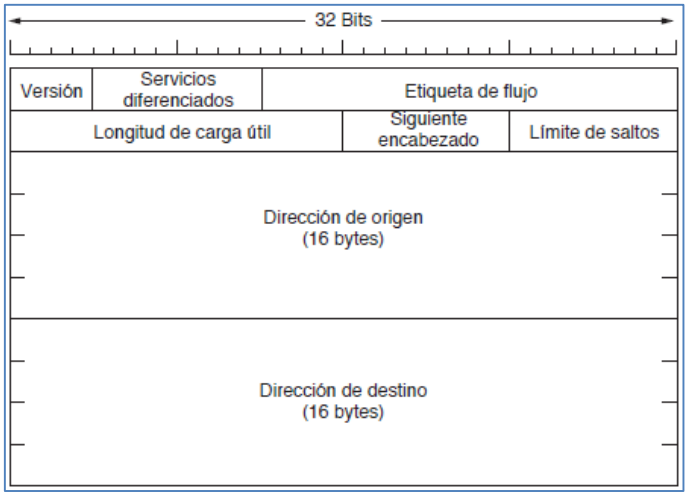


Ilustración 54 - El encabezado fijo de IPv6 (requerido).

Luego vienen los campos *Dirección de origen* y *Dirección de destino*. La propuesta original de Deering, SIP, usaba direcciones de 8 bytes pero durante el periodo de revisión muchas personas sintieron que, con direcciones de 8 bytes, en unas cuantas décadas el IPv6 se quedaría sin direcciones, mientras que con direcciones de 16 bytes nunca se agotarían. Otros argumentaban que 16 bytes era demasiado, mientras que unos más estaban a favor de usar direcciones de 20 bytes para hacerlas compatibles con el protocolo de datagramas de OSI. Para colmo, otro grupo quería direcciones de tamaño variable. Tras mucho debate se decidió que la mejor medida sería una dirección de 16 bytes de longitud fija.

Se ha desarrollado una nueva notación para escribir direcciones de 16 bytes, las cuales se escriben como ocho grupos de cuatro dígitos hexadecimales, separando los grupos por dos puntos, como el siguiente ejemplo:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Ya que muchas direcciones tendrán muchos ceros en ellas, se han autorizado tres optimizaciones. Primero, se pueden omitir los ceros a la izquierda dentro de un grupo, por lo que es posible escribir 0123 como 123. Segundo, se pueden reemplazar uno o más grupos de 16 bits cero por un par de signos de dos puntos. Por lo tanto, la dirección anterior se vuelve ahora

8000::123:4567:89AB:CDEF

Por último, las direcciones IPv4 se pueden escribir como un par de signos de dos puntos y un número decimal anterior separado por puntos, por ejemplo:

::192.31.20.46

Tal vez no sea necesario ser tan explícitos al respecto, pero hay muchas direcciones de 16 bytes. Específicamente hay 2^{128} de ellas, lo que aproximadamente equivale a $3x10^{38}$. Si la Tierra completa, incluidos los océanos, estuviera cubierta de computadoras, el IPv6 permitiría $7x10^{23}$ direcciones IP por metro cuadrado.

En la práctica, el espacio de direcciones no se usará de manera eficiente, justo igual que el espacio de direcciones de números telefónicos. En el RFC 3194, Durand y Huitema calcularon que, si se usa la asignación de números telefónicos como guía, hasta en la situación más pesimista habrá más de 1000 direcciones IP por metro cuadrado de la superficie terrestre (tierra y agua). En cualquier situación probable, habrá miles de millones de ellas por metro cuadrado. En pocas palabras, parece poco probable que se vayan a agotar en el futuro previsible.

Para fines instructivos, podemos comparar el encabezado de IPv4 con el de IPv6 para ver lo que se ha dejado fuera del IPv6. El campo *IHL* se fue porque el encabezado de IPv6 tiene una longitud fija. El campo *Protocolo* se retiró porque el campo *Siguiente encabezado* indica lo que sigue después del último encabezado IP (por ejemplo, un segmento UDP o TCP).

Se eliminaron todos los campos relacionados con la fragmentación, puesto que el IPv6 tiene un enfoque distinto en cuanto a la fragmentación. Para empezar, todos los hosts que cumplen con el IPv6 deben determinar en forma dinámica el tamaño de paquete a usar. Para ello utilizan el procedimiento de descubrimiento de MTU de la ruta descrito anteriormente. En resumen, cuando un host envía un paquete IPv6 demasiado grande, en vez de fragmentarlo, el enrutador que no puede reenviarlo descarta el paquete y envía un mensaje de error de vuelta al host emisor. Este mensaje indica al host que divida todos los paquetes futuros para ese destino. Hacer que el host envíe paquetes del tamaño correcto en primer lugar es, en última instancia, mucho más eficiente que hacer que los enrutadores los fragmenten sobre la marcha. Asimismo, el tamaño mínimo de paquete se incrementó de 576 a 1280 bytes para permitir 1024 bytes de datos y muchos encabezados.

Por último, el campo *Suma de verificación* desapareció porque al calcularlo se reduce en gran medida el desempeño. Con las redes confiables de hoy, además del hecho de que la capa de enlace de datos y las capas de transporte por lo general tienen sus propias sumas de verificación, la ventaja de tener otra suma de verificación no valía el costo de desempeño que generaba. Al quitar estas características, el resultado es un protocolo de capa de red compacto y sólido. Por lo tanto, con este diseño se cumplió la meta del IPv6 (un protocolo rápido y flexible con bastante espacio de direcciones).

Encabezados de extensión

Como de todas formas ocasionalmente se requieren algunos de los campos faltantes del IPv4, el IPv6 introdujo el concepto de **encabezados de extensión** (opcionales). Estos encabezados se pueden usar para proporcionar información adicional, pero codificada de una manera eficiente. Hay seis tipos de encabezados de extensión definidos en la actualidad, los cuales se listan en la Ilustración 55. Todos son opcionales, pero si hay más de uno presente, deben aparecer justo después del encabezado fijo y de preferencia en el orden listado.

Encabezado de extensión	Descripción
Opciones salto por salto.	Información diversa para los enrutadores.
Opciones de destino.	Información adicional para el destino.
Enrutamiento.	Lista informal de los enrutadores a visitar.
Fragmentación.	Manejo de fragmentos de datagramas.
Autentificación.	Verificación de la identidad del emisor.
Carga útil de seguridad cifrada.	Información sobre el contenido cifrado.

Ilustración 55 - Encabezados de extensión de IPv6.

Algunos de los encabezados tienen un formato fijo; otros contienen un número variable de opciones de longitud variable. En éstos, cada elemento está codificado como una tupla (Tipo, Longitud, Valor). El *Tipo* es un campo de 1 byte que indica la opción de la que se trata. Los valores de *Tipo* se han escogido de modo que los 2 primeros bits indiquen a los enrutadores, que no saben cómo procesar la opción, lo que tienen que hacer. Las posibilidades son: omitir la opción, descartar el paquete, descartar el paquete y enviar de vuelta un paquete ICMP, y descartar el paquete pero no enviar paquetes ICMP para direcciones de multidifusión (para evitar que un paquete de multidifusión malo genere millones de informes ICMP).

La *Longitud* también es un campo de 1 byte, e indica la longitud del valor (0 a 255 bytes). El *Valor* es cualquier información requerida de hasta 255 bytes.

El **encabezado de salto por salto** se usa para la información que deben examinar todos los enrutadores a lo largo de la ruta. Hasta ahora, se ha definido una opción: manejo de datagramas de más de 64K. El formato de este encabezado se muestra en la Ilustración 56. Cuando se utiliza, el campo *Longitud de carga útil* del encabezado fijo se establece a 0.

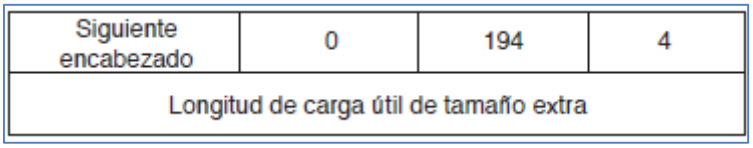


Ilustración 56 - El encabezado de extensión salto por salto para datagramas grandes (jumbogramas).

Como todos los encabezados de extensión, éste comienza con un byte que indica el tipo de encabezado que sigue. A este byte le sigue uno que indica la longitud del encabezado salto por salto en bytes, excluyendo los primeros 8 bytes, que son obligatorios. Todas las extensiones empiezan de esta manera.

Los siguientes 2 bytes indican que esta opción define el tamaño del datagrama (código 194) y que el tamaño es un número de 4 bytes. Los últimos 4 bytes indican el tamaño del datagrama. No se permiten tamaños menores a 65536 bytes, pues de lo contrario el primer enrutador descartará el paquete y devolverá un mensaje ICMP de error. Los datagramas que usan este encabezado de extensión se llaman *jumbogramas*. El uso de *jumbogramas* es importante para las aplicaciones de supercomputadoras que deben transferir con eficiencia gigabytes de datos a través de Internet.

El **encabezado de opciones de destino** está reservado para campos que sólo se necesitan interpretar en el host de destino. En la versión inicial de IPv6, las únicas opciones definidas son las opciones nulas para rellenar este encabezado a un múltiplo de 8 bytes, por lo que en un principio no se usará. Se incluyó para asegurarse que el nuevo enrutamiento y el software del host pudieran manejarlo, en caso de que algún día alguien pensara en una opción de destino.

El **encabezado de enrutamiento** lista uno o más enrutadores que deben visitarse en el camino hacia el destino. Es muy similar al *enrutamiento de origen libre* del IPv4 en cuanto a que todas las direcciones listadas se deben visitar en orden, pero también se pueden visitar otros enrutadores no listados que se encuentren dentro del trayecto. El formato del encabezado de enrutamiento se muestra en la Ilustración 57.

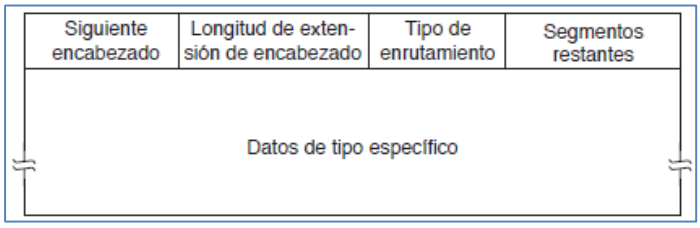


Ilustración 57 - El encabezado de extensión para enrutamiento.

Los primeros 4 bytes del encabezado de extensión de enrutamiento contienen cuatro enteros de 1 byte. Anteriormente describimos los campos *Siguiente encabezado* y *Longitud de extensión del encabezado*. El campo *Tipo de enrutamiento* proporciona el formato del resto del encabezado. El tipo 0 indica que una palabra reservada de 32 bits va después de la primera palabra, seguida por cierto número de direcciones de IPv6. Pueden inventarse otros tipos en el futuro según se necesite. Por último, el campo *Segmentos restantes* registra cuántas direcciones de la lista no se han visitado todavía. Se decrementa cada vez que se visita una de ellas. Cuando llega a 0, el paquete está por su cuenta, sin más guía sobre qué ruta seguir. Por lo general, a estas alturas está tan cerca del destino que la mejor ruta es obvia.

El **encabezado de fragmentación** maneja ésta de una manera parecida a la del IPv4. El encabezado contiene el identificador del datagrama, el número de fragmento y un bit que indica si seguirán más fragmentos. A diferencia del IPv4, en el IPv6 sólo el host de origen puede fragmentar un paquete. Los enrutadores a lo largo del camino no pueden hacerlo. Este cambio representa un rompimiento filosófico con el IP original, aunque se mantiene a la par con la práctica habitual del IPv4. Además, simplifica el trabajo del enrutador y acelera el proceso de enrutamiento. Como se mencionó antes, si un enrutador confronta un paquete demasiado grande, lo descarta y devuelve un paquete de error ICMP al origen. Esta información permite que el host de origen fragmente el paquete en piezas más pequeñas mediante el uso de este encabezado y lo intente de nuevo.

El **encabezado de autentificación** proporciona un mecanismo mediante el cual el receptor de un paquete puede estar seguro de quién lo envió. La **carga útil de seguridad de cifrado** hace posible cifrar el contenido de

un paquete, de modo que sólo el receptor pretendido pueda leerlo. Estos encabezados usan las técnicas criptográficas que describiremos en las próximas unidades.

Protocolos de control en Internet

Además el IP que se utiliza para la transferencia de datos, Internet tiene varios protocolos de control complementarios que se utilizan en la capa de red, como ICMP, ARP y DHCP. En esta sección los analizaremos por separado y describiremos las versiones que corresponden a IPv4, ya que son los protocolos de uso común. ICMP y DHCP tienen versiones similares para IPv6; el equivalente de ARP se llama **NDP** (Protocolo de Descubrimiento de Vecino, del inglés *Neighbor Discovery Protocol*) para IPv6.

ICMP: Protocolo de Mensajes de Control en Internet

Los enrutadores supervisan muy de cerca el funcionamiento de Internet. Cuando ocurre algo inesperado durante el procesamiento de un paquete en un enrutador, **ICMP** (Protocolo de Mensajes de Control en Internet, del inglés *Internet Control Message Protocol*) informa sobre el evento al emisor. ICMP también se utiliza para probar Internet. Hay definidos alrededor de una docena de tipos de mensajes ICMP, cada uno de los cuales se transporta encapsulado en un paquete IP. Los más importantes se listan en la Ilustración 58.

El mensaje *DESTINATION UNREACHABLE* (DESTINO INACCESIBLE) se usa cuando el enrutador no puede localizar el destino o cuando un paquete con el bit *DF* no puede entregarse porque hay una red de "paquetes pequeños" que se interpone en el camino.

Tipo de mensaje	Descripción
Destination unreachable (Destino inaccesible).	No se pudo entregar el paquete.
Time exceeded (Tiempo excedido).	El tiempo de vida llegó a cero.
Parameter problem (Problema de parámetros).	Campo de encabezado inválido.
Source quench (Fuente disminuida).	Paquete regulador.
Redirect (Redireccionar).	Enseña a un enrutador la geografía.
Echo and echo reply (Eco y respuesta de eco).	Verifica si una máquina está viva.
Timestamp request/reply (Estampa de tiempo, Petición/respuesta).	Igual que solicitud de eco, pero con marca de tiempo.
Router advertisement/solicitation (Enrutamiento anuncio/solicitud).	Busca un enrutador cercano.

Ilustración 58 - Los principales tipos de mensajes ICMP.

El mensaje *TIME EXCEEDED* (TIEMPO EXCEDIDO) se envía al descartar un paquete porque su contador *Ttl* (Tiempo de vida) ha llegado a cero. Este evento es un síntoma de que los paquetes se están repitiendo o que los valores establecidos en el contador son muy bajos.

Un uso inteligente de este mensaje de error es la herramienta *traceroute* desarrollada por Van Jacobson en 1987. *Traceroute* encuentra los enrutadores a lo largo de la ruta desde el host hasta una dirección IP de destino. Encuentra esta información sin ningún tipo de soporte de red privilegiado. Este método simplemente envía una secuencia de paquetes al destino, primero con un *TtL* de 1, después con un *TtL* de 2, 3 y así en lo sucesivo. Los contadores en estos paquetes llegarán a cero en los enrutadores sucesivos a lo largo de la ruta. Cada uno de estos enrutadores enviará obedientemente un mensaje *TIME EXCEEDED* de vuelta al host. A partir de estos mensajes el host puede determinar las direcciones IP de los enrutadores a lo largo de la ruta, así como mantener estadísticas y tiempos en ciertas partes de la ruta.

El mensaje *PARAMETER PROBLEM* (PROBLEMAS DE PARÁMETROS) indica que se ha descubierto un valor ilegal en un campo de encabezado. Este problema indica un error en el software de IP del host emisor, o tal vez en el software de un enrutador por el que se transita.

El mensaje *SOURCE QUENCH* (FUENTE DISMINUIDA) se utilizaba hace tiempo para regular a los hosts que estaban enviando demasiados paquetes. Se esperaba que cuando un host recibiera este mensaje, redujera la velocidad. En la actualidad raras veces se usa pues cuando ocurre una congestión, estos paquetes tienden a agravar más la situación y no está claro cómo responderles. Ahora, el control de congestión en Internet se

hace sobre todo en la capa de transporte, en donde se utilizan las pérdidas de paquetes como señales de congestión.

El mensaje *REDIRECT* (REDIRECCIONAR) se usa cuando un enrutador se percata de que un paquete parece estar mal enrutado. Lo utiliza el enrutador para avisar al host emisor que se actualice con una mejor ruta.

Los mensajes *ECHO* (ECO) y *ECHO REPLY* (RESPUESTA DE ECO) se utilizan para ver si un destino dado es alcanzable y está vivo. Se espera que el destino envíe de vuelta un mensaje *ECHO REPLY* luego de recibir el mensaje *ECHO*. Estos mensajes se utilizan en la herramienta ping que verifica si un host está activo en Internet.

Los mensajes *TIMESTAMP REQUEST* (PETICIÓN DE ESTAMPA DE TIEMPO) y *TIMESTAMP REPLY* (RESPUESTA DE ESTAMPA DE TIEMPO) son similares, excepto que el tiempo de llegada del mensaje y el tiempo de salida de la respuesta se registran en ésta. Esta característica se puede usar para medir el desempeño de la red.

Los mensajes *ROUTER ADVERTISEMENT* (ANUNCIO DE ENRUTADOR) y *ROUTER SOLICITATION* (SOLICITUD DE ENRUTADOR) se usan para permitir que los hosts encuentren los enrutadores cercanos. Un host necesita aprender la dirección IP de por lo menos un enrutador para enviar paquetes por la red local.

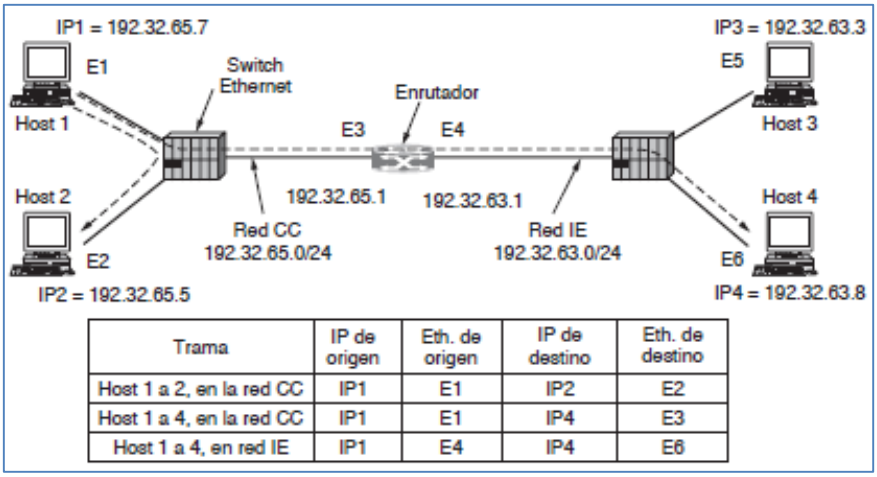
Además de estos mensajes, se han definido otros. La lista en línea se conserva ahora en www.iana.org/assignments/icmp-parameters.

ARP: Protocolo de Resolución de Direcciones

Aunque en Internet cada máquina tiene una o más direcciones IP, en realidad éstas no son suficientes para enviar paquetes. Las NIC (Tarjetas de Interfaz de Red) de la capa de enlace de datos no entienden las direcciones de Internet. En el caso de Ethernet, cada NIC de las que se hayan fabricado viene equipada con una dirección Ethernet única de 48 bits. Los fabricantes de NIC Ethernet solicitan un bloque de direcciones Ethernet al IEEE para asegurar que no haya dos NIC con la misma dirección (y evitar conflictos en caso de que las dos NIC aparezcan alguna vez en la misma LAN). Las NIC envían y reciben tramas basadas en direcciones Ethernet de 48 bits. No saben nada sobre direcciones IP de 32 bits.

La pregunta ahora es: ¿cómo se convierten las direcciones IP en direcciones de la capa de enlace de datos, como Ethernet? Para explicar cómo funciona esto, veamos el ejemplo de la Ilustración 59 en donde se muestra una universidad pequeña con dos redes /24. Una red (CC) es una Ethernet conmutada y está en el Departamento de Ciencias Computacionales. Tiene el prefijo 192.32.65.0/24. La otra LAN (IE), que también es Ethernet conmutada, está en el Departamento de Ingeniería Eléctrica y tiene el prefijo 192.32.63.0/24. Las dos LAN están conectadas por un enrutador IP. Cada máquina en una Ethernet y cada interfaz en el enrutador tienen una dirección única de Ethernet, etiquetadas de E1 a E6, además de una dirección IP única en la red CC o IE.

Empecemos por ver cómo un usuario en el host 1 envía un paquete a un usuario en el host 2 de la red CC. Supongamos que el emisor sabe el nombre del receptor pretendido, posiblemente algo así como eagle.cc.uni.edu. El primer paso es encontrar la dirección IP para el host 2. Esta consulta es realizada por el DNS, que estudiaremos en el capítulo 7. Por el momento supongamos que el DNS devuelve la dirección IP del host 2 (192.32.65.5).



Trama	IP de origen	Eth. de origen	IP de destino	Eth. de destino
Host 1 a 2, en la red CC	IP1	E1	IP2	E2
Host 1 a 4, en la red CC	IP1	E1	IP4	E3
Host 1 a 4, en red IE	IP1	E4	IP4	E6

Ilustración 59 - Dos redes LAN Ethernet conmutadas, unidas por un enrutador.

El software de la capa superior en el host 1 elabora ahora un paquete con 192.32.65.5 en el campo Dirección de destino y lo entrega al software de IP para que lo transmita. El software IP puede buscar la dirección y ver que el destino está en la red CC (es decir, su propia red). Pero de todas formas necesita alguna manera de encontrar la dirección Ethernet de destino para enviar la trama. Una solución es tener un archivo de configuración en alguna parte del sistema que asocie las direcciones IP con direcciones Ethernet. Aun cuando esta solución es ciertamente posible, para las organizaciones con miles de máquinas, conservar todos estos archivos actualizados es una tarea propensa a errores y consume mucho tiempo.

Una mejor solución es que el host 1 envíe un paquete de difusión hacia Ethernet y pregunte quién posee la dirección IP 192.32.65.5. La difusión llegará a cada máquina en la Ethernet CC y cada una verificará su dirección IP. Al host 2 le bastará responder con su dirección de Ethernet (E2). De esta manera, el host 1 aprende que la dirección IP 192.32.65.5 está en el host con la dirección Ethernet E2. El protocolo utilizado para hacer esta pregunta y obtener la respuesta se llama **ARP** (Protocolo de Resolución de Direcciones, del inglés *Address Resolution Protocol*). Casi todas las máquinas en Internet lo ejecutan. La definición de ARP está en el RFC 826.

La ventaja de usar ARP en lugar de archivos de configuración es su simpleza. El administrador del sistema sólo tiene que asignar a cada máquina una dirección IP y decidir respecto a las máscaras de subred. ARP se hace cargo del resto.

A estas alturas, el software IP en el host 1 crea una trama Ethernet dirigida a E2, pone el paquete IP (dirigido a 192.32.65.5) en el campo de carga útil y lo descarga hacia la Ethernet. Las direcciones IP y Ethernet de este paquete se muestran en la Ilustración 59. La NIC Ethernet del host 2 detecta esta trama, la reconoce como una trama para sí mismo, la recoge y provoca una interrupción. El controlador de Ethernet extrae el paquete IP de la carga útil y lo pasa al software IP, el cual ve que esté direccionado de forma correcta y lo procesa.

Es posible hacer varias optimizaciones para que ARP trabaje con más eficiencia. Para empezar, una vez que una máquina ha ejecutado ARP, guarda el resultado en caché, en caso de que tenga que ponerse en contacto con la misma máquina en poco tiempo. La siguiente vez encontrará la asociación en su propio caché, con lo cual se elimina la necesidad de una segunda difusión. En muchos casos, el host 2 necesitará devolver una respuesta y se verá forzado también a ejecutar el ARP para determinar la dirección Ethernet del emisor. Podemos evitar esta difusión de ARP haciendo que el host 1 incluya su asociación IP a Ethernet en el paquete ARP. Cuando la difusión de ARP llega al host 2, se introduce el par (192.32.65.7, E1) en la caché ARP del host 2. De hecho, todas las máquinas en Ethernet pueden introducir esta asociación en su caché ARP.

Para permitir que las asociaciones cambien, por ejemplo, al configurar un host para que use una nueva dirección IP (pero que mantenga su vieja dirección Ethernet), las entradas en la caché ARP deben expirar después de unos cuantos minutos. Una manera inteligente de ayudar a mantener actualizada la información en la caché y optimizar el desempeño es hacer que cada máquina difunda su asociación cuando se configure. Por lo general, esta difusión se realiza en forma de un ARP que busca su propia dirección IP. No debe haber una respuesta, pero un efecto colateral de la difusión es crear o actualizar una entrada en la caché ARP de

todos. A esto se le conoce como **ARP gratuito**. Si una respuesta llega (en forma inesperada), quiere decir que se asignó la misma dirección IP a dos máquinas. El administrador de la red debe resolver este error antes de que ambas máquinas puedan usarla.

Ahora veamos de nuevo la Ilustración 59, sólo que esta vez el host 1 quiere enviar un paquete al host 4 (192.32.63.8) en la red IE. El host 1 verá que la dirección IP de destino no está en la red CC. Sabe enviar todo ese tráfico fuera de la red al enrutador, el cual también se conoce como **puerta de enlace predeterminada**. Por convención, la puerta de enlace predeterminada es la dirección más baja en la red (198.31.65.1). Para enviar una trama al enrutador, el host 1 debe conocer de todas formas la dirección Ethernet de la interfaz del enrutador en la red CC. Para descubrirla envía una difusión ARP para 198.31.65.1, a partir de la cual aprende E3. Después envía la trama. Los mismos mecanismos de búsqueda se utilizan para enviar un paquete de un enrutador al siguiente, a través de una secuencia de enrutadores en una ruta de Internet.

Cuando la NIC Ethernet del enrutador recibe esta trama, entrega el paquete al software IP. Sabe con base en las máscaras de red que el paquete se debe enviar a la red IE, en donde alcanzará al host 4. Si el enrutador no conoce la dirección Ethernet para el host 4, entonces usará ARP de nuevo. La tabla en la Ilustración 59 lista las direcciones Ethernet e IP de origen y destino que están presentes en las tramas, como se observa en las redes CC e IE. Observe que las direcciones Ethernet cambian con la trama en cada red, mientras que las direcciones IP permanecen constantes (puesto que indican las terminales a través de todas las redes interconectadas).

También es posible enviar un paquete del host 1 al host 4 sin que el primero sepa que el segundo está en una red diferente. La solución es hacer que el enrutador responda a los ARP en la red CC para el host 4 y proporcione su dirección Ethernet, E3, como respuesta. No es posible hacer que el host 4 responda directamente, ya que no verá la solicitud ARP (puesto que los enrutadores no reenvían difusiones a nivel de Ethernet). A continuación, el enrutador recibirá las tramas enviadas a 192.32.63.8 y las reenviará a la red IE. A esta solución se le conoce como **ARP por proxy** y se utiliza en casos especiales en los que un host desea aparecer en una red, aun cuando en realidad reside en otra. Por ejemplo, una situación común es una computadora móvil que desea que algún otro nodo recoja los paquetes por ella cuando no se encuentre en su red local.

DHCP: el Protocolo de Configuración Dinámica de Host

ARP (así como los demás protocolos de Internet) asume que los hosts están configurados con cierta información básica, como sus propias direcciones IP. ¿Cómo obtienen los hosts esta información? Es posible configurar en forma manual cada computadora, pero es un proceso tedioso y propenso a errores. Existe una mejor manera de hacerlo, conocida como **DHCP** (Protocolo de Configuración Dinámica de Host, del inglés *Dynamic Host Configuration Protocol*).

Con DHCP, cada red debe tener un servidor DHCP responsable de la configuración. Al iniciar una computadora, ésta tiene integrada una dirección Ethernet u otro tipo de dirección de capa de enlace de datos en la NIC, pero no cuenta con una dirección IP. En forma muy parecida al ARP, la computadora difunde una solicitud de una dirección IP en su red. Para ello usa un paquete llamado *DHCP DISCOVER*. Este paquete debe llegar al servidor DHCP. Si el servidor no está conectado directamente a la red, el enrutador se configurará para recibir difusiones DHCP y transmitirlas al servidor DHCP en donde quiera que se encuentre. Cuando el servidor recibe la solicitud, asigna una dirección IP libre y la envía al host en un paquete *DHCP OFFER* (que también se puede transmitir por medio del enrutador). Para que esto pueda funcionar incluso cuando los hosts no tienen direcciones IP, el servidor identifica a un host mediante su dirección Ethernet (la cual se transporta en el paquete *DHCP DISCOVER*).

Un problema que surge con la asignación automática de direcciones IP de una reserva es determinar qué tanto tiempo se debe asignar una dirección IP. Si un host sale de la red y no devuelve su dirección IP al servidor DHCP, esa dirección se perderá en forma permanente. Después de un tiempo, tal vez se pierdan muchas direcciones. Para evitar que eso ocurra, la asignación de direcciones IP puede ser por un periodo fijo de tiempo, una técnica conocida como **arrendamiento**. Justo antes de que expire el arrendamiento, el host debe pedir una renovación al DHCP. Si no puede hacer una solicitud o si ésta se rechaza, tal vez el host ya no pueda usar la dirección IP que recibió antes.

El DHCP se describe en los RFC 2131 y 2132. Se utiliza ampliamente en Internet para configurar todo tipo de parámetros, además de proporcionar a los hosts direcciones IP. Al igual que en las redes de negocios y domésticas, los ISP usan DHCP para establecer los parámetros de los dispositivos a través del enlace de acceso a Internet, de modo que los clientes no tengan que comunicarse por teléfono con sus ISP para obtener esta información. Algunos ejemplos comunes de la información que se configura incluyen la máscara de red, la dirección IP de la puerta de enlace predeterminada y las direcciones IP de los servidores DNS y de tiempo. DHCP ha reemplazado en gran parte los protocolos anteriores (conocidos como RARP y BOOTP), con una funcionalidad más limitada.

Conmutación mediante etiquetas y MPLS

Hasta ahora nos hemos enfocado exclusivamente en los paquetes como datagramas que los enrutadores IP reenvían. También hay otro tipo de tecnología que está empezando a tener mucho auge, en especial entre los ISP, para desplazar el tráfico de Internet entre sus redes. A esta tecnología se le conoce como **MPLS** (Conmutación Multiprotocolo Mediante Etiquetas, del inglés *MultiProtocol Label Switching*) y es cercana a la tecnología de conmutación de circuitos. A pesar del hecho de que muchas personas en la comunidad de Internet tienen una intensa aversión por las redes orientadas a conexión, la idea parece estar regresando. Sin embargo, existen diferencias esenciales entre la forma en que Internet maneja la construcción de rutas y la manera en que lo hacen las redes orientadas a conexión, por lo que sin duda la técnica no es una conmutación de circuitos tradicional.

MPLS agrega una etiqueta en frente de cada paquete; el reenvío se basa en la etiqueta en vez de la dirección de destino. Al convertir la etiqueta en un índice de una tabla interna, sólo es cuestión de buscar en la tabla la línea de salida correcta. Mediante el uso de esta técnica, el reenvío se puede llevar a cabo con mucha rapidez. Esta ventaja fue la motivación original detrás de MPLS, que empezó como una tecnología propietaria conocida por varios nombres, incluyendo **conmutación por marcas** (*tag switching*). En un momento dado, la IETF empezó a estandarizar la idea. Se describe en el RFC 3031 y en muchos otros RFC. Con el tiempo, los principales beneficios han sido un enrutamiento flexible y un reenvío adecuado para la calidad del servicio, así como rápido.

La primera pregunta que surge es en dónde colocar la etiqueta. Como los paquetes IP no se diseñaron para circuitos virtuales, no hay un campo disponible para números de circuito virtual dentro del encabezado IP. Por esta razón se tuvo que agregar un nuevo encabezado MPLS enfrente del encabezado IP. En una línea de enrutador a enrutador que utiliza PPP como protocolo de entramado, el formato de trama, incluyendo los encabezados PPP, MPLS, IP y TCP, es como se muestra en la Ilustración 60.

El encabezado MPLS genérico tiene una longitud de 4 bytes y cuatro campos. El más importante es el campo *Etiqueta* que contiene el índice. El campo *QoS* indica la clase de servicio. El campo *S* se relaciona con el apilamiento de múltiples etiquetas (lo que veremos a continuación). El campo *TtL* indica cuántas veces se puede reenviar el paquete. Se decrementa en cada enrutador y, si llega a 0, se descarta el paquete. Esta característica evita los ciclos infinitos en caso de inestabilidad del enrutamiento.

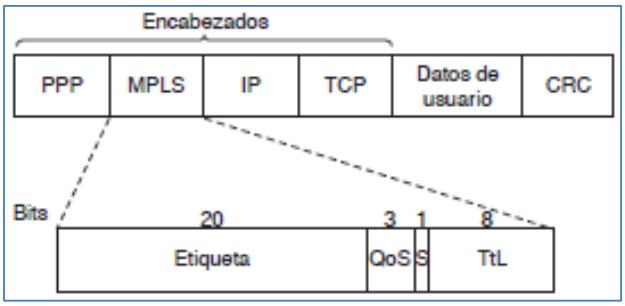


Ilustración 60 - Transmisión de un segmento TCP mediante IP, MPLS y PPP.

MPLS se encuentra entre el protocolo IP de la capa de red y el protocolo PPP de la capa de enlace. En realidad no es un protocolo de capa 3, pues depende de direcciones IP o de otra capa de red para establecer las rutas de las etiquetas. En realidad tampoco es un protocolo de capa 2, pues reenvía los paquetes a través de varios saltos, no de un solo enlace. Por esta razón, algunas veces se denomina protocolo de capa 2.5. Es una

ilustración de que los protocolos reales no siempre se ajustan perfectamente a nuestro modelo ideal de protocolos por capas.

Por el lado optimista, como los encabezados MPLS no son parte del paquete de capa de red ni de la trama de capa de datos, MPLS es en gran parte independiente de ambas capas. Entre otras cosas, esta propiedad significa que es posible construir switches MPLS que puedan reenviar tanto paquetes IP como paquetes que no sean IP, dependiendo de lo que se presente. Esta característica es la razón del término "*multiprotocolo*" en el nombre de MPLS. También puede transportar paquetes IP sobre redes que no sean IP.

Cuando llega un paquete con capacidad para MPLS a un **LSR** (Enrutador de Conmutación de Etiquetas, del inglés *Label Switched Router*), la etiqueta se utiliza como un índice en una tabla para determinar la línea de salida y la nueva etiqueta a utilizar. Esta conmutación de etiquetas se utiliza en todas las redes de circuitos virtuales. Las etiquetas sólo tienen importancia local y dos enrutadores diferentes pueden transmitir paquetes no relacionados que contengan la misma etiqueta hacia otro enrutador para transmitirlos en la misma línea de salida. Para diferenciarlos en el otro extremo, las etiquetas se tienen que volver a asociar en cada salto. En la Ilustración 3 vimos este mecanismo en acción. MPLS utiliza la misma técnica.

Como observación adicional, algunas personas hacen la distinción entre reenvío y conmutación. El reenvío es el proceso de encontrar la mejor coincidencia para una dirección de destino en una tabla y decidir por dónde enviar los paquetes. Un ejemplo es el algoritmo del prefijo más largo coincidente que se utiliza para el reenvío IP. En contraste, la conmutación usa una etiqueta que se toma del paquete como un índice hacia una tabla de reenvío. Es más simple y más rápido. Sin embargo, estas definiciones están lejos de ser universales.

Como la mayoría de los hosts y enrutadores no entienden MPLS, también deberíamos preguntar cómo y cuándo se adjuntan las etiquetas a los paquetes. Esto ocurre cuando un paquete IP llega al extremo de una red MPLS. El **LER** (Enrutador de Etiquetas de Borde, del inglés *Label Edge Router*) inspecciona la dirección IP de destino y otros campos para ver qué ruta MPLS debe seguir el paquete, y coloca la etiqueta correcta al frente del paquete. Dentro de la red MPLS, esta etiqueta se utiliza para reenviar el paquete. En el otro extremo de la red MPLS, la etiqueta ha cumplido su propósito y se elimina para revelar el paquete IP a la siguiente red. Este proceso se muestra en la Ilustración 61. Una diferencia de los circuitos virtuales tradicionales es el nivel de agregación. Sin duda es posible que cada flujo tenga su propio conjunto de etiquetas a través de la red MPLS. Sin embargo, es más común que los enrutadores agrupen varios flujos que terminan en un enrutador o LAN en particular, y que usen una sola etiqueta para ellos. Se dice que los flujos que se agrupan bajo una sola etiqueta pertenecen a la misma **FEC** (Clase de Equivalencia de Reenvío, del inglés *Forwarding Equivalence Class*). Esta clase no sólo cubre hacia dónde van los paquetes, sino también su clase de servicio (en el sentido de los servicios diferenciados), ya que todos los paquetes se tratan de la misma forma para fines de reenvío.

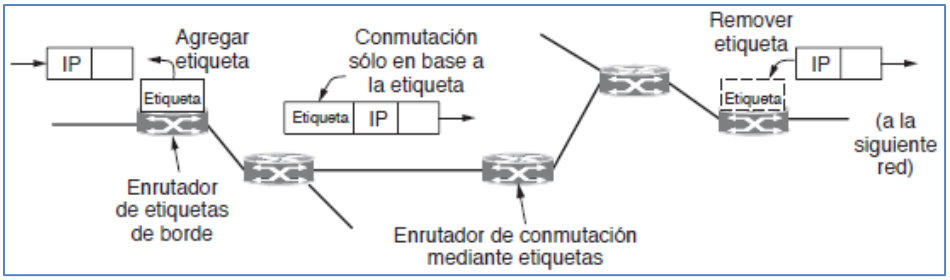


Ilustración 61 - Reenvío de un paquete IP por medio de una red MPLS.

En el enrutamiento tradicional de circuitos virtuales, no es posible agrupar varias rutas distintas con diferentes terminales en el mismo identificador de circuito virtual, ya que no habría forma de diferenciarlas en el destino final. Con MPLS, los paquetes aún contienen su dirección de destino final, además de la etiqueta. Al final de la ruta etiquetada, se puede eliminar el encabezado de etiqueta y el reenvío puede proseguir de la forma usual, mediante el uso de la dirección de destino de la capa de red.

En realidad, MPLS va aún más allá. Puede operar a múltiples niveles al mismo tiempo, para lo cual agrega más de una etiqueta al frente de un paquete. Por ejemplo, suponga que hay muchos paquetes que ya tienen distintas etiquetas (puesto que deseamos tratar a los paquetes de manera distinta en alguna parte de la red),

las cuales deben seguir una ruta común hacia cierto destino. En vez de establecer muchas rutas de conmutación, una para cada una de las distintas etiquetas, podemos establecer una sola ruta. Cuando los paquetes ya etiquetados llegan al inicio de esta ruta, se agrega otra etiqueta al frente. A esto se le conoce como **pila de etiquetas**. La etiqueta más externa guía a los paquetes a lo largo de la ruta. La etiqueta se elimina al final de la ruta y las etiquetas que se revelan, si las hay, se utilizan para reenviar el paquete el resto del camino. El bit *S* en la Ilustración 60 permite a un enrutador eliminar una etiqueta para saber si quedan etiquetas adicionales. Se establece en 1 para la etiqueta inferior y en 0 para todas las demás etiquetas.

La pregunta final por hacer es cómo se establecen las tablas de reenvío de etiquetas de manera que los paquetes las sigan. Ésta es un área de gran diferencia entre los diseños de MPLS y de los circuitos virtuales convencionales. En las redes tradicionales de circuitos virtuales, cuando un usuario desea establecer una conexión, se lanza un paquete de establecimiento por la red para crear la ruta y las entradas en la tabla de reenvío. MPLS no involucra a los usuarios en la fase de establecimiento. Exigir a los usuarios que hagan algo además de enviar un datagrama quebrantaría a la mayor parte del software existente en Internet.

En cambio, la información de reenvío se establece mediante protocolos que son una combinación de protocolos de enrutamiento y de establecimiento de conexión. Estos protocolos de control están totalmente separados del reenvío de etiquetas, lo cual nos permite usar múltiples protocolos de control distintos. Una de las variantes funciona así. Cuando se enciende un enrutador, éste revisa cuáles son las rutas para las que será el destino final (por ejemplo, qué prefijos pertenecen a sus interfaces). Después crea una o más FECs para estas rutas, asigna una etiqueta para cada una de ellas y pasa las etiquetas a sus vecinos. Éstos a su vez, introducen las etiquetas en sus tablas de reenvío y envían nuevas etiquetas a sus vecinos, hasta que todos los enrutadores hayan adquirido la ruta. También se pueden reservar recursos a medida que se construye la ruta para garantizar una calidad de servicio apropiada. Otras variantes pueden establecer distintas rutas, como las de ingeniería de tráfico que toman en cuenta la capacidad no utilizada, y crean rutas bajo demanda para soportar ofrecimientos de servicios tales como la calidad del servicio.

Aunque las ideas básicas detrás de MPLS son simples y directas, los detalles son complicados, con muchas variaciones y casos de uso en desarrollo constante.

OSPF: un protocolo de enrutamiento de puerta de enlace interior

Hemos terminado nuestro estudio acerca de cómo se reenvían los paquetes en Internet. Ya es tiempo para pasar al tema siguiente: el enrutamiento en Internet. Como lo mencionamos antes, Internet se compone de una gran cantidad de redes independientes o **Sistemas Autónomos** (AS), los cuales son operados por distintas organizaciones, por lo general una empresa, universidad o ISP. Dentro de su propia red, una organización puede usar su propio algoritmo de enrutamiento interno, o **enrutamiento intradominio**, como se le conoce con más frecuencia. Sin embargo, hay sólo unos cuantos protocolos estándar que son populares. En esta sección estudiaremos el problema del enrutamiento intradominio y analizaremos el protocolo OSPF, que se utiliza mucho en la práctica. Un protocolo de enrutamiento intradominio también se conoce como **protocolo de puerta de enlace interior**. En la siguiente sección, estudiaremos el problema de enrutamiento entre redes operadas de manera independiente, o **enrutamiento interdominio**. Para ese caso, todas las redes deben usar el mismo protocolo de enrutamiento interdominio, o **protocolo de puerta de enlace exterior**. El protocolo que se utiliza en internet es **BGP** (Protocolo de Puerta de Enlace de Frontera, del inglés *Border Gateway Protocol*).

Los primeros protocolos de enrutamiento intradominio usaban un diseño de vector de distancia basado en el algoritmo Bellman-Ford distribuido, que se heredó de ARPANET. **RIP** (Protocolo de Información de Enrutamiento, del inglés *Routing Information Protocol*) es el principal ejemplo que se usa en la actualidad. Funciona bien en sistemas pequeños, pero su desempeño disminuye a medida que las redes aumentan su tamaño. También sufre del problema del conteo al infinito y por lo general de una convergencia lenta. ARPANET cambió a un protocolo de estado del enlace en mayo de 1979 debido a estos problemas; en 1988, la IETF empezó a trabajar en un protocolo de estado del enlace para el enrutamiento intradominio. Ese protocolo, conocido como **OSPF** (Abrir primero la ruta más corta, del inglés *Open Shortest Path First*), se convirtió en un estándar en 1990. Se valió de un protocolo llamado **IS-IS** (Sistema Intermedio a Sistema Intermedio, del inglés *Intermediate-System to Intermediate-System*), el cual se convirtió en un estándar de

ISO. Debido a su herencia compartida, los dos protocolos son mucho más parecidos que diferentes. Si desea conocer la historia completa, consulte el RFC 2328. Son los protocolos de enrutamiento intradominio dominantes; la mayoría de los distribuidores ofrecen ahora soporte para ambos. OSPF se utiliza más en las redes de compañías; IS-IS se utiliza más en las redes de ISP. De los dos, veremos un bosquejo sobre la forma en que funciona OSPF.

Dada la extensa experiencia con otros protocolos de enrutamiento, el grupo diseñador de OSPF tenía una larga lista de requerimientos por cumplir. Primero, **había que publicar el algoritmo en la literatura abierta**, de aquí que se incluya una "O" en OSPF. No sería factible usar una solución propietaria perteneciente a una sola compañía. Segundo, el nuevo protocolo **tenía que soportar una variedad de métricas de distancia**, incluyendo la distancia física, el retardo, etc. Tercero, **tenía que ser un algoritmo dinámico**, uno que se adaptara a los cambios en la topología de manera automática y rápida.

Cuarto (y lo que constituía una novedad para OSPF), **tenía que soportar un enrutamiento con base en el tipo de servicio**. El nuevo protocolo tenía que ser capaz de enrutar el tráfico en tiempo real de una manera y el tráfico restante de una manera distinta. En ese tiempo, IP tenía un campo Tipo de servicio pero ningún protocolo existente lo usaba. Este campo se incluyó en OSPF pero de todas formas nadie lo usó, por lo que se eliminó eventualmente. Tal vez este requerimiento estaba adelantado a su tiempo, pues precedió al trabajo de la IETF sobre los servicios diferenciados, lo cual ha rejuvenecido las clases de servicio.

Quinto (y con relación a lo anterior), OSPF **tenía que balancear las carga para dividirla entre múltiples líneas**. Los protocolos anteriores enviaban todos los paquetes a través de una mejor ruta, incluso si había dos rutas igual de buenas. La otra ruta no se utilizaba para nada. En muchos casos, dividir la carga entre múltiples rutas produce un mejor desempeño.

Sexto, se necesitaba un **soporte para los sistemas jerárquicos**. Para 1988 algunas redes habían crecido tanto que no se podía esperar que un enrutador conociera toda la topología. Había que diseñar a OSPF de modo que ningún enrutador tuviera que hacerlo.

Séptimo, se **requería un mínimo de seguridad** para evitar que los estudiantes en busca de diversión se burlaran de los enrutadores al enviarles información de enrutamiento falsa. Por último, había que tomar las medidas necesarias para **lidiar con los enrutadores conectados a Internet por medio de un túnel**. Los protocolos anteriores no manejaban bien esto.

OSPF soporta los enlaces punto a punto (por ejemplo, SONET) y las redes de difusión (la mayoría de las LAN). En realidad, es capaz de soportar redes con múltiples enrutadores, cada uno de los cuales se puede comunicar en forma directa con los demás (a éstas se les conoce como **redes multiacceso**) incluso aunque no tengan capacidad de difusión. Los primeros protocolos no manejaban bien este caso.

En la Ilustración 62(a) se muestra un ejemplo de una red con un sistema autónomo. Se omiten los hosts debido a que en general no desempeñan ningún papel en OSPF, mientras que los enrutadores y las redes (que pueden contener hosts) sí. La mayoría de los enrutadores en la Ilustración 64(a) están conectados a otros enrutadores mediante enlaces punto a punto, y a redes para llegar a los hosts en ellas. Sin embargo, los enrutadores *R3*, *R4* y *R5* están conectados mediante una LAN de difusión tal como una red Ethernet conmutada.

Para operar, OSPF resume la colección de redes reales, enrutadores y enlaces en un grafo dirigido en el que a cada arco se le asigna un peso (distancia, retardo, etc.). Una conexión punto a punto entre dos enrutadores se representa por un par de arcos, uno en cada dirección. Sus pesos pueden ser diferentes. Una red de difusión se representa con un nodo para la red en sí, más un nodo para cada enrutador. Los arcos de ese nodo de la red a los enrutadores tienen un peso de 0. Sin embargo son importantes, puesto que sin ellos no habrá una ruta a través de la red. Otras redes, que sólo tienen hosts, tienen únicamente un arco que llega a ellas y no uno que regresa. Esta estructura proporciona rutas a los hosts, pero no a través de ellos.

La Ilustración 64(b) muestra la representación gráfica de la red de la Ilustración 64(a). En esencia, lo que OSPF hace es representar la red real mediante un grafo como éste y después usa el método de estado del enlace para hacer que cada enrutador calcule la ruta más corta desde sí mismo hacia todos los demás nodos. Se pueden encontrar varias rutas que sean igual de cortas. En este caso, OSPF recuerda el conjunto de rutas más

cortas y, durante el reenvío de paquetes, el tráfico se divide entre ellas. Este proceso ayuda a balancear la carga y se conoce como **ECMP** (Multiruta de Igual Costo, del inglés *Equal Cost MultiPath*).

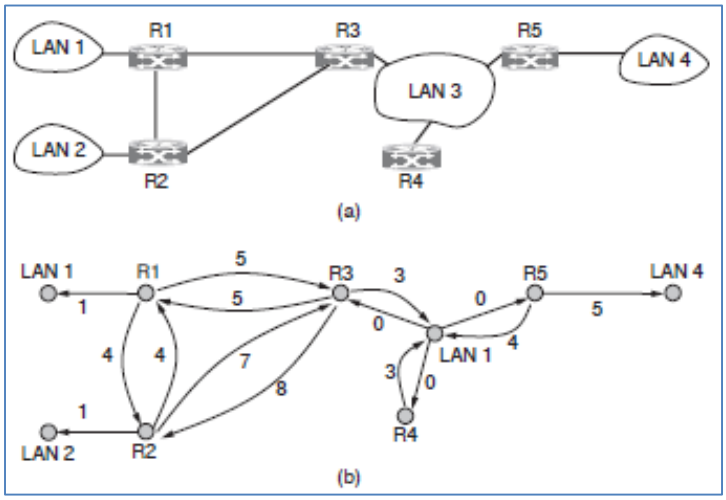


Ilustración 62 - (a) Un sistema autónomo. (b) Una representación gráfica de (a).

Muchos de los sistemas autónomos (AS) en Internet son grandes por sí mismos y nada sencillos de administrar. Para trabajar a esta escala, OSPF permite dividir un AS en áreas numeradas, en donde un área es una red o un conjunto de redes contiguas. Las áreas no se traslapan ni necesitan ser exhaustivas; es decir, algunos enrutadores no necesitan pertenecer a ningún área. Los enrutadores que están totalmente dentro de un área se llaman **enrutadores internos**. Un área es una generalización de una red individual. Fuera de un área, sus destinos son visibles pero su topología no. Esta característica ayuda a escalar el enrutamiento.

Cada AS tiene un **área troncal** (*backbone area*), llamada área 0. Los enrutadores en esta área se llaman **enrutadores troncales** (*backbone routers*). Todas las áreas se conectan a la red troncal, posiblemente mediante túneles, de modo que es posible ir desde cualquier área en el AS a cualquier otra área en el AS mediante la red troncal. En el grafo, un túnel se representa como otro arco más con un costo. Al igual que con otras áreas, la topología de la red troncal no es visible fuera de ésta.

Cada enrutador que se conecta a dos o más áreas se denomina **enrutador de frontera de área**. También debe formar parte de la red troncal. El trabajo de un enrutador de frontera de área es resumir todos los destinos en un área e inyectar este resumen en las otras áreas a las que está conectado. El resumen incluye la información de costo, pero no todos los detalles de la topología dentro de un área. Pasar la información de costo, permite a los hosts en otras áreas encontrar el mejor enrutador de frontera de área que puede usar para entrar a un área. Al no pasar la información sobre la topología se reduce el tráfico y se simplifican los cálculos de la ruta más corta para los enrutadores en otras áreas. No obstante, si sólo hay un enrutador de frontera fuera de un área, no es necesario ni siquiera pasar el resumen. Los enrutadores a destinos fuera del área siempre empiezan con la instrucción “Ir al enrutador de frontera”. Este tipo de área se denomina **área aislada** (*stub area*).

El último tipo de enrutador es el **enrutador de límite de AS**. Éste inyecta en el área las rutas a destinos externos en otros sistemas autónomos. Después, las rutas externas aparecen como destinos a los que se puede llegar por medio del enrutador de límite de AS con un cierto costo. Una ruta externa se puede inyectar en uno o más enrutadores de límite de AS. En la Ilustración 63 se muestra la relación entre los sistemas autónomos, las áreas y los diversos tipos de enrutadores. Un enrutador puede desempeñar distintos roles; por ejemplo, un enrutador de frontera puede ser también un enrutador troncal.

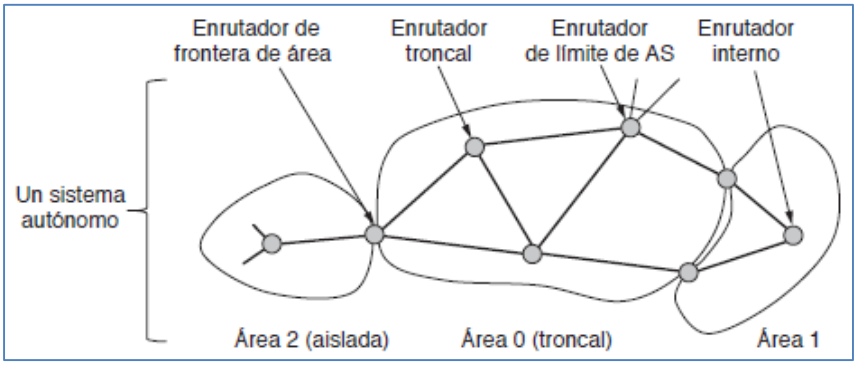


Ilustración 63 - La relación entre sistemas autónomos, redes troncales y áreas en OSPF.

Durante la operación normal, cada enrutador dentro de un área tiene la misma base de datos de estado del enlace y ejecuta el mismo algoritmo de la ruta más corta. Su tarea principal es calcular la ruta más corta desde sí mismo hasta cada uno de los demás enrutadores y redes en todo el AS. Un enrutador de frontera de área necesita las bases de datos para todas las áreas a las que está conectado y debe ejecutar el algoritmo de la ruta más corta para cada área por separado.

Para un origen y un destino en la misma área, se selecciona la mejor ruta intra-área (que se encuentre totalmente dentro del área). Para un origen y un destino en distintas áreas, la ruta inter-área debe ir del origen a la red troncal, a través de la red troncal hasta el área de destino, y después al destino. Este algoritmo obliga a una configuración de estrella en OSPF, en donde la red troncal es el hub y las demás áreas son las extensiones. Puesto que se selecciona la ruta con el menor costo, es probable que los enrutadores en distintas partes de la red usen distintos enrutadores de frontera de área para entrar a la red troncal y al área de destino. Los paquetes se enrutan del origen al destino "*como están*". No se encapsulan ni tunelizan (a menos que vayan a un área cuya única conexión a la red troncal sea un túnel). Además, las rutas a los destinos externos pueden incluir el costo externo del enrutador de límite de AS a través de la ruta externa, si se desea, o sólo el costo interno para el AS.

Cuando un enrutador se enciende, envía mensajes *HELLO* en todas sus líneas punto a punto y los envía por multidifusión a las LAN al grupo que consiste en los enrutadores restantes. Cada enrutador descubre quiénes son sus vecinos gracias a las respuestas. Todos los enrutadores en la misma LAN son vecinos. Para operar, OSPF intercambia información entre enrutadores adyacentes, que no es lo mismo que entre enrutadores vecinos. En particular, es ineficiente que cada enrutador en la LAN se comunique con cualquier otro enrutador en la LAN. Para evitar esta situación, se elige un enrutador como **enrutador designado**. Se dice que es **adyacente** a todos los demás enrutadores en su LAN, e intercambia información con ellos. En efecto, actúa como el único nodo que representa a la LAN. Los enrutadores vecinos que no son adyacentes no intercambian información entre sí. Un enrutador designado como respaldo siempre se mantiene actualizado para facilitar la transición en caso de que el enrutador designado primario falle y haya que reemplazarlo de inmediato.

Durante la operación normal, cada enrutador inunda periódicamente con mensajes *LINK STATE UPDATE* (ACTUALIZACIÓN DEL ESTADO DEL ENLACE) a cada uno de sus enrutadores adyacentes. Estos mensajes dan su estado y proporcionan los costos usados en la base de datos topológica. Para hacerlos confiables, se confirma la recepción de los mensajes de inundación. Cada mensaje tiene un número de secuencia para que un enrutador pueda ver si un *LINK STATE UPDATE* entrante es más viejo o más nuevo que el que tiene actualmente. Los enrutadores también envían estos mensajes cuando un enlace se activa o desactiva, o cuando cambia su costo.

Los mensajes *DATABASE DESCRIPTION* (DESCRIPCIÓN DE LA BASE DE DATOS) dan los números de secuencia de todas las entradas de estado del enlace que contiene el emisor en ese momento. Al comparar sus propios valores con los del emisor, el receptor puede determinar quién tiene los valores más recientes. Estos mensajes se usan cuando se activa un enlace.

Cualquier socio puede solicitar información del estado del enlace al otro mediante los mensajes *LINK STATE REQUEST* (PETICIÓN DEL ESTADO DEL ENLACE). El resultado de este algoritmo es que cada par de enrutadores

adyacentes verifica quién tiene los datos más recientes; de esta manera se difunde la nueva información a lo largo del área. Todos estos mensajes se envían de manera directa como paquetes IP. En la Ilustración 64 se resumen los cinco tipos de mensajes.

Tipo de mensaje	Descripción
Hello.	Se utiliza para descubrir quiénes son los vecinos.
Link state update.	Proporciona los costos del emisor a sus vecinos.
Link state ack.	Confirma la recepción de la actualización del estado del enlace.
Database description.	Anuncia qué actualizaciones tiene el emisor.
Link state request.	Solicita información del socio.

Ilustración 64 - Los cinco tipos de mensajes OSPF.

Finalmente podemos reunir todas las piezas. Mediante la inundación de mensajes, cada enrutador informa a todos los demás enrutadores en su área sobre sus enlaces a otros enrutadores y redes, junto con el costo de éstos. Esta información permite a cada enrutador construir el grafo para su(s) área(s) y calcular las rutas más cortas. El área red troncal también hace esto. Además, los enrutadores troncales aceptan la información de los enrutadores de frontera de área para calcular la mejor ruta desde cada enrutador troncal hasta cada uno de los demás enrutadores. Esta información se propaga de vuelta a los enrutadores de frontera de área, quienes la anuncian dentro de sus áreas. Mediante el uso de esta información, los enrutadores internos pueden seleccionar la mejor ruta a un destino fuera de su área, incluyendo el mejor enrutador de salida hacia la red troncal.

BGP: el protocolo de enrutamiento de puerta de enlace exterior

Dentro de un solo sistema autónomo, OSPF e IS-IS son los protocolos de uso común. Entre los sistemas autónomos se utiliza un protocolo diferente, conocido como **BGP** (Protocolo de Puerta de Enlace de Frontera, del inglés *Border Gateway Protocol*). Se necesita un protocolo diferente debido a que los objetivos de un protocolo intradominio y de un protocolo interdominio no son los mismos. Todo lo que tiene que hacer un protocolo intradominio es mover paquetes de la manera más eficiente posible desde el origen hasta el destino. No tiene que preocuparse por las políticas.

En contraste, los protocolos de enrutamiento interdominio tienen que preocuparse en gran manera por la política. Por ejemplo, tal vez un sistema autónomo corporativo desee la habilidad de enviar paquetes a cualquier sitio de Internet y recibir paquetes de cualquier sitio de Internet. Sin embargo, quizás no esté dispuesto a llevar paquetes de tránsito que se originen en un AS foráneo y estén destinados a un AS foráneo diferente, aun cuando su propio AS se encuentre en la ruta más corta entre los dos sistemas autónomos foráneos. Por otro lado, podría estar dispuesto a llevar el tráfico del tránsito para sus vecinos o incluso para otros sistemas autónomos específicos que hayan pagado por este servicio. Por ejemplo, las compañías telefónicas podrían estar contentas de actuar como empresas portadoras para sus clientes, pero no para otros. En general, los protocolos de puerta de enlace exterior (y BGP en particular) se han diseñado para permitir que se implementen muchos tipos de políticas de enrutamiento en el tráfico entre sistemas autónomos.

Las políticas típicas implican consideraciones políticas, de seguridad, o económicas. Algunos ejemplos de posibles restricciones de enrutamiento son:

1. No transportar tráfico comercial en la red educativa.
2. Usar TeliaSonera en vez de Verizon porque es más económico.
3. No usar AT&T en Australia porque el desempeño es pobre.
4. El tráfico que empieza o termina en Apple no debe transitar por Google.

Como puede imaginar de esta lista, las políticas de enrutamiento pueden ser muy individuales. A menudo son propietarias pues contienen información comercial delicada. Sin embargo, podemos describir algunos patrones que capturan el razonamiento anterior de la compañía y que se utilizan con frecuencia como un punto de partida.

Para implementar una política de enrutamiento hay que decidir qué tráfico puede fluir a través de cuáles enlaces entre los sistemas autónomos. Una política común es que un ISP cliente pague a otro ISP proveedor por entregar paquetes a cualquier otro destino en Internet y recibir los paquetes enviados desde cualquier otro destino. Se dice que el ISP cliente compra **servicio de tránsito** al ISP proveedor. Es justo igual que cuando un cliente doméstico compra servicio de acceso a Internet con un ISP. Para que funcione, el proveedor debe anunciar las rutas a todos los destinos en Internet al cliente a través del enlace que los conecta. De esta forma, el cliente tendrá una ruta para enviar paquetes a cualquier parte. Por el contrario, el cliente sólo debe anunciar al proveedor las rutas a los destinos en su red. Esto permitirá al proveedor enviar tráfico al cliente sólo para esas direcciones; al cliente no le conviene manejar el tráfico destinado a otras partes.

En la Ilustración 65 podemos ver un ejemplo del servicio de tránsito. Hay cuatro sistemas autónomos conectados. Con frecuencia, la conexión se hace mediante un enlace en **IXPs** (Puntos de Intercambio de Internet, del inglés *Internet eXchange Points*): instalaciones con las que muchos ISP tienen un enlace para fines de conectarse con otros ISP. *AS2*, *AS3* y *AS4* son clientes de *AS1*, pues le compran servicio de tránsito. Así, cuando la fuente *A* envía al destino *C*, los paquetes viajan de *AS2* hacia *AS1* y finalmente a *AS4*. Los anuncios de enrutamiento viajan en dirección opuesta a los paquetes. *AS4* anuncia a *C* como un destino a su proveedor de tránsito *AS1*, para que las fuentes puedan llegar a *C* por medio de *AS1*. Después, *AS1* anuncia a sus otros clientes, incluyendo *AS2*, una ruta a *C* para que éstos sepan que pueden enviar tráfico a *C* por medio de *AS1*.

En la Ilustración 65, los demás sistemas autónomos compran servicio de tránsito a *AS1*. Este servicio les proporciona conectividad para que puedan interactuar con cualquier host en Internet. Sin embargo, tienen que pagar por este privilegio. Suponga que *AS2* y *AS3* intercambian mucho tráfico. Dado que sus redes ya se encuentran conectadas, si lo desean pueden usar una política diferente: pueden enviar tráfico directamente uno al otro sin costo. Esto reducirá la cantidad de tráfico que debe entregar *AS1* a cuenta de *AS2* y *AS3*, y con suerte reducirá sus facturas. A esta política se le conoce como c**omunicación entre pares** (*peering*).

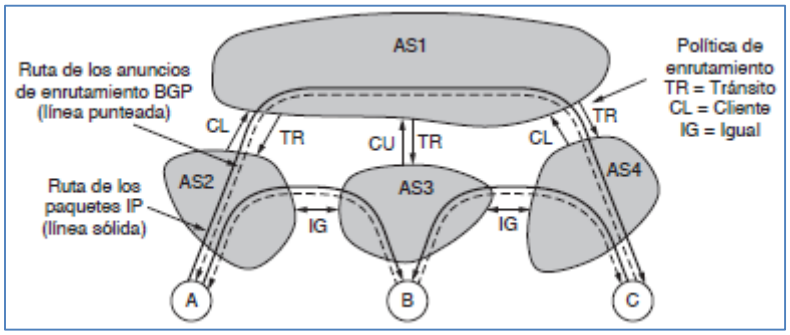


Ilustración 65 - Políticas de enrutamiento entre cuatro sistemas autónomos.

Para implementar la comunicación entre pares, dos sistemas autónomos se envían anuncios de enrutamiento entre sí, respecto a las direcciones que residen en sus redes. Al hacer esto, *AS2* puede enviar a *AS3* paquetes de *A* destinados a *B* y viceversa. Sin embargo, hay que tener en cuenta que la comunicación entre iguales no es transitiva. En la Ilustración 65, *AS3* y *AS4* también se comunican entre sí. Esta comunicación de igual a igual permite que el tráfico de *B*, que está destinado a *C*, se envíe directamente a *AS4*. ¿Qué ocurre si *C* envía un paquete a *A*? *AS3* sólo está anunciando a *AS4* una ruta a *B*. No está anunciando una ruta a *A*. La consecuencia es que el tráfico no pasará de *AS4* a *AS3* a AS2, aun cuando existe una ruta física. Esta restricción es justo lo que *AS3* quiere. Se comunica con *AS4* para intercambiar tráfico, pero no quiere transportar tráfico de *AS4* a otras partes de Internet, ya que no se le paga por hacerlo. En cambio, *AS4* recibe servicio de tránsito de *AS1*. Por ende, es *AS1* quien transportará el paquete de *C* a *A*.

Ahora que sabemos sobre el tránsito y la comunicación entre iguales, también podemos ver que *A*, *B* y *C* tienen arreglos de tránsito. Por ejemplo, *A* debe comprar acceso a Internet a *AS2*. *A* podría ser una sola computadora doméstica o la red de una compañía con muchas LAN. Sin embargo, no necesita ejecutar BGP debido a que es una **red aislada** (*stub network*) que está conectada al resto de Internet sólo mediante un enlace. Por tanto, el único lugar para enviar paquetes destinados a puntos que estén fuera de la red es a través del enlace a *AS2*. No hay ningún otro lugar a dónde ir. Para arreglar esta ruta, sólo hay que establecer una ruta predeterminada.

Por esta razón no hemos mostrado a *A*, *B* y *C* como sistemas autónomos que participan en el enrutamiento interdominio.

Por otro lado, las redes de algunas compañías están conectadas a varios ISP. Esta técnica se utiliza para mejorar la confiabilidad, ya que si la ruta a través de un ISP falla, la compañía puede usar la ruta a través del otro ISP. Esta técnica se conoce como **multihoming**. En este caso, la red de la compañía probablemente ejecute un protocolo de enrutamiento interdominio (como BGP) para indicar a otros sistemas autónomos qué enlaces de ISP pueden llegar a cuáles direcciones.

Hay muchas variaciones posibles de estas políticas de tránsito y comunicación entre iguales, pero todas ilustran cómo las relaciones de negocios y el control sobre el camino que pueden tomar los anuncios de rutas pueden implementar distintos tipos de políticas. Ahora consideraremos con más detalle cómo los enrutadores que ejecutan BGP se anuncian rutas entre sí y seleccionan rutas a través de las cuales pueden reenviar los paquetes.

BGP es una forma de protocolo de vector de distancia, aunque es bastante distinto a los protocolos de vector de distancia intradominio como RIP. Ya vimos antes que la política, y no la distancia mínima, se utiliza para elegir qué rutas usar. Otra gran diferencia es que, en vez de mantener sólo el costo de la ruta a cada destino, cada enrutador BGP lleva el registro de la ruta utilizada. Esta metodología se conoce como **protocolo de vector de ruta**. La ruta consiste en el enrutador del siguiente salto (que puede estar del otro lado del ISP y no necesariamente ser adyacente) y la secuencia de sistemas autónomos (o **ruta AS**) en el recorrido (se proporciona en orden inverso). Por último, los pares de enrutadores BGP se comunican entre sí mediante el establecimiento de conexiones TCP. Al operar de esta forma se obtiene una comunicación confiable; además se ocultan todos los detalles de la red que se está atravesando.

En la Ilustración 66 se muestra un ejemplo de cómo se anuncian las rutas BGP. Hay tres sistemas autónomos y el de en medio provee tránsito a los ISP izquierdo y derecho. Un anuncio de ruta para el prefijo *C* empieza en *AS3*. Cuando se propaga a través del enlace a *R2c* en la parte superior de la figura, tiene la ruta AS que consiste tan sólo en *AS3* y el enrutador del siguiente salto de *R3a*. En la parte inferior tiene la misma ruta AS pero un siguiente salto distinto, debido a que provino de un enlace diferente. Este anuncio continúa su propagación y atraviesa el límite hacia *AS1*. En el enrutador *R1a*, en la parte superior de la figura, la ruta AS es *AS2*, *AS3* y el siguiente salto es *R2a*.

Al transportar la ruta completa con el recorrido, es fácil para el enrutador receptor detectar e interrumpir los ciclos de enrutamiento. La regla es que cada enrutador que envíe un recorrido hacia fuera del AS anteponga su propio número de AS a este recorrido (ésta es la razón por la cual la lista está en orden inverso). Cuando un enrutador recibe un recorrido, verifica que su propio número de AS ya se encuentre en la ruta AS. Si es así, se ha detectado un ciclo y se descarta el anuncio. No obstante (aunque suene un poco irónico), a finales de la década de 1990 se descubrió que a pesar de esta precaución, BGP sufre de una versión del problema de conteo al infinito. No hay ciclos de larga duración, pero algunas veces los recorridos pueden ser lentos para converger y tienen ciclos transitorios.

Proporcionar una lista de sistemas autónomos es una forma muy burda de especificar una ruta. Un AS podría ser una compañía pequeña o una red troncal internacional. No hay forma de saberlo con base en la ruta. BGP ni siquiera lo intenta, ya que los distintos sistemas autónomos pueden usar protocolos intradominio diferentes, cuyos costos no se puedan comparar. Incluso si se pudieran comparar, tal vez un AS no quiera revelar su métrica interna. Ésta es una de las diferencias entre los protocolos de enrutamiento interdominio y los protocolos intradominio.

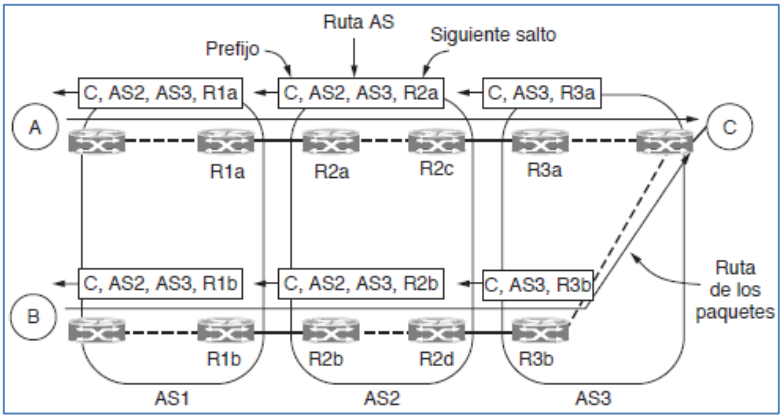


Ilustración 66 - Propagación de los anuncios de rutas de BGP.

Hasta ahora hemos visto cómo se envía un anuncio de ruta a través del enlace entre dos ISP. Todavía se necesita una manera de propagar rutas BGP de un lado del ISP al otro, para que se puedan enviar al siguiente ISP. El protocolo intradominio podría manejar esta tarea, pero ya que BGP es muy bueno para escalar a grandes redes, lo común es utilizar una variante de este protocolo, conocido como **iBGP** (BGP interno, del inglés *internal BGP*) para diferenciarlo del uso regular de BGP como *eBGP* (BGP externo, del inglés *external BGP*).

La regla para propagar rutas dentro de un ISP es que cada enrutador en el límite del ISP aprenda de todas las rutas vistas por todos los demás enrutadores de límite, para fines de consistencia. Si un enrutador de límite en el ISP aprende un prefijo para el IP 128.208.0.0/16, todos los demás enrutadores conocerán este prefijo. Así, se podrá llegar al prefijo desde cualquier parte del ISP, sin importar cuántos paquetes entren al ISP provenientes de otros sistemas autónomos.

No mostramos esta propagación en la Ilustración 66 para evitar un desorden, pero por ejemplo, el enrutador *R2b* sabrá que puede llegar a *C* por medio del enrutador *R2c* en la parte superior o del enrutador *R2d* en la parte inferior. El siguiente salto se actualiza a medida que la ruta atraviesa el interior del ISP, de modo que los enrutadores del extremo opuesto del ISP sepan qué enrutador usar para salir del ISP por el otro lado. Podemos ver esto en las rutas de más a la izquierda, en donde el siguiente salto apunta a un enrutador en el mismo ISP y no a un enrutador en el siguiente ISP.

Ahora podemos describir la pieza clave faltante, que es la forma en que los enrutadores BGP eligen la ruta a usar para cada destino. Cada enrutador BGP puede conocer una ruta para un destino dado por medio del enrutador al que está conectado en el siguiente ISP y de todos los demás enrutadores de límite (que han escuchado distintas rutas de los enrutadores conectados a otros ISP). Cada enrutador debe decidir cuál ruta de este conjunto de rutas es la mejor que puede usar. En última instancia, la respuesta es que el ISP debe escribir una política para elegir la ruta preferida. Sin embargo, esta explicación es muy general y no del todo satisfactoria, por lo que al menos podemos describir algunas estrategias comunes.

La primera estrategia es seleccionar las rutas a través de redes entre iguales en vez de las rutas a través de los proveedores de tránsito. Las primeras son gratuitas; las segundas cuestan dinero. Una estrategia similar es dar a las rutas de los clientes la mayor preferencia. Es un buen negocio enviar tráfico directamente a los clientes que pagan.

Un tipo distinto de estrategia es la regla predeterminada que establece que las rutas AS más cortas son mejores. Es algo debatible, puesto que un AS podría ser una red de cualquier tamaño, por lo que una ruta a través de tres pequeños sistemas autónomos podría en realidad ser más corta que una ruta a través de un AS extenso. Sin embargo, lo más corto tiende a ser mejor en promedio, y esta regla es un punto de desempate común.

La estrategia final es preferir la ruta que tenga el menor costo dentro del ISP. Ésta es la estrategia que se implementa en la Ilustración 66. Los paquetes enviados de *A* a *C* salen de *AS1* por el enrutador superior, *R1a*. Los paquetes enviados desde *B* salen a través del enrutador inferior, *R1b*. La razón es que tanto *A* como *B* están tomando la ruta de menor costo, o la ruta más rápida, para salir de *AS1*. Como se encuentran en distintas

partes del ISP, la salida más rápida para cada uno es distinta. Lo mismo ocurre cuando los paquetes pasan a través de *AS2*. En el último tramo, *AS3* tiene que transportar el paquete proveniente de *B* a través de su propia red.

Esta estrategia se conoce como **enrutamiento de salida anticipada** o **enrutamiento de la papa caliente**. Tiene el curioso efecto colateral de tender a crear rutas asimétricas. Por ejemplo, considere la ruta que se toma cuando *C* envía un paquete de vuelta a *B*. El paquete saldrá rápidamente de *AS3*, en el enrutador superior, para evitar desperdiciar sus recursos. Asimismo, permanecerá en la parte superior cuando *AS2* pase el paquete a *AS1* lo más rápido que pueda. Después el paquete tendrá un viaje más largo en *AS1*. Ésta es una imagen espejo de la ruta que se toma de *B* a *C*.

La discusión anterior debería dejar en claro que cada enrutador BGP selecciona su propia mejor ruta a partir de las posibilidades conocidas. No es el caso que BGP seleccione una ruta a seguir en el nivel de AS y OSPF seleccione rutas dentro de cada uno de los sistemas autónomos. BGP y el protocolo de puerta de enlace interior están integrados de una manera mucho más profunda. Esto significa que, por ejemplo, BGP puede encontrar el mejor punto de salida de un ISP al siguiente y este punto variará a lo largo del ISP, como es el caso en la política de la papa caliente. También significa que los enrutadores BGP en distintas partes de un AS pueden elegir distintas rutas AS para llegar al mismo destino. El ISP debe tener cuidado al configurar todos los enrutadores BGP para hacer elecciones compatibles teniendo en cuenta toda esta libertad, pero se puede hacer esto en la práctica.

Aunque parezca sorprendente, sólo hemos arañado la superficie de BGP. Para obtener más información, consulte la especificación de la versión 4 de BGP en el RFC 4271 y los RFC relacionados. Sin embargo, cabe mencionar que la mayor parte de su complejidad está en las políticas, las cuales no se describen en la especificación del protocolo BGP.

Multidifusión de Internet

La comunicación normal de IP está entre un emisor y un receptor. Sin embargo, para algunas aplicaciones es útil que un proceso pueda enviar a una gran cantidad de receptores en forma simultánea. Algunos ejemplos son: transmitir por flujo continuo un evento deportivo para muchos espectadores, entregar actualizaciones de programas a una reserva de servidores replicados y manejar llamadas telefónicas de conferencias digitales (es decir, entre varios participantes).

IP apoya la comunicación de uno a varios, o **multidifusión**, mediante el uso de direcciones IP clase D. Cada dirección clase D identifica a un grupo de hosts. Hay 28 bits disponibles para identificar a los grupos, de modo que pueden existir al mismo tiempo más de 250 millones de grupos. Cuando un proceso envía un paquete a una dirección clase D, se hace el mejor esfuerzo por entregarlo a todos los miembros del grupo direccionado, pero no se da garantía alguna. Quizá algunos miembros no reciban el paquete.

El rango de direcciones IP 224.0.0.0/24 está reservado para multidifusión en la red local. En este caso no se necesita un protocolo de enrutamiento. Para enviar los paquetes por multidifusión, simplemente se difunden en la LAN con una dirección de multidifusión. Todos los hosts en la LAN reciben las difusiones y los hosts que pertenecen al grupo procesan el paquete. Los enrutadores no reenvían el paquete fuera de la LAN. Algunos ejemplos de direcciones de multidifusión locales son:

224.0.0.1 Todos los sistemas en una LAN.

224.0.0.2 Todos los enrutadores en una LAN.

224.0.0.5 Todos los enrutadores OSPF en una LAN.

224.0.0.251 Todos los servidores DNS en una LAN.

Otras direcciones de multidifusión pueden tener miembros en distintas redes. En este caso se necesita un protocolo de enrutamiento. Pero primero, los enrutadores multidifusión necesitan saber qué hosts son miembros de un grupo. Un proceso pide a su host que se una a un grupo específico. También puede pedir a su host que salga del grupo. Cada host lleva el registro de los grupos a los que pertenecen actualmente sus procesos. Cuando el último proceso de un host sale de un grupo, el host deja de ser miembro de ese grupo.

Aproximadamente una vez por minuto, cada enrutador multidifusión envía un paquete de consulta a todos los hosts en su LAN (mediante la dirección de multidifusión local 224.0.0.1, por supuesto) y les pide que se reporten de vuelta a los grupos a los que pertenecen en la actualidad. Los enrutadores multidifusión pueden colocarse o no con los enrutadores estándar. Cada host envía respuestas de vuelta a todas las direcciones de clase D en las que está interesado. Estos paquetes de consulta y respuesta usan un protocolo llamado **IGMP** (Protocolo de Administración de Grupo de Internet, del inglés *Internet Group Management Protocol*), que se describe en el RFC 3376.

Se puede usar cualquiera de varios protocolos de enrutamiento multidifusión para construir árboles de expansión de multidifusión que proporcionen rutas de los emisores a todos los miembros del grupo. Los algoritmos que se utilizan son los que describimos anteriormente. Dentro de un AS, el protocolo principal que se utiliza es **PIM** (Multidifusión Independiente del Protocolo, del inglés *Protocol Independent Multicast*). Hay varios tipos de PIM. En el *PIM en modo denso*, se crea un árbol de reenvío por ruta inversa recortado. Éste es adecuado para los casos en que los miembros están en todas partes en la red, como al distribuir archivos a muchos servidores dentro de la red de un centro de datos. En el *PIM en modo disperso*, los árboles de expansión que se construyen son similares a los árboles de núcleo. Esto es adecuado para los casos en que un proveedor de contenido transmite por multidifusión la señal de TV a los suscriptores en su red IP. Una variante de este diseño, conocida como *PIM de multidifusión específico del origen*, está optimizada para el caso en que sólo hay un emisor para el grupo. Por último, hay que usar extensiones multidifusión para BGP o túneles para poder crear rutas multidifusión cuando los miembros en el grupo se encuentran en más de un AS.

IP móvil

Muchos usuarios de Internet tienen computadoras móviles y desean permanecer conectados cuando están lejos de su hogar, e incluso durante el camino. Por desgracia, con el sistema de direccionamiento de IP, trabajar lejos de casa es más fácil de decir que de hacer, como veremos en breve. De todas formas, cuando las personas empezaron a exigir esta capacidad, la IETF estableció un grupo de trabajo para encontrar una solución. El grupo de trabajo formuló con rapidez varias metas consideradas como deseables en cualquier solución. Las principales fueron:

1. Cada host móvil debía ser capaz de **usar su dirección IP base en cualquier parte**.
2. **No se permitían cambios de software** en los hosts fijos.
3. No se permitían cambios en el software **ni en las tablas del enrutador**.
4. La mayoría de paquetes para host móviles **no debían desviarse de la ruta**.
5. **No se debía incurrir en sobrecarga** cuando un host móvil estuviera en casa.

La solución que se eligió se describió en la sección "*Enrutamiento para host móviles*". En resumen, cada sitio que desee permitir a sus usuarios vagar tiene que crear un ayudante en el sitio, conocido como **agente de base**. Cuando aparece un host móvil en un sitio foráneo, obtiene una nueva dirección IP (conocida como **dirección de custodia**) del sitio foráneo. Después el móvil indica al agente de base su ubicación actual, para lo cual le proporciona la dirección de custodia. Cuando llega un paquete para el móvil en el sitio base y éste se encuentra en otra parte, el agente de base agarra el paquete y lo envía a través de un **túnel** al móvil, a la dirección actual de custodia. El móvil puede enviar paquetes de respuesta directamente a cualquiera con quien se esté comunicando, pero sigue usando su dirección base como la dirección de origen. Esta solución cumple con todos los requerimientos indicados antes, excepto que los paquetes para los hosts móviles sí se desvían.

Ya que cubrimos la capa de red de Internet, podemos analizar la solución con más detalle. La necesidad de soporte móvil en primer lugar proviene del mismo esquema de direccionamiento IP. Cada dirección IP contiene un número de red y un número de host. Por ejemplo, considere la máquina con la dirección IP 160.80.40.20/16. La parte 160.80 representa el número de red; la parte 40.20 es el número de host. Los enrutadores de todo el mundo tienen tablas de enrutamiento, las cuales les indican qué enlace deben usar para llegar a la red 160.80. Cada vez que llega un paquete con una dirección IP de destino de la forma 160.80.xxx.yyy, pasa por esa línea. Si de repente la máquina con esa dirección se transporta a un sitio distante, los paquetes para ella se seguirán enrutando hacia su LAN (o enrutador) base.

En esta etapa hay dos opciones, aunque ambas son poco atractivas. La primera es que podríamos crear una ruta hacia un prefijo más específico. Es decir, si el sitio distante anuncia una ruta a 160.80.40.20/32, los

paquetes que se envíen al destino empezarán a llegar al lugar correcto otra vez. Esta opción depende del algoritmo del prefijo más largo coincidente que se utiliza en los enrutadores. Sin embargo, hemos agregado una ruta a un prefijo IP con una sola dirección IP en él. Todos los ISP en el mundo conocerán este prefijo. Si todos cambiaran las rutas IP globales de esta forma al mover su computadora, cada enrutador tendría millones de entradas en la tabla, con un costo astronómico para Internet. **Esta opción no es funcional**.

La segunda opción es cambiar la dirección IP del móvil. Es cierto, los paquetes que se envíen a la dirección IP base ya no se entregarán sino hasta que se informa a todas las personas, programas y bases de datos relevantes sobre el cambio. Pero el móvil todavía puede usar Internet en la nueva ubicación para navegar en la Web y ejecutar otras aplicaciones. Esta opción maneja la movilidad en una capa superior. Es lo que ocurre comúnmente cuando un usuario lleva su computadora portátil a una cafetería y usa Internet a través de la red inalámbrica local. La desventaja es que interrumpe algunas aplicaciones y no mantiene la conectividad mientras el equipo se desplaza de un lado a otro.

Como observación adicional, la movilidad también se puede manejar en una capa inferior: la capa de enlace. Esto es lo que ocurre al usar una computadora portátil en una sola red inalámbrica 802.11. La dirección IP del móvil no cambia y la ruta de red permanece igual. Es el enlace inalámbrico quien proporciona la movilidad. Sin embargo, el grado de movilidad es limitado. Si la computadora portátil se aleja demasiado, tendrá que conectarse a Internet por medio de otra red, con una dirección IP distinta.

La solución de IP móvil para IPv4 se proporciona en el RFC 3344. Funciona con el enrutamiento de Internet existente y permite a los hosts permanecer conectados con sus propias direcciones IP a medida que se desplazan. Para que funcione, el móvil debe ser capaz de detectar cada vez que se mueve. Para ello se utiliza el anuncio del enrutador ICMP y los mensajes de solicitud. Los móviles escuchan los anuncios periódicos del enrutador o envían una solicitud para descubrir el enrutador más cercano. Si este enrutador no tiene la dirección usual del enrutador cuando el móvil está en su base, debe estar en una red foránea. Si este enrutador cambió desde la última vez, el móvil se desplazó a otra red foránea. Este mismo mecanismo permite a los hosts móviles encontrar a sus agentes de base.

Para obtener una dirección IP de custodia en la red foránea, un móvil puede simplemente usar DHCP. Como alternativa, si hay una escasez de direcciones IPv4, el móvil puede enviar y recibir paquetes a través de un agente foráneo que ya tenga una dirección IP en la red. Para buscar un agente foráneo, el host móvil usa el mismo mecanismo ICMP que se utiliza para buscar el agente de base. Una vez que el móvil obtiene una dirección IP o encuentra un agente foráneo, puede usar la red para enviar un mensaje a su agente de base e informarle sobre su ubicación actual.

El agente de base necesita una manera de interceptar los paquetes que se envían al móvil, sólo cuando éste no está en su base. ARP ofrece un mecanismo conveniente para ello. Para enviar un paquete a través de una red Ethernet a un host IP, el enrutador necesita conocer la dirección Ethernet del host. El mecanismo usual es que el enrutador envíe una consulta ARP para preguntar, por ejemplo, cuál es la dirección Ethernet de 160.80.40.20. Cuando el móvil está en su base, responde a las consultas ARP destinadas a su dirección IP con su propia dirección Ethernet. Cuando el móvil está fuera de su base, el agente de base responde a esta consulta, para lo cual proporciona su dirección Ethernet. A continuación el enrutador envía los paquetes para 160.80.40.20 al agente de base. Recuerde que a esto se le denomina **ARP de proxy**.

Para actualizar de forma rápida las asociaciones ARP de un lado a otro, cuando el móvil sale de su base o llega a ella, se puede usar otra técnica ARP conocida como **ARP gratuito**. Básicamente, el móvil o el agente de base se envían a sí mismos una consulta ARP para la dirección IP móvil que suministre la respuesta correcta, de modo que el enrutador detecte y actualice su asociación.

La tunelización para enviar un paquete entre el agente de base y el host móvil en la dirección de custodia se lleva a cabo mediante el encapsulamiento del paquete con otro encabezado IP destinado para la dirección de custodia. Cuando el paquete encapsulado llega a la dirección de custodia, se elimina el encabezado IP exterior para revelar el paquete.

Al igual que con muchos protocolos de Internet, el secreto está en los detalles; con más frecuencia, en los detalles de la compatibilidad con otros protocolos que se implementan. Existen dos complicaciones. En primer

lugar, las cajas NAT necesitan hurgar más allá del encabezado IP para analizar el encabezado TCP o UDP. La forma original de tunelización para IP móvil no usaba estos encabezados, por lo que no funcionaba con cajas NAT. La solución era cambiar el encapsulamiento para incluir un encabezado UDP.

La segunda complicación es que algunos ISP verifican las direcciones IP de origen de los paquetes para ver que coincidan en donde el protocolo de enrutamiento cree que se debería encontrar el origen. Esta técnica se denomina filtrado de ingreso y es una medida de seguridad con el propósito de descartar tráfico con direcciones aparentemente incorrectas que pueden ser maliciosas. Sin embargo, los paquetes que se envían del móvil a otros hosts de Internet cuando se encuentra en una red foránea tendrán una dirección IP de origen que estará fuera de lugar, por lo que se descartarán. Para resolver este problema, el móvil puede usar la dirección de custodia como una fuente para enviar mediante un túnel los paquetes de vuelta al agente de base. A partir de aquí, se pueden enviar a Internet desde lo que parece ser la ubicación correcta. El costo es que la ruta es más indirecta.

La seguridad es otro aspecto que no hemos examinado. Cuando un agente de base recibe un mensaje en el que se le pide que reenvíe todos los paquetes de Roberta a cierta dirección IP, es mejor que no acceda a menos que esté convencido de que Roberta es la fuente de esta petición, y no alguien tratando de hacerse pasar por ella. Para este fin se utilizan los protocolos de autentificación criptográficos, que estudiaremos en la próxima unidad.

Los protocolos de movilidad para IPv6 se basan en los fundamentos de IPv4. El esquema anterior sufre del problema de enrutamiento en triángulo, en donde los paquetes que se envían al móvil toman una ruta errónea a través de un agente de base distante. En IPv6 se utiliza la optimización de rutas para seguir una ruta directa entre el móvil y las otras direcciones IP, una vez que los paquetes iniciales han seguido la ruta larga. El IPv6 móvil se define en el RFC 3775.

Hay otro tipo de movilidad que también se está definiendo para Internet. Algunos aviones cuentan con redes inalámbricas integradas, que los pasajeros pueden usar para conectar sus computadoras portátiles a Internet. El avión tiene un enrutador que se conecta con el resto de Internet a través de un enlace inalámbrico. Por lo tanto, ahora tenemos un enrutador volador, lo cual significa que toda la red es móvil. Los diseños de movilidad de red soportan esta situación sin que las computadoras portátiles se den cuenta de que el avión es móvil. En cuanto a lo que a ellas concierne, es sólo otra red más. Puesto que algunas de las computadoras portátiles pueden usar el IP móvil para mantener sus direcciones base mientras están en el avión, entonces tenemos dos niveles de movilidad. La movilidad de red para IPv6 se define en el RFC 3963.

Referencias

- *Comunicaciones y Redes de Computadores 7ma Edición*, William Stallings, Pearson Educación, 2004.
- *Redes de Computadoras 5ta Edición*, Andrew S. Tanenbaum, Pearson Educación, 2012