La capa de red

René Serral i Gracià
Miquel Font Rosselló
Xavier Vilajosana Guillén
Eduard Lara Ochoa

PID_00171191

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción..... 5

1. Funciones básicas: encaminamiento..... 7

2. Servicios de red..... 10
2.1. Modelo de red en modo de circuitos virtuales 10
2.2. Modelo de red en modo datagrama 11
2.3. Servicio de red orientado y no orientado a la conexión 12

3. Direccionamiento en Internet. El protocolo IP..... 14
3.1. IPv4 14
3.1.1. La cabecera IP 14
3.1.2. Direccionamiento IPv4 19
3.1.3. CIDR 23
3.1.4. Tipos de datagramas IP 27
3.1.5. El futuro de IPv4 29
3.2. IPv6 30
3.2.1. Motivación 30
3.2.2. Cabecera IPv6 31
3.2.3. Problemas de la migración a IPv6 35
3.2.4. Mecanismos para asistir la transición 36
3.3. Protocolos de soporte a IP 37
3.3.1. ICMP 37
3.3.2. ARP 39
3.3.3. NDP 40
3.3.4. BOOTP 41
3.3.5. DHCP 42
3.3.6. DNS 42

4. Algoritmos y mecanismos de encaminamiento..... 45
4.1. Algoritmo de encaminamiento por la ruta más corta 46
4.2. Inundación 49
4.3. Algoritmo de encaminamiento de estado del enlace 50
4.4. Algoritmo de encaminamiento vector-distancia 52
4.5. Encaminamiento basado en difusión 57
4.6. Encaminamiento basado con multidifusión 58

5. Protocolos de encaminamiento en Internet..... 60
5.1. RIP 62
5.2. OSPF 65
5.3. BGP 67

Resumen..... 70

Bibliografía..... 71

Introducción

La capa de red se encarga de proporcionar conectividad y de ofrecer mecanismos para la selección del mejor camino entre dos puntos separados de la red, lo que permite la interconexión de equipos que pueden estar ubicados en redes geográficamente separadas entre ellas y garantiza la conectividad extremo a extremo, independientemente de la tecnología de enlace de datos utilizada y del camino que siga la información en los puntos intermedios.

Las principales ventajas que nos proporciona esta capa son, por una parte, independencia de la tecnología de red (hacia capas inferiores) y, por otra, un sistema de abstracción que permite utilizar una gran variedad de aplicaciones y protocolos de transporte (hacia capas superiores), como los TCP o UDP vistos en el módulo "La capa de transporte de datos".

Básicamente, la capa de red, sobre todo en Internet, está compuesta por tres grandes bloques. En primer lugar, el protocolo, que describe el modo de enviar información; en segundo lugar, el protocolo de encaminamiento, que decide por dónde deben ir los datagramas para llegar a su destino; y, en tercer lugar, la capa de red, que también identifica el mecanismo para informar de cualquier error que se haya producido en el envío de la información.

El objetivo de este módulo es describir cómo implementa la comunicación la capa de red. Así, el apartado 1 introduce los fundamentos y las funciones básicas presentes en esta capa. El apartado 2 describe los diferentes servicios que proporciona la capa de red desde el punto de vista del envío de información entre los diferentes nodos. El módulo continúa con el apartado 3, que contiene la parte más importante del capítulo, la descripción del protocolo de red más utilizado hoy: *Internet protocol*; se explica en qué consiste y cómo se utiliza éste en un entorno real como Internet. Para completar la comprensión del protocolo se introducen los mecanismos existentes para hacer llegar la información enviada a cualquier parte de la red. El módulo termina con las conclusiones de la capa de red y del protocolo IP.

1. Funciones básicas: encaminamiento

Una red está compuesta básicamente por dos tipos de entidades: los equipos finales[1] y los encaminadores[2].

(1)En inglés, *hosts*. También se pueden denominar clientes.

(2)En inglés, *routers*.

Los equipos finales son los equipos de red encargados de la comunicación; son el origen y el final de ésta. Normalmente, son servidores de información o equipos de usuarios finales que acceden a los servidores. Por su parte, los encaminadores, a pesar de que en según qué casos también pueden ser equipos finales, se limitan a enviar la información que reciben por una interfaz de entrada a la correspondiente de salida que lleve los datagramas a su destino. Para poder saber hacia dónde va la información, los encaminadores se ayudan de lo que se conoce como tablas de encaminamiento.

Si bien dejamos para el apartado 4 la descripción de los principales algoritmos de encaminamiento, y para el apartado 5 la del encaminamiento dentro de una red como Internet, en esta sección detallaremos las tareas generales en la capa de red que realizan los encaminadores, cómo llevan a cabo el envío de información y los problemas con los que se pueden encontrar.

La capa de red necesita que tanto los encaminadores como los equipos finales tengan un identificador único. Este identificador permite que cualquier otro equipo de la red lo pueda localizar y enviarle información. En particular, en una red como Internet estos identificadores se conocen como direcciones (direcciones IP).

Ved también

Podéis ver las direcciones IP en el apartado 3 de este módulo didáctico.

La figura 1 muestra una red con 8 encaminadores y dos equipos finales. En la figura también se puede observar una simplificación de cómo funciona un encaminador internamente. Para simplificar, en lugar de indicar las direcciones de los diferentes equipos, los hemos identificado, por una parte, como R (de *router*) y un número que identifica los diferentes encaminadores, y, por otra, como H (de *host*) y un número para identificar los diferentes equipos finales.

Los encaminadores están compuestos por una serie de interfaces de entrada y salida, que son las encargadas de recibir los datagramas de los equipos vecinos; estas interfaces están controladas por unas colas[3] que almacenan los paquetes (de entrada o de salida) para poder enviarlos cuando sea posible o, lo que es lo mismo, cuando el encaminador tenga recursos para atender las colas de entrada, o cuando la red tenga recursos (ancho de banda disponible) para las colas de salida. Internamente, el encaminador dispone de una lógica para

(3)En inglés, *buffers*.

decidir qué hacer con los datagramas que llegan. Normalmente, esta decisión implica enviar el datagrama por otra interfaz que lo llevará más cerca de su destino.

Así, el datagrama va saltando por los encaminadores hasta llegar al destino. Cada equipo de red por el que pasa el datagrama se conoce como salto[4]. Debemos señalar que los encaminadores trabajan a nivel de red, lo que significa que no interpretan los campos presentes en los niveles superiores, tal como muestra la figura 2.

(4)En inglés, *hop*.

Figura 1. Ejemplo de red con encaminadores y equipos finales

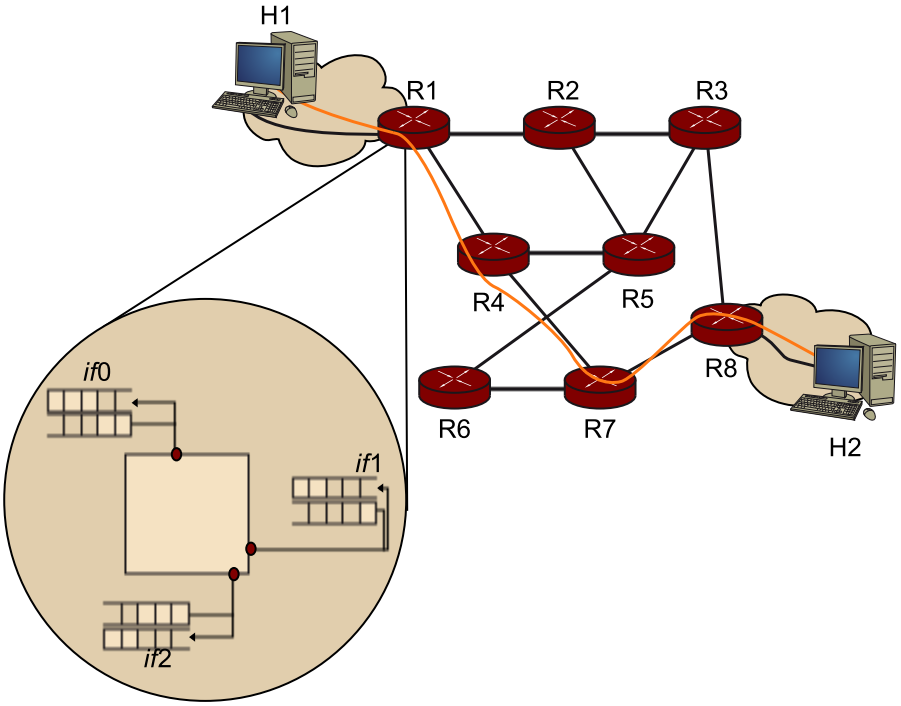
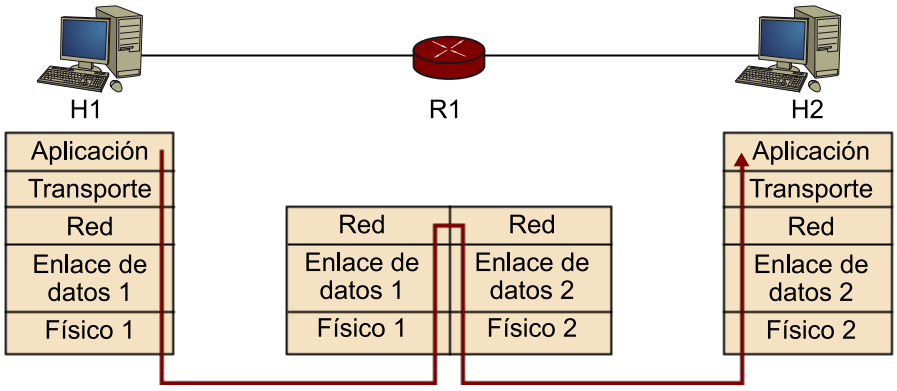


Figura 2. Capas usadas por el encaminamiento en protocolos de red



Cuando un equipo envía un datagrama a un destino, inicialmente este datagrama va dirigido al encaminador asociado a la red del equipo. Este encaminador mirará el destino del datagrama y lo enviará mediante la interfaz que lo lleve a su destino, según una tabla de encaminamiento. El siguiente encaminador hará lo mismo, hasta que el datagrama llegue a su destino final. La lista de encaminadores que sigue un datagrama se conoce como camino[5] del

(5)En inglés, *path*.

datagrama. Cabe destacar que este camino será diferente en función del origen y el destino del datagrama. Por ejemplo, en la figura 1 podemos ver que el camino que siguen los datagramas para ir desde H1 hasta H2 es H1-R1-R4-R7-R8-H2, con lo que dan un total de 5 saltos para llegar al destino.

Hemos señalado que el equipo envía el datagrama al encaminador de su red, lo que implica un conocimiento a priori de cómo llegar a este encaminador con el fin de poder enviarle el datagrama. La secuencia específica de acciones que realiza el equipo se puede ver en la figura 3 más detalladamente:

1) Se crea el datagrama con las direcciones de red origen y destino apropiadas.

2) Se busca la dirección de red del encaminador: próximo salto[6] (o del equipo final si está directamente conectado al encaminador: último salto[7]).

[6] En inglés, *next hop*.

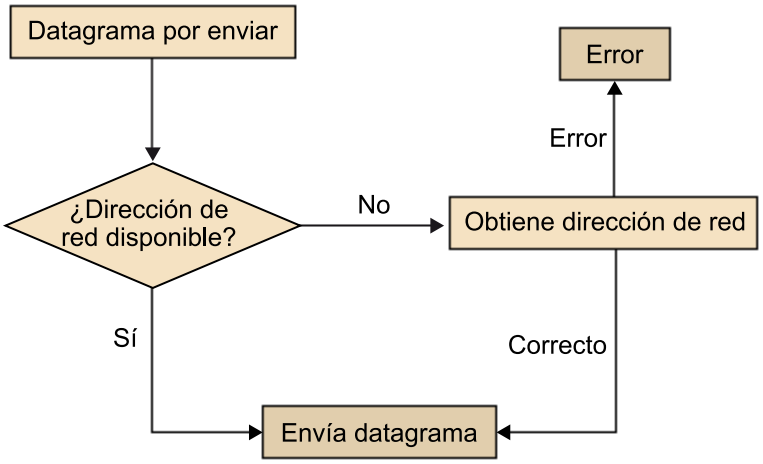
[7] En inglés, *last hop*.

3) Si no se dispone de la dirección de red, se parte con error, ya que no sabemos cuál es el próximo salto para enviar el datagrama.

4) Si hemos podido conseguir la dirección, enviamos el datagrama (con las direcciones de red origen y destino originales) al próximo salto del camino o al equipo final, si estamos en el último salto.

5) Se repite desde el paso 2 hasta que el datagrama llegue a su destino.

Figura 3. Diagrama de bloques simplificado del envío de un datagrama a un equipo de red



Un punto importante que hay que considerar es que el datagrama no sigue cualquier camino, sino que los encaminadores disponen de una tabla de encaminamiento[8] que indica por qué interfaz se deben enviar los datagramas en función de su destino. Para llenar estas tablas es necesario utilizar unos algoritmos de encaminamiento, que veremos con detalle en el apartado 4.

[8] En inglés, *forwarding table* o *routing table*.

2. Servicios de red

El servicio de red define las características que debe tener el transporte punto a punto de los datos en la capa de red. Así, se definen características como la fiabilidad, enviando la información, el orden de llegada de los paquetes, umbrales de retraso al hacer llegar la información al destino, la información de congestión en la red, etc., entre los diferentes emisores y receptores dentro de ésta.

Actualmente existen dos modelos de servicios de red claramente diferenciados: el modelo de circuito virtual y el modelo de datagrama. A continuación, se describen los dos, haciendo énfasis en el modelo de datagrama, dado que es el utilizado por el nivel de red propuesto por Internet y, por lo tanto, el más relevante en la actualidad.

2.1. Modelo de red en modo de circuitos virtuales

Los circuitos virtuales serán explicados con detalle en el módulo 4, ya que en general se consideran pertenecientes al nivel de enlace de datos. Este subapartado sólo realiza una breve introducción para comprender mejor la capa de red.

Ved también

Podéis ver los circuitos virtuales en el módulo "Nivel de enlace y redes de área local" de esta asignatura.

Un circuito virtual es un camino que se preconfigura entre dos puntos de la red, de manera que los nodos intermedios saben a priori la dirección a la que se debe enviar la información perteneciente a cada circuito.

Este paradigma permite acelerar enormemente el envío de paquetes entre dos puntos. Dado que el procesamiento intermedio es mínimo, esta prerreserva también permite garantizar una serie de recursos de red para el tráfico que pasa por el circuito. Por ello, este modelo de red se pensó para servicios en tiempo real (multimedia).

En cualquier circuito virtual se pueden distinguir tres fases claramente diferenciadas:

1) Establecimiento del circuito virtual: esta fase se inicia en la capa de red del emisor, utilizando la dirección del receptor. El emisor envía un datagrama de creación de circuito que provoca que cada nodo intermedio reserve los recursos pedidos de manera iterativa hasta llegar al destino. Cada uno de los nodos intermedios deberá actualizar su estado para acomodar el nuevo circuito, o denegar la creación en el caso de que no queden más recursos disponibles

(normalmente ancho de banda). Si el establecimiento del circuito puede llegar hasta el destinatario, se avisa al emisor indicando que la conexión ha sido satisfactoria y que se puede empezar a enviar información.

2) Transferencia de datos: en el caso de que se haya podido establecer el circuito virtual, se puede empezar a enviar datos entre los dos puntos.

3) Desconexión del circuito virtual: esta desconexión puede ser iniciada tanto por el emisor como por el receptor, y se avisa secuencialmente mediante la capa de red a todos los nodos intermedios hasta llegar al otro extremo. Esta desconexión permite liberar los recursos ocupados por el circuito.

Ejercicios

1. ¿Qué diferencias creéis que pueden existir entre el inicio de un circuito virtual en la capa de red y el establecimiento de una conexión en la capa de transporte? (por ejemplo, el *3 way handshaking*).

Solución ejercicio 1

El establecimiento de la conexión de la capa de transporte involucra únicamente a dos sistemas finales. Los dos extremos acuerdan la comunicación y determinan los parámetros de conexión, mientras que los nodos intermedios de la red no intervienen en ella. En contraposición, el establecimiento de un circuito virtual en la capa de red obliga a involucrar a todos los nodos intermedios.

2. Indicad tres tecnologías de red que utilicen circuitos virtuales.

Solución ejercicio 2

ATM, Frame relay y X.25. Muchos autores han considerado y consideran ATM y Frame relay como tecnologías de nivel de enlace; en cualquier caso, esta consideración se realiza por el hecho de que ambas pueden transportar tráfico IP (el tráfico presente en Internet).

El principal inconveniente que tiene la utilización de circuitos virtuales es que los nodos intermedios deben mantener las reservas de recursos solicitadas, independientemente de que se estén utilizando o no, lo que implica el posible problema de infrautilizar la red.

2.2. Modelo de red en modo datagrama

Si enviar información a través de un circuito virtual implica previamente establecer un camino y reservar recursos, en una red en modo datagrama (también denominada **conmutación de paquetes**) el paquete se envía directamente a la red con una dirección origen y una dirección destino. Por lo tanto, es trabajo de la red (mediante las tablas de encaminamiento de cada encaminador) hacer llegar el paquete a su destino.

Como se puede comprobar, en este tipo de comunicación no existe ni reserva de recursos ni camino preestablecido entre los extremos de la comunicación. Por lo tanto, a un encaminador pueden llegar datagramas de diferentes des-

tinos a la vez y los datagramas pueden seguir caminos diferentes para llegar al destino (según los algoritmos de encaminamiento). Esto provoca el efecto colateral de que los paquetes puedan llegar fuera de orden (por ejemplo, que el paquete número 2 llegue antes que el número 1).

Las redes en modo datagrama son las más usadas actualmente, sobre todo porque el protocolo de red de Internet (IP) lo utiliza.

Aunque, como hemos visto, el modelo de datagrama realiza un uso de los recursos más eficiente, lo que implica un coste. Con este tipo de redes se complica muchísimo la priorización del tráfico, ya que nunca se sabe a priori cuánto tráfico se recibirá y, lo que es más grave, no se sabe qué prioridad se debe dar a cada uno de los flujos de datos presentes en la red, más aún teniendo en cuenta que Internet se basa en el conocido paradigma ***best effort***, que implica que la red no nos da ninguna garantía de calidad y que "lo hará lo mejor que pueda" para hacer llegar el datagrama a su destino.

Ejercicios

3. ¿Cuál de los dos modelos de red vistos consideráis que efectúa un uso de los recursos más eficiente?

Solución ejercicio 3

El hecho de que un circuito virtual obligue a hacer una prerreserva de recursos implica que previamente se conozca el modelo y el patrón de tráfico que sigue la aplicación. Pero como muchas veces eso no es posible a priori, se suele a hacer lo que se conoce como *overprovisioning* (reservar más recursos de los que se consideran necesarios), lo que inequívocamente lleva a un sistema menos eficiente en términos de recursos.

Por su parte, utilizar el modo datagrama no implica ninguna prerreserva, por lo que la red siempre enviará tan rápido como pueda la información, siempre y cuando existan recursos disponibles.

2.3. Servicio de red orientado y no orientado a la conexión

Al igual que en los protocolos de transporte que hemos visto anteriormente, en el nivel de red también podemos tener protocolos orientados a conexión y otros que no lo sean. La principal diferencia entre las dos alternativas es que en el servicio orientado a conexión se conserva el estado de la conexión o, lo que es lo mismo, se tiene conocimiento de todas las conexiones establecidas, mientras que en el caso del servicio no orientado a conexión, no se tiene constancia de las conexiones existentes. Un ejemplo claro de servicio de red orientado a la conexión es el modelo de circuitos virtuales que ya hemos visto.

Debemos señalar que el diseño de un protocolo de red no orientado a conexión no excluye que a niveles superiores (transporte) se pueda definir un protocolo orientado a conexión. El ejemplo más indicativo de esto es la pila de protocolos TCP/IP[9], en la que TCP está orientado a la conexión e IP no. De

[9]TCP es la sigla de *transmission control protocol*. IP es la sigla de *Internet protocol*

hecho, la arquitectura actual de Internet sólo proporciona el modelo de servicio de datagrama, que no garantiza el orden de los paquetes, el no-retraso en el envío ni la llegada del datagrama.

3. Direccionamiento en Internet. El protocolo IP

El protocolo de capa de red por excelencia es *Internet protocol*. IP es un protocolo basado en el intercambio de información con el modelo no orientado a conexión. También es el protocolo utilizado en Internet para identificar los nodos de la red, así como para enviar la información de una manera estándar e independiente de la tecnología de red utilizada (podéis encontrar más información sobre tecnologías de red en los apartados siguientes). Otra característica muy importante es que IP no implementa mecanismos que garanticen la integridad de los datos que se envían por la red (esto se lleva a cabo en la capa de transporte); sólo se verifica que no se produzcan errores de transmisión en la cabecera.

Todos los protocolos de red requieren algún mecanismo con el fin de identificar los nodos de la red. Esta identificación en el protocolo IP se realiza mediante la dirección IP.

Actualmente, existen dos versiones diferentes del protocolo IP: IPv4 e IPv6. IPv4 es el protocolo más utilizado en Internet, pero dado el gran crecimiento que ha experimentado la red, se ha propuesto una extensión, IPv6, más actual y que se prevé que algún día sustituya a IPv4. En los siguientes subapartados se detalla cómo funcionan los dos protocolos y qué ventajas e inconvenientes tienen.

3.1. IPv4

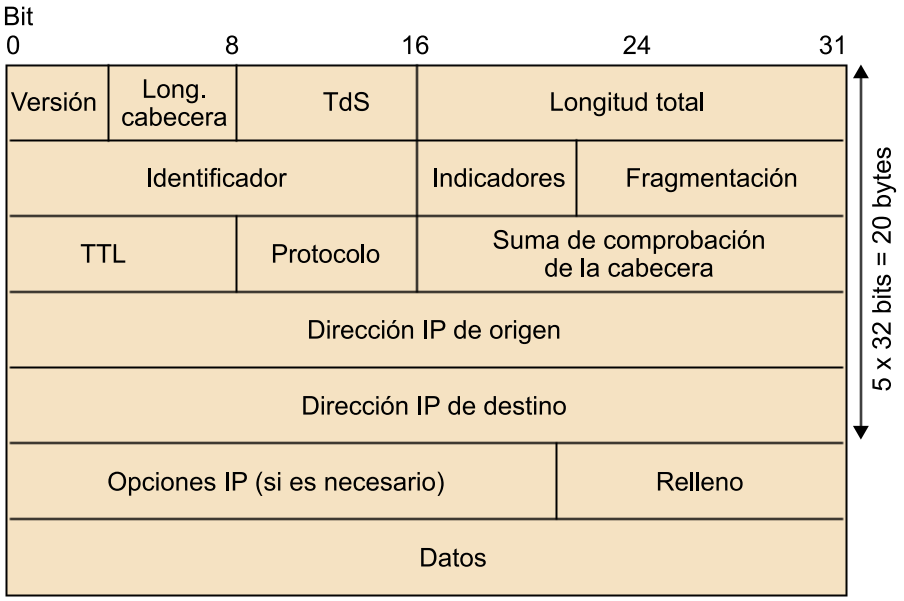
IPv4 fue propuesto en 1981 y todavía es el protocolo de red por excelencia. IPv4 define el formato que se debe utilizar para enviar información entre dos puntos distantes de la red. El protocolo proporciona mecanismos que determinan cómo se divide el direccionamiento de un modo escalable en una red tan grande como Internet.

3.1.1. La cabecera IP

IPv4 define qué información de control y qué formato deben tener los paquetes que se envían a la red. Por ello, al igual que sucede con los protocolos de transporte vistos en el capítulo anterior, es necesario definir una cabecera que sirva para poder identificar los paquetes.

La cabecera de IPv4 se puede ver en la figura 4.

Figura 4. Cabecera IPv4



Partes de la cabecera IPv4:

- **Versión** [4 bits]: indica qué protocolo de red utiliza este datagrama. Para IPv4 está fijado en 0x04.

- **Longitud de cabecera** [4 bits]: la cabecera IP puede tener una medida variable a causa del campo de opciones. Esta medida indica en qué punto empiezan los datos del protocolo de transporte. En particular, este campo indica el valor en función de la cantidad de palabras de 4 bytes que tiene la cabecera; así, un valor de 0x05 significa una cabecera de 20 bytes, que es el valor usado en la mayoría de los casos por ser la medida por defecto cuando no hay opciones.

- **Tipo de servicio (TdS)** [8 bits]: este campo permite distinguir entre diferentes tipos de datagramas IP. En un principio, se definieron parámetros en función de: retraso bajo, tasa de transferencia alta o fiabilidad. Así, en función del tipo de tráfico que contenga el paquete (por ejemplo, tráfico interactivo), puede quererse un retraso bajo o de coste mínimo (cuando el tráfico sea de baja prioridad). En realidad, los encaminadores suelen ignorar este campo y utilizan la técnica *best effort* para encaminar los paquetes.

- **Longitud total** [16 bits]: indica la medida total del datagrama en bytes, lo que incluye la cabecera y el campo de datos. Los 16 bits indican una medida máxima del datagrama de 65.535 bytes. Aunque, en general, la medida máxima utilizada es 1.500 bytes.

- **Identificador** [16 bits], **indicadores** [3 bits] y **fragmentación** [13 bits]: estos campos hacen referencia a lo que se conoce por fragmentación IP. La fragmentación será explicada más adelante en esta misma sección.

- **TTL** [8 bits]: inicialmente, este campo hacía referencia al tiempo de vida del datagrama en milisegundos. Sin embargo, en la práctica contiene el máximo número de encaminadores que puede atravesar el paquete hasta que llegue a su destino. En cada salto, un encaminador decrementa con 1 el valor de este campo; cuando el TTL llega a 0, el paquete es descartado. Con esta técnica se permite descartar datagramas en el caso de que haya algún bucle provocado por algún problema con el sistema de encaminamiento, y así evitar tener paquetes en la red más tiempo del necesario. De este campo se puede derivar que el "diámetro" máximo posible de Internet es de 255 saltos. No obstante, en la actualidad no se suelen superar los 30.

- **Protocolo** [8 bits]: este campo indica el protocolo presente en la capa de transporte, que será capaz de interpretarlo. Normalmente, este campo puede ser 0x06 para TCP o 0x11 para UDP[10]. La lista completa se puede encontrar en Internet. Este enlace entre la capa de red y la de transporte permite tener varios protocolos de transporte y poder distinguirlos fácilmente. Para ello se pasa el control al correspondiente de una manera eficiente.

(10)UDP es la sigla de *user datagrama protocol*.

- **Suma de comprobación de la cabecera** [16 bits]: permite detectar algún tipo de error de transmisión en la cabecera. Es importante señalar que no se comprueba la integridad de las capas de transporte y superiores. Recordemos que IP no garantiza la recepción de los datos. La suma de comprobación[11] se calcula tratando como enteros cada dos bytes de la cabecera y sumándolos mediante una aritmética de complemento a 1, ignorando para la suma el mismo campo que contiene la suma de comprobación. La integridad se comprueba al comparar la suma con la almacenada en la cabecera. En el caso de error, el paquete se descarta. Un pequeño inconveniente de esta suma de comprobación es que cada encaminador debe recalcularlo para cada paquete, dado que el campo TTL (y quizá algunas opciones) cambia en cada salto.

(11)En inglés, *checksum*.

- **Dirección de origen** [32 bits]: indica la dirección de origen del paquete. Se puede encontrar más información sobre el direccionamiento más adelante.

- **Dirección de destino** [32 bits]: a donde va dirigido el paquete.

- **Opciones IP**: este campo facilita que la cabecera IP sea variable en tamaño. Las opciones, que normalmente no se utilizan, permiten ampliar las funcionalidades de la cabecera IP. A pesar de no emplearse casi nunca, el hecho de comprobar la existencia en cada encaminador baja mucho el rendimiento del protocolo IPv4. Por ello, durante el diseño de la versión 6 del protocolo se cambió el modo de implementar estas opciones.

- ***Padding* (relleno)**: por motivos de eficiencia los datos deben empezar en una posición múltiple de 4 bytes, por lo tanto, en caso de que algunas op-

ciones introduzcan una desalineación, el *padding*, que normalmente son todo ceros, alinea la palabra del siguiente campo.

- ***Data* (*payload*)**: los datos del datagrama que se pasarán al nivel de transporte, o la información que realmente se quiere transmitir.

Fragmentación IP

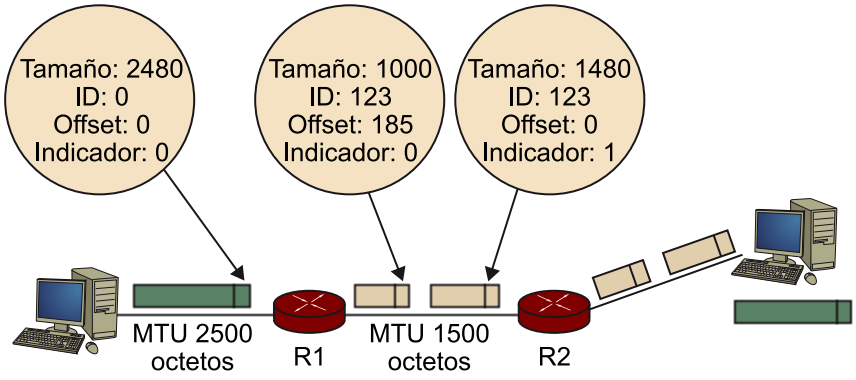
Durante el diseño del protocolo, uno de los puntos más críticos IP fue la necesidad de introducir la fragmentación. La fragmentación IP es necesaria porque, como se ve en el módulo siguiente, ni todas las redes ni todos los protocolos de enlace de datos pueden transportar paquetes de tamaño arbitrario. En general, el tamaño máximo estará delimitado en función de la tecnología de red utilizada. Por lo tanto, a causa de la variedad de tecnologías que coexisten en Internet actualmente, nos podemos encontrar casos en los que el tamaño máximo de trama permitida, MTU[12], sea menor en algún encaminador dentro del camino que deben seguir los datagramas, lo que fuerza a IP a dividir la trama en fragmentos menores que puedan ser transmitidos. Un ejemplo de esto puede ser Ethernet, que permite tramas de tamaño máximo de 1.500 bytes, mientras que una tecnología como ATM tiene, en general, un máximo de 9.180 bytes. Debemos señalar que cuando un datagrama IP se fragmenta, cada fragmento se debe autocontener y ser ensamblado en el destino final (hacerlo en los encaminadores intermedios supondría una pérdida de rendimiento considerable). Por lo tanto, no es suficiente dividir el datagrama; es necesario hacer algún tipo de proceso.

(12)MTU es la sigla de *maximum transfer unit*.

Cuando se debe dividir un datagrama IP, primero se repite la cabecera IP para cada fragmento y, a continuación, se actualizan los campos de la cabecera *identification*, *flag* y *fragmentation offset*. Así, todos los fragmentos pertenecientes al mismo datagrama tendrán el mismo identificador, cada fragmento contendrá el desplazamiento y, por último, un indicador[13], que será 1 si hay más paquetes o 0 si es el último. Por restricciones con la implementación, y para reducir el número de bits que se utilizan para almacenar este desplazamiento, se decidió emplear múltiplos de 8 bytes. De este modo, un desplazamiento de 64 bytes (es decir, que el fragmento IP contiene desde el byte 65 del datagrama original) se representará con un 8 en el campo *fragmentation offset* (ya que $8 \times 8 = 64$). La figura 5 muestra un ejemplo de esto.

(13)En inglés, *flag*.

Figura 5. Ejemplo de fragmentación



En la figura se puede ver un caso en el que un equipo envía un paquete de tamaño MTU 2.500 = bytes. Esto significa que el paquete tendrá 2.480 bytes de información útil y 20 de cabecera. En el momento de fragmentar, se generan dos paquetes diferentes, uno de 1.480 + 20 y otro de 1.000 + 20. Como se puede ver, el tamaño útil no cambia, pero el hecho de tener dos paquetes diferentes supone la repetición de la cabecera. El valor del identificador viene dado por un contador interno en el encaminador que fragmenta. El desplazamiento[14] para el primer fragmento es 0 y el indicador 1; esto indica que todavía existen más fragmentos. En el segundo fragmento el desplazamiento contiene un 185, ya que se especifica con grupos de 8 bytes, y un 0 en el indicador, lo que indica que se trata del último fragmento del datagrama original. Cuando el datagrama llegue a su destino final será reensamblado y pasado a los niveles superiores de modo transparente.

[14]En inglés, *offset*.

Ejercicios

4. ¿Por qué creéis que se realiza una verificación de la suma de comprobación tanto en el nivel de red como en el nivel de transporte?

Solución ejercicio 4

Cada una de las verificaciones tiene funcionalidades diferentes. El objetivo de IP es enviar el paquete a un destino; por lo tanto, valida que la cabecera sea correcta. Si se produce un error en las capas superiores, se encargarán de realizar la verificación. Otro motivo es la eficiencia. Si cada encaminador ha de verificar que todo el *payload* es correcto, se crea demasiado coste computacional. Por otra parte, TCP, como ya hemos visto, debe garantizar que los datos llegan correctamente, para lo que se realiza una verificación de todo el *payload* directamente en el destino. Esto evita sobrecargar innecesariamente la red.

5. ¿Qué implicaciones tiene el hecho de tener un campo de opciones en la cabecera IP?

Solución ejercicio 5

El hecho de tener opciones en la cabecera IP implica que tenga un tamaño variable (con funciones opcionales), lo que a su vez obliga a tener más campos en la cabecera, así como más procesamiento en los encaminadores intermedios (ya que algunas de las opciones necesitan ser procesadas por el mismo encaminador).

6. Conociendo el tamaño de las cabeceras TCP e IP, ¿cuál es la eficiencia máxima en la transmisión que podemos conseguir utilizando estos protocolos?

Solución ejercicio 6

La eficiencia en la transmisión se puede calcular observando cuántos bits útiles se envían respecto al total. Así:

$$E_t = 1 - \frac{H_{IP} + H_T}{M}$$

donde H_{IP} es el tamaño de la cabecera IP, H_T es el tamaño de la cabecera de transporte y M es el tamaño total del datagrama. En el caso de TCP/IP y un tamaño de datagrama de 1.500 bytes, la eficiencia es:

$$E_t = 1 - \frac{20 + 20}{1.500} = 0{,}973 = 97{,}3\%$$

o, lo que es lo mismo, tenemos una penalización del 2,6% en el envío de información.

7. Un paquete sigue un camino en el que las MTU son: 5.000 y 1.500. Un equipo envía un paquete de 5.000 bytes. ¿Cómo se fragmentará y qué desplazamiento e indicadores contendrá cada fragmento?

Solución ejercicio 7

Como la menor MTU es de 1.500 bytes, el datagrama se dividirá en cuatro fragmentos, que deben contener un total de 4.980 bytes divididos del siguiente modo: 1.480, 1.480, 1.480 y 560 bytes, respectivamente. Así, los desplazamientos serán: 0, 1.480/8, (2 × 1.480)/8 y (3 × 1.480)/8, o sea: 0, 185, 370 y 555. Finalmente, los indicadores serán los 1, exceptuando el último paquete, que contendrá un 0.

3.1.2. Direccionamiento IPv4

Como ya se ha comentado, los protocolos de red necesitan disponer de una dirección única que permita identificar todos los nodos de la red. En el caso de IPv4, tal como puede deducirse de la cabecera IPv4, la máxima cantidad de direcciones disponibles es muy grande 2^{32} (4.294.967.296).

Para simplificar su escritura, se dividen los 32 bits en 4 bloques de 8 bits cada uno; además, en lugar de utilizar la representación binaria, que es poco legible, en la práctica una dirección IP se escribe en notación decimal, separada por puntos. Una dirección estará formada por 4 bloques de números entre 0 y 2^8 – 1 (255). Por ejemplo:

Representación binaria y decimal de una dirección IP

```
10001111 00101101 00000001 00010111
   143   .   45   .    1   .   23
```

Asimismo, tener un número tan grande de direcciones supone un enorme problema de gestión, por lo que se propuso un sistema de asignación de direcciones jerárquico. En Internet, las direcciones IP están compuestas por dos partes, la parte de red y la parte del equipo, que se utiliza para poder estructurar las direcciones y organizarlas por zonas administrativas.

La parte de red está formada por los bits superiores de la dirección IP, e indica a qué red pertenece un conjunto de equipos o, lo que es lo mismo, quién es el encaminador de salida del conjunto de equipos. Por el contrario, la parte del equipo son los bits inferiores de la dirección IP, que identifican el equipo dentro de su red. Inicialmente, esta división con redes se elaboró mediante clases; concretamente se definieron 5 clases diferentes (A, B, C, D y E), tal como muestra la siguiente tabla:

División de redes por clases			
Clase	**Bits iniciales**	**Bits red**	**Rango red**
A	0	7 (+ 1)	1.0.0.0 – 127.0.0.0
B	1 0	14 (+ 2)	128.0.0.0 – 191.255.0.0
C	1 1 0	21 (+ 3)	192.0.0.0 – 223.255.255.0
D	1 1 1 0	-	224.0.0.0 – 239.0.0.0
E	1 1 1 1 0	-	240.0.0.0 – 255.0.0.0

Las direcciones de clase A son las destinadas a grandes empresas, como IBM, o a grandes operadoras americanas, como AT&T WorldNet Services. Proporcionan acceso a 2^{24} (16.777.216) equipos por red, donde 8 bits están destinados a identificar la red y el resto, hasta los 32, se utiliza para los equipos finales. Existe un total de 2^7 (255) direcciones de clase A. Como veremos más adelante, este reparto de clases A es uno de los causantes de la fuerte carencia de direcciones IP en la actualidad.

Las direcciones de clase B son las que se dan a grandes entidades, universidades y determinados proveedores de Internet. Permiten repartir 2^{16} (65.536) equipos por red, y existe un total de 2^{14} (16.384) direcciones de clase B.

Las direcciones de clase C se destinan a medianas empresas con fuerte presencia en Internet. En este caso, se dispone de 2^8 (256) direcciones, con un total de 2^{21} (2.097.152) direcciones de tipo C para repartir.

Por su parte, las direcciones de clase D se consideran un tipo de clase especial, denominadas clases multidifusión, y sirven para enviar tráfico punto multipunto. Detallaremos este tipo de tráfico más adelante.

Por último, las direcciones de clase E están reservadas para un uso futuro.

Direcciones de propósito específico

Aparte de la división en redes, también se destinaron una serie de direcciones de propósito específico para casos especiales, como las direcciones de equipo final, las direcciones de red, las direcciones de difusión[15], las direcciones de *loopback* y las direcciones privadas.

(15)En inglés, *broadcast*.

- Las direcciones de equipo final indican un equipo dentro de la red actual y tienen la forma 0.**host**, donde host es la parte del equipo de la red actual, es decir, que la parte de la dirección de red es todo 0.

- Las direcciones de red hacen referencia a la red, pero no a los equipos dentro de ella; las direcciones de red pertenecen al modo **red**.0 para direcciones de clase C, **red**.0.0 para la clase B y **red**.0.0.0 para la clase A, o, en otras palabras, toda la dirección del equipo de red está en 0. Un caso especial es la dirección 0.0.0.0, que indica "este equipo final" de "esta red", aunque no siempre se implementa en los sistemas operativos actuales.

- Las direcciones de difusión indican todos los equipos de una red concreta. La dirección se representa con **red**.255 para direcciones de clase C. Como en el caso de las direcciones de red, las de clase B serán **red**.255.255 y las de clase A, **red**.255.255.255; es decir, que la dirección del equipo de red es todo 1. Siempre que se reciba un datagrama en la dirección de difusión, todos los equipos deben responder a aquél. Asimismo, las direcciones de difusión tienen una dirección especial, la 255.255.255.255, que hace referencia a toda la red (Internet). Por lo tanto, si alguien enviara un datagrama a la dirección 255.255.255.255, Internet completa debería responder. Como ello provocaría graves problemas de escalabilidad y exceso de tráfico, no existe ningún encaminador que reenvíe tráfico difusión por sus interfaces. El tráfico de difusión siempre se quedará en la red que lo ha emitido.

Ejercicios

8. Dada la dirección IP 120.1.32.54, indicad cuál es la dirección de red, la dirección del equipo final y la dirección de difusión de la red.

Solución ejercicio 8

La dirección 120.1.32.54 forma parte de las direcciones de clase A, por lo tanto, la dirección de red será 120.0.0.0; la del equipo final, 0.1.32.54; y la de difusión, 120.255.255.255.

- Las direcciones de *loopback* van desde la 127.0.0.0 hasta la 127.0.0.255, y son las que utilizan de manera interna los equipos. Cuando un equipo arranca, automáticamente crea una interfaz virtual (interfaz de *loopback*) para uso interno del sistema operativo; por lo general, sólo se utiliza la 127.0.0.1.

- Las direcciones privadas son las que emplean redes locales internas que no salen en Internet. La lista con todos los rangos privados se puede encontrar en la siguiente tabla. Como se puede observar, se pueden configurar internamente varios rangos de direcciones privadas; su función es evitar colisiones en asignaciones de direcciones en configuraciones internas con otros nodos de otras redes del exterior. Otro uso es permitir la asignación de más direcciones a nuestras redes, que IP pública asignadas por los operadores.

Lista de rangos de direcciones privadas		
Clase	**Rango red**	**Número de subredes**
A	10.0.0.0 - 10.255.255.255	1
B	172.16.0.0 - 172.31.255.255	16
C	192.168.0.0 - 192.168.255.255	255

El principal problema de las direcciones privadas es que no pueden acceder a Internet directamente; los encaminadores nunca enviarán a Internet el tráfico originado o con destino a direcciones privadas, ya que el próximo salto no sabe cómo encaminarlo. Para evitar esta limitación, y permitir la transferencia de datos entre direcciones privadas y públicas, los encaminadores incluyen una técnica denominada NAT[16].

(16)NAT es la sigla de *network address translation*.

NAT

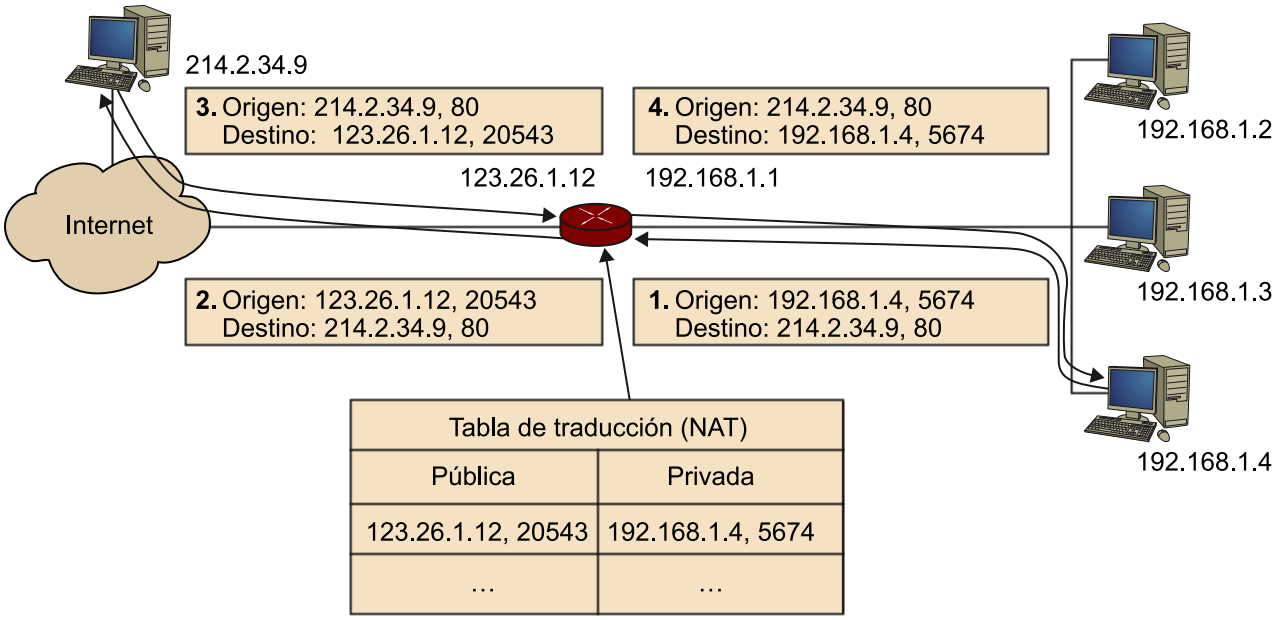
Por lo que se puede deducir de lo visto hasta ahora, cada equipo de una red IPv4 debe disponer de una IP pública para poder acceder a la red. Uno de los problemas principales que se encuentran cuando se piden IP a las operadoras es que, generalmente, el usuario (o la empresa) tiene más equipos que IP asignados. Un ejemplo de ello es aquel usuario con conexión ADSL que recibe un único IP por parte de la compañía telefónica y que, sin embargo, posee varios equipos, como un PC de sobremesa, un portátil, una PDA, etc. Para permitir que todos los equipos se puedan conectar a la red al mismo tiempo, existen dos opciones: pedir más IP (solución difícil y cara) o utilizar direcciones IP privadas y configurar el encaminador para que haga la conversión desde la dirección IP privada a la IP pública disponible. Esto se puede conseguir mediante la NAT.

Ejemplo de utilización de NAT

Si un cliente con dirección privada quiere establecer una conexión con un equipo que tiene una dirección IP pública (punto 1 de la figura 6); por ejemplo, un servidor, en ese caso, el cliente enviará el paquete al encaminador de su red. Éste tendrá configurada una tabla de traducción que transformará la IP origen del datagrama en una IP pública que tenga reservada a tal efecto. Para completar la traducción, el encaminador mapeará el puerto origen (de la capa de transporte) a un puerto nuevo origen asignado por el encaminador. El punto 2 de la figura 6 muestra un ejemplo de esto en el que el encaminador transforma la IP origen (192.168.1.4) y el puerto origen (5.674) del equipo en la IP pública del encaminador (123.26.1.12) y un puerto asignado dinámicamente (20543 en el ejemplo). La estación destino ve un datagrama como si hubiera sido enviado por el encaminador, a lo que le responde de la manera habitual, usando TCP/IP. Finalmente, el encaminador, al recibir la respuesta, mira la tabla de traducción y deshace el cambio para

enviar el paquete final a la estación origen. Si la entrada no hubiera estado en la tabla, el encaminador habría asumido que el paquete iba realmente dirigido a él.

Figura 6. Ejemplo de red con NAT



Este mecanismo es muy útil para ahorrar el uso de direcciones públicas, a pesar de tener una serie de inconvenientes que lo hacen inútil en determinados entornos. Por un lado, existen protocolos de aplicación (por ejemplo, FTP) que incrustan la IP del cliente en el datagrama; esta IP es usada por el servidor para establecer una nueva conexión (es el caso del FTP activo), y como el cliente incrusta la IP privada se impide que se pueda establecer la conexión.

Otro problema importante es que todas las conexiones deben iniciarse desde el equipo con una IP privada, ya que el encaminador debe establecer la entrada en la tabla de traducción antes de poder enviar información al equipo con IP privada. Por esta razón, normalmente no se pueden tener servidores con IP privadas. Debemos señalar, sin embargo, que esto se puede solucionar con una técnica denominada PAT[17], donde el encaminador tiene configurado de manera estática un mapeo, por lo que cuando llega un datagrama a un puerto concreto, automáticamente reenvía el paquete hacia el equipo con IP privada que esté configurado. En según qué entornos el PAT se conoce también como DNAT[18] o incluso como puerto *forwarding*, pero la idea de fondo es la misma.

[17]PAT es la sigla de *port address translation*.

[18]DNAT es la sigla de *destination network address translation*.

3.1.3. CIDR

Una vez definidas las diferentes clases de redes, observamos que esta solución es claramente insuficiente, ya que igualmente forzaba a las operadoras grandes y medianas (con clases A y B) a gestionar desde un solo equipo un número de direcciones demasiado elevado. Por esta razón se propuso el CIDR[19].

[19]CIDR es la sigla de *classless inter domain routing*.

CIDR propone un mecanismo más flexible para poder subdividir nuestras redes. Es importante señalar que CIDR no sustituye la división por clases, que continúan siendo las unidades básicas de asignación de direcciones, sino que divide las direcciones asignadas en subredes más pequeñas y manejables.

Así, con CIDR la separación entre el equipo y la red se consigue mediante una máscara. Esta máscara tiene la forma de una dirección IP, que enmascara los bits de una dirección normal para poder distinguir el equipo y la red de manera sencilla. Por ejemplo, una máscara de 255.255.255.0 permite separar la dirección de red de la del equipo final haciendo AND con la dirección IP. Así:

```
143 . 45  .  1  . 23
255 . 255 . 255 . 0
143 . 45  .  1  . 0
```

De donde se puede extraer la dirección de la subred (los unos de la máscara – 143.45.1) y la dirección del equipo (los ceros de la máscara – 23). En este caso, la red constará de 2^8 IP válidas, como una clase C, de las que $2^8 - 2$ serán asignables a equipos. Debemos recordar que las direcciones especiales de red y de difusión no son asignables (143.45.1.0 y 143.45.1.255, respectivamente). La representación de esta subred emplea la siguiente nomenclatura: 143.45.1.23/255.255.255.0.

Como se puede observar, esto nos da un nivel más fino de división que nos simplificará mucho la gestión interna de redes. Si una entidad dispone de una clase B (146.43.0.0), internamente la entidad puede decidir subdividir las 65.536 direcciones en varias subredes, por ejemplo, con 256 subredes de 256 IP cada una: desde la 146.43.0.0/255.255.255.0 a la 146.43.255.0/255.255.255.0. Observad que los valores de 255 y 0 para la dirección de red son correctos, sin representar direcciones de difusión y de red, respectivamente. Esto lo podemos saber gracias a la máscara.

Una restricción no escrita, pero generalmente adoptada a la hora de definir las máscaras, es que todos los unos de la máscara deben ser consecutivos. Así, máscaras como 255.145.0.0 se consideran inválidas, ya que traducidas a formato binario sería:

```
11111111 10010001 00000000 00000000
```

Mientras que otras como 255.255.128.0 son totalmente correctas, ya que en formato binario resulta:

```
11111111 11111111 11111110 00000000
```

Donde todos los unos son consecutivos, a pesar de no estar alineados al byte.

Esta restricción de los unos consecutivos nos permite simplificar la representación de la máscara a un formato más compacto. En este sentido, otro modo de indicar la separación entre la red y el equipo se realiza mediante un formato que indica cuántos bits representan la red; por ejemplo, 143.45.1.23/24 indica que la máquina 143.45.1.23 pertenece a la red 143.45.1.0/255.255.255.0 o, lo que es lo mismo, que tiene 24 bits para la dirección de red y 8 para la de los equipos.

Por otra parte, gracias a la clasificación para subredes, los encaminadores tienen el trabajo más fácil, ya que para poder decidir la ruta que debe tomar cualquier datagrama basta con mirar la red de destino, y no es necesario comprobar toda la dirección IP. Idealmente, la dirección IP entera sólo la mirará el último encaminador de la cadena, o sea, el que esté dentro de la misma subred a la que pertenezca aquel destino. Para implementar este mecanismo, los encaminadores basan la decisión de encaminamiento en una política denominada *longest prefix match*, que significa que entre todas las rutas posibles siempre se elige la que tiene más bits coincidentes con el destino del paquete.

Ved también

Podéis ver más información sobre la política del *longest prefix match* en el apartado 5 de este módulo didáctico.

Ejercicios

9. De las siguientes subredes e IP, indicad cuántas IP asignables puede contener la red y explicad su significado:

Dirección	IP asignables	Explicación
147.83.32.0/24		
1.23.167.23/32		
1.23.167.0/32		
147.83.32.0/16		

Solución ejercicio 9

Dirección	IP asignables	Explicación
147.83.32.0/24	254	Es una dirección de red en la que tenemos 8 bits para los equipos. Teniendo en cuenta que la dirección de difusión y la de red no son asignables, nos encontramos con un total de $2^8 - 2$ direcciones para asignar.
1.23.167.23/32	1	Dirección con una subred y un único equipo. No es útil en un caso real pero es correcta.
1.23.167.0/32	1	Como no hay parte de equipo, todo es red. El hecho de que el último byte sea 0 no implica que la IP sea una dirección de red genérica, sino una específica, como el caso anterior.
147.83.32.0/16	1	Se refiere al equipo 32.0 de la red de clase B 147.83.0.0. Debemos señalar que no es una dirección de red, ya que no todos los bits de fuera de la máscara son 0. Por lo tanto, se trata como una dirección de equipo.

10. De la dirección de clase B 143.45.0.0/16, indicad qué subredes /20 se pueden crear y cuántos equipos contienen cada una.

Solución ejercicio 10

Red	Subred	Equipo		
143.45	SSSS	HHHH.HHHHHHHH	143.45.0.0	Clase B
143.45	0000	HHHH.HHHHHHHH	143.45.0.0/20	2^{12} – 2 equipos = 4094
143.45	0001	HHHH.HHHHHHHH	143.45.16.0/20	
143.45	0010	HHHH.HHHHHHHH	143.45.32.0/20	
143.45	0011	HHHH.HHHHHHHH	143.45.48.0/20	
…				
143.45	1111	HHHH.HHHHHHHH	143.45.240.0/20	

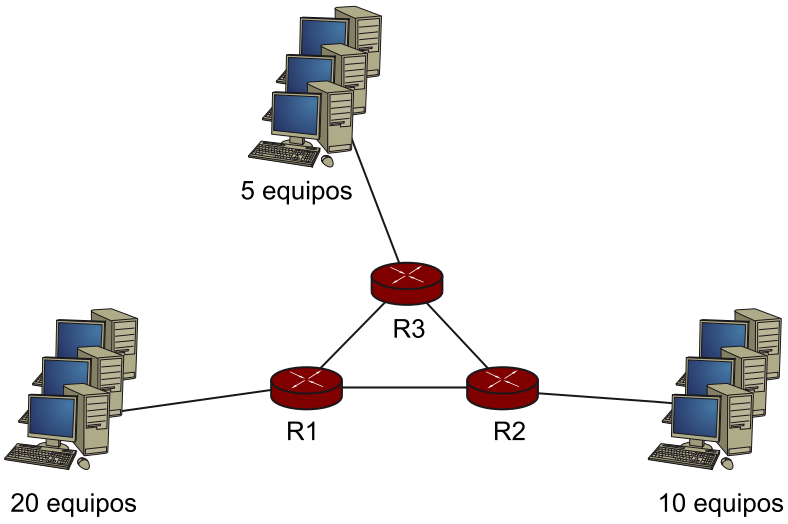
Generalmente, el CIDR se utiliza en conjunción con el VLSM[20], una técnica cuyo objetivo es optimizar el uso de las direcciones IP mediante una asignación inteligente de las máscaras de red. En este caso, la asignación se efectuará teniendo en cuenta el número de máquinas de cada subred y se asignarán máscaras de tamaño ajustado a las necesidades particulares de cada una. VLSM se puede ver como la creación de subredes de las subredes.

[20]VLSM es la sigla de *variable length subnet mask*. En castellano, máscara de subred de tamaño variable.

Ejercicios

11. Dada la red de la figura 7, se nos proporciona el rango de direcciones 147.83.85.0/24. Se pide que se asignen rangos de direcciones a todas las subredes y a los enlaces entre los encaminadores.

Figura 7



Solución ejercicio 11

La asignación de direcciones se puede llevar a cabo siguiendo la siguiente política:

- Los 20 equipos y el encaminador necesitan un total de 5 bits ($2^5 = 32$).
- Los 10 + 1 equipos tienen suficiente con 4 bits ($2^4 = 16$).
- Los 5 + 1 equipos necesitan 3 ($2^3 = 8$), con lo que esta subred no podrá crecer más.

- Por último, los enlaces punto a punto necesitarán 2 bits, ya que necesitamos espacio para las direcciones de difusión y de red.

  En resumen, necesitaremos un prefijo /27, otro /28, otro /29 y 3 prefijos /30, respectivamente.

  De este modo, una posible asignación sería:

  Los 20 equipos pueden usar 147.83.85.0/27, en la que el último byte sería: 000XXXXX. Con dirección de red 147.83.85.0 y dirección de difusión 147.83.85.31.

  Los 10 equipos dispondrán de 147.83.85.32/28, en la que el último byte será: 0010XXXX. Con dirección de red 147.83.85.32 y dirección de difusión 147.83.85.47.

  En el caso de los 5 equipos utilizaremos la subred 147.83.85.48/29, en la que el último byte será: 00110XXX. Con dirección de red 147.83.85.48 y dirección de difusión 147.83.85.55.

  Por último, los tres prefijos /30 (de los enlaces entre los encaminadores) se pueden dividir con el último byte 001110XX, 001111XX y 010000XX, respectivamente. Es decir, 147.83.85.56/30, 147.83.85.60/30 y 147.83.85.64/30. Con direcciones de red 147.83.85.56, 147.83.85.60 y 147.83.85.64. Con direcciones de difusión 147.83.85.59, 147.83.85.63 y 147.83.85.67.

12. ¿Qué problema tiene la asignación de direcciones efectuada en el ejercicio 11?

Solución ejercicio 12

El problema de esta asignación viene dado por el hecho de que se han ajustado demasiado el número de bits para cada subred y en caso de que crezca el número de equipos (especialmente en la subred de 5 equipos) nos tocaría redimensionar la red de nuevo con el coste que ello supone.

3.1.4. Tipos de datagramas IP

IPv4 especifica tres tipos de tráfico claramente diferenciados dentro de la red: unidifusión, difusión y multidifusión.

El tráfico unidifusión es el más común. La comunicación está formada por dos interlocutores que se intercambian información; a menudo estas conexiones se producen entre un cliente y un servidor, que a la vez puede tener conexiones unidifusión con otros clientes.

El tráfico difusión se basa en el envío de información a todos los equipos presentes en una subred. Como ya hemos visto, esto se puede conseguir enviando un paquete a una dirección que sea la dirección de red y todo 1 a la dirección del equipo final. Por ejemplo, para la red 126.76.31.0/24, la dirección de difusión sería 126.76.31.255. Para conseguir que la difusión sea realmente eficiente (que sólo se envíe un paquete y lo reciban todos los equipos de la subred), es necesario tener soporte del protocolo de enlace de datos, tal como veremos en el siguiente módulo.

Sin embargo, cabe señalar que normalmente enviar tráfico difusión requiere algún privilegio en la red (ser administrador). Además, los encaminadores en general no propagan este tipo de tráfico para evitar problemas de seguridad, como ataques tipo *denial of service* (DoS).

Finalmente, el caso del tráfico multidifusión se basa en el paradigma de enviar información desde un único origen a muchos destinos a la vez. La base del tráfico multidifusión es que el emisor no necesita conocer quiénes serán sus receptores (al contrario que la política de unidifusión, que requiere conocer a los interlocutores). Esto se consigue mediante lo que se conoce como grupos de multidifusión. Como ya hemos visto, la IANA[21] ha reservado las direcciones de tipo D a multidifusión. Éstas son las que van del rango 224.0.0.0 hasta el rango 239.0.0.0. En este grupo de direcciones existen unas cuantas reservadas a grupos multidifusión, conocidos como permanentes. La lista completa se puede encontrar en Internet.

[21]IANA es la sigla de Internet Assigned Numbers Authority.

En este sentido, si una estación concreta está interesada en recibir un contenido multidifusión, se suscribirá al servicio mediante el protocolo IGMP[22], que especifica el formato del paquete que se debe generar con el fin de poder registrarse en un grupo y poder recibir el contenido de éste. IGMP soporta dos tipos de paquetes, los de pregunta y los de respuesta. Normalmente, los de pregunta son unos paquetes dirigidos a todos los equipos donde los que tienen sesiones multidifusión activas responden. Así, los encaminadores (que deben tener soporte para multidifusión) pueden construir lo que se conoce como árbol multidifusión, que encamina los paquetes a sus destinos.

[22]IGMP es la sigla de Internet Group Management Protocol.

Ved también

Podéis ver más información sobre árboles multidifusión en el apartado 4 de este módulo didáctico.

La ventaja principal de multidifusión es que la información que se envía, en lugar de repetirse desde el origen una vez por destino, crea un árbol que minimiza el número de copias. Un ejemplo de ello se puede ver en la figura del subapartado 4.5.

Ejercicios

13. En un momento dado, un servidor de chat tiene un total de 80 clientes conectados por todo el mundo. Indicad qué número y de qué tipo son las conexiones que tiene abiertas este servidor.

Solución ejercicio 13

Dado que el chat es un protocolo que utiliza TCP/IP y que los clientes, a pesar de hablar entre ellos, pasan siempre por el servidor, se trata del típico escenario con 80 conexiones unidifusión entre los 80 clientes y el servidor.

14. Un administrador de la red 147.83.0.0/16 quiere enviar un paquete de difusión a la subred 147.83.20.0/24. Indicad qué dirección de destino tendría el paquete, cuántos paquetes se generarían y a cuántas máquinas como máximo podría llegar.

Solución ejercicio 14

Dado que la subred que se quiere enviar la difusión tiene 8 bits, se generará un único paquete con dirección destino 147.83.20.255 y que se recibiría como má-

ximo en 255 – 2 = 253 estaciones. Esto se debe a que la dirección 147.83.20.0 y la 147.83.20.255 están reservadas para la dirección de red y la de difusión, respectivamente.

3.1.5. El futuro de IPv4

Cuando se diseñó IPv4 se creía que su gran número de direcciones IP (2^{32}) sería suficiente para poder soportar el gran crecimiento que se esperaba de una red como Internet. Debemos recordar que Internet entró en funcionamiento en 1969 con el nombre de ARPANet, un proyecto subvencionado por el Departamento de Defensa de Estados Unidos. Esto provocó, cuando unos años después Internet se desarrollo en la red comercial, que el reparto de direcciones no se realizara equitativamente y que las grandes empresas americanas pudieran adjudicarse una gran cantidad de direcciones de clase A, dejando a países como China y otros, que se han desarrollado posteriormente, con muchas menos direcciones de las necesarias. Como referencia, Estados Unidos tiene alrededor de 1.500 millones de direcciones asignadas, mientras que China, con una población mucho más numerosa, sólo dispone aproximadamente de 200 millones. Para tener una idea, hoy España tiene asignadas cerca de 22 millones.

Con este paradigma, se comprende fácilmente que con la actual política de reparto de direcciones no tardarán en agotarse las direcciones IPv4 disponibles para asignar. Inevitablemente, esto implicará que Internet no pueda crecer más.

Con el fin de minimizar este problema se diseñó el NAT, que, como ya hemos visto, permite utilizar direcciones privadas para acceder a la red con una sola IP pública. Actualmente, países como China o India están haciendo un uso intensivo del NAT por falta de direcciones disponibles.

Como esta solución no es escalable e implica una serie muy importante de problemas a los proveedores de servicios, se llegó a la conclusión de que los 32 bits de direccionamiento del protocolo IPv4 eran insuficientes. Por ello se diseñó el protocolo IPv6, como veremos a continuación. Por motivos económicos, IPv6 no ha sido todavía implantado y si la demanda de direcciones IPv4 continúa al ritmo actual, se prevé que IANA asignará el último rango de direcciones IPv4 a mitad del 2011, y que las autoridades regionales agotarán las que tienen pendientes para asignar en el 2012. A buen seguro, esto forzará a muchos países a adoptar prematuramente IPv6. En este sentido, países como Japón, China, India y algunos de Sudamérica ya han adoptado el protocolo y utilizan algunas técnicas, como veremos más adelante, que permiten la interoperabilidad de los dos protocolos.

3.2. IPv6

La carencia de direcciones IPv4 que hemos visto antes incentivó el diseño de un nuevo protocolo de red, IPv6. En la actualidad, IPv6 está totalmente desarrollado, aunque todavía no es posible utilizarlo dentro de la red comercial, ya que los operadores todavía no han preparado sus equipos y tampoco han repartido las direcciones a sus usuarios. Esto, junto con la dificultad de implantar progresivamente esta nueva versión y sustituirla por la anterior, es lo que está retrasando su incorporación al ámbito comercial.

Este subapartado describe brevemente este protocolo y se remarcan las diferencias con la versión anterior, así como las novedades que incorpora. Para concluir, se describen los principales problemas surgidos a causa de la migración de IPv4 a IPv6.

3.2.1. Motivación

En el momento de iniciar el protocolo se pensó que no era necesario crear un protocolo entero, bastaría con realizar una adaptación de IPv4. Sin embargo, muy pronto se comprobó que para poder disfrutar de buenas optimizaciones, en comparación con la versión anterior, se necesitarían más cambios. Así, se optó por un diseño que tiene poco en común con la versión anterior.

El motivo principal que llevó a plantearse una nueva versión del protocolo era el limitado rango de direcciones que permite IPv4, que –aunque pueda parecer muy elevado– se mostró claramente insuficiente para la demanda del mercado en un futuro. IPv6 soluciona este problema al proponer un campo de direcciones de 128 bits.

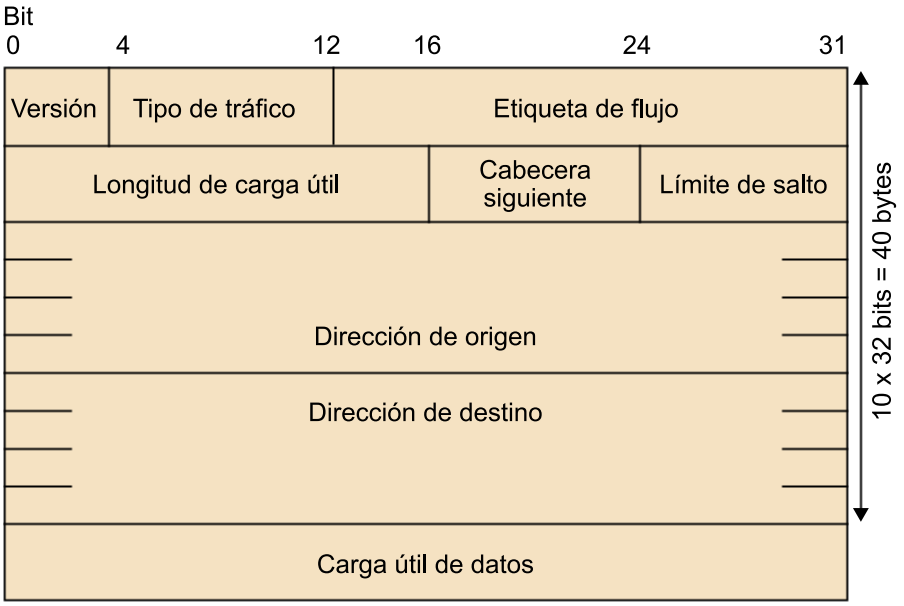
Fundamentalmente, la aparición durante los últimos años de una gran cantidad de dispositivos móviles que quieren formar parte de la gran red que es Internet ha provocado que rápidamente los 32 bits de direccionamiento IPv4 sean insuficientes. De hecho, si todos estos dispositivos se quisieran conectar simultáneamente a la red, es probable que los operadores tuvieran problemas por la falta de direcciones IPv4. Para ver este problema basta con comparar el número de teléfonos móviles existente en la actualidad sólo en el Estado español (unos 44 millones) con el de IP que tiene asignadas actualmente el país (unos 22 millones). Y eso sin considerar a los usuarios que se conectan desde sus hogares. Podríamos pensar que el empleo de NAT podría minimizar el problema, pero a la larga esta solución podría suponer un grave problema de rendimiento de los encaminadores en el mantenimiento de las tablas de traducción de direcciones de millones de conexiones a la vez. Además, cada vez existen más pequeñas y medianas empresas que quieren ofrecer a sus clientes

una serie de servicios que precisan una conexión permanente a la red, con el consecuente gasto de direcciones y la imposibilidad de utilizar el NAT masivamente.

3.2.2. Cabecera IPv6

La cabecera IPv6 tiene una longitud fija de 40 bytes (podéis ver la figura 8) y consta de los siguientes campos:

Figura 8. Cabecera IPv6



- ***Version*** (4 bits): indica la versión del protocolo que contiene el paquete. Este campo tiene el mismo significado que el de la versión IPv4, pero ahora con el valor 0 × 06.

- ***Traffic class*** (8 bits): este campo clasifica un paquete dentro de un tipo de tráfico determinado. Conceptualmente equivale al TOS de IPv4.

- ***Flow label*** (20 bits): sirve para etiquetar un conjunto de paquetes que posean las mismas características y para ofrecer calidad de servicio.

- ***Payload length*** (16 bits): longitud del *payload* del paquete, esto es, el paquete sin la cabecera IP. El tamaño viene representado en bytes.

- ***Next header*** (8 bits): este campo es una gran innovación de IPv6 respecto a IPv4, ya que permite tener una cabecera básica de tamaño fijo. Asimismo, indica la posición en la que se puede encontrar la siguiente cabecera, lo que permite ahorrar tiempo de proceso a los encaminadores intermedios al no haber opciones.

- ***Hop limit*** (8 bits): este campo es equivalente al TTL de IPv4, pero directamente cuenta saltos y no tiempo.

- ***Source address*** (128 bits): dirección del equipo final que ha originado el paquete.

- ***Destination address*** (128 bits): dirección del equipo final al que va destinado el paquete.

En los datos de la cabecera se puede observar que la diferencia más directa que existe entre ambos protocolos es la longitud de las direcciones IP: IPv4 tiene 32 bits, mientras que IPv6 pasa a tener 128. Este aumento en el espacio de direccionamiento permite que el rango de direcciones de la red pase de 2^{32} a 2^{128} direcciones posibles. Como ahora tenemos muchos más bits para la dirección, el modo de especificar direcciones IPv6 se lleva a cabo con "la notación de los dos puntos". Aquí, una dirección IPv6 se representa con bloques de 16 bits, representados en hexadecimal y separados por el símbolo ":". Por ejemplo: 2001:0DB8:0000:0000:0319:8A2E:0370:7348.

Una simplificación de esta notación se puede aplicar en el caso de que una dirección tenga muchos 0 consecutivos. El modo abreviado de representarla es utilizando el símbolo "::". Así, la manera compacta de representar la dirección anterior sería: 2001:0DB8::0319:8A2E:0370:7348.

Otra diferencia notable entre IPv4 e IPv6 es la jerarquización de las direcciones. En su momento, la asignación de direcciones IPv4 se llevó a cabo de un modo muy anárquico, ya que no se esperaba que el crecimiento de Internet fuera tan espectacular. Actualmente, cada corporación o cada operadora de telefonía tiene rangos de direcciones muy dispersos y mal dimensionados, lo que hace extremadamente difícil la gestión de las direcciones disponibles, la asignación de las nuevas y el encaminamiento global. Por esa razón, IPv6 ha jerarquizado de un modo más inteligente el reparto de sus direcciones, y cada país, operador o ISP dispone de un rango concreto y de un número de direcciones proporcional a su posible utilización de la red. Independientemente de la mejora de esta jerarquía en cuanto a localización geográfica, el hecho de separar de esta manera las direcciones permite asignar otras con mayor sencillez que hasta ahora.

De un modo similar a IPv4, se puede identificar qué tipo de dirección es sólo con el prefijo de la dirección IPv6, como indica la siguiente tabla:

Asignación de direcciones	
Prefijo	**Espacio de asignación**
0000::/8	Reservado. Las direcciones de *loopback* y las direcciones con integración de IPv4 salen de este prefijo.

Asignación de direcciones	
Prefijo	**Espacio de asignación**
0100::/8	Reservado.
0200::/7	Reservado.
0400::/6	Reservado.
0800::/5	Reservado.
1000::/4	Reservado.
2000::/3	Dirección unidifusión global. De aquí sale el rango de direcciones que se repartirán a los usuarios. Existen 2^{125} direcciones disponibles.
4000::/3	Reservado.
6000::/3	Reservado.
8000::/3	Reservado.
A000::/3	Reservado.
C000::/3	Reservado.
E000::/4	Reservado.
F000::/5	Reservado.
F800::/6	Reservado.
FC00::/7	Dirección unidifusión local única.
FE00::/9	Reservado.
FE80::/10	Dirección de enlace local unidifusión.
FEC0::/10	Reservado.
FF00::/8	Direcciones multidifusión.

Un hecho muy interesante que se consideró para elaborar esta asignación de direcciones es que da la posibilidad de representar direcciones de diferentes tecnologías incrustadas dentro de la nueva versión del protocolo. De esta manera se pueden representar direcciones IPv4, e incluso direcciones hardware del enlace de datos (como Ethernet).

La gran ventaja de insertar otros tipos de direcciones directamente en la IPv6 es que tienen un prefijo asignado. Así, por ejemplo, para tener una dirección Ethernet de un equipo dentro de una IPv6, el prefijo LAN es FE80::/10. Por lo tanto, si la dirección de la tarjeta Ethernet es: 00:90:F5:0C:0F:ED, entonces la dirección IPv6 queda: FE80::0090:F50C:0FED. Como se puede observar, el proceso también se puede realizar a la inversa: cuando llega un paquete con el

(23)En inglés, *link-local*.

prefijo de red FE80, se puede suponer que se trata de una dirección local[23] y su dirección hardware se puede extraer fácilmente. Además, con este mecanismo cualquier interfaz de red puede configurarse automática y autónomamente.

También hay que remarcar que IPv6, aparte de tener direcciones unidifusión, difusión y multidifusión como IPv4, añade soporte para un cuarto tipo, que son las direcciones *anycast*.

Las direcciones *anycast* son una gran innovación de IPv6, sobre todo porque aprovechan las direcciones unidifusión ya existentes. Así, una dirección unidifusión se convierte en *anycast* desde el momento en el que una misma IPv6 se asigna a más de una interfaz (incluyendo equipos diferentes). La idea que hay detrás de esta implementación es que responda las peticiones a un servicio concreto de la estación más próxima. Imaginemos, por ejemplo, dos servidores web con la misma IPv6, como: 2001:0DB8::0319:8A2E:0370:7348. Cuando la red reciba un paquete dirigido a esta IPv6, lo enviará a las dos estaciones, y la primera que responda será la que esté más cerca del equipo que realiza la petición. Actualmente, por las complejidades de la implementación de este tipo de direcciones sólo se utilizan para encaminadores. En este sentido, una subred puede tener más de un encaminador para salir a Internet mediante el mismo prefijo, y cada equipo utiliza el que está más próximo a la estación, con lo que de manera sencilla consigue un sistema de balanceo de carga.

Otra innovación relevante que incorpora IPv6 es la utilización mucho más intensiva del tráfico multidifusión dentro de las redes locales. De hecho, los equipos, por defecto, escuchan direcciones multidifusión con el prefijo FF02::1:FF00:0000/104, como veremos en secciones posteriores, para evitar la generación de tráfico difusión que afecta a todos los equipos de la subred y que no siempre es deseable.

Otras mejoras menores que introduce este protocolo son:

- **Mecanismo de opciones ampliado**: las opciones forman parte de una cabecera colocada entre la cabecera IP propiamente dicha y la cabecera de la capa de transporte. Este modo de poner las opciones permite una gestión más simple de las cabeceras por los dispositivos que deben tratar el paquete hasta llegar a su destino, lo que a su vez ofrece un sistema más simple y flexible.

- **Direcciones de autoconfiguración**: la asignación dinámica de direcciones ha sido sustancialmente mejorada con respecto a su predecesor. Uno de los motivos principales de esto es el hecho de que se pueda añadir la dirección hardware a la dirección IPv6. Así, basta un prefijo dado por el dispositivo de encaminamiento más próximo y la dirección hardware para

garantizar una dirección única a nivel mundial, siempre que se utilice el prefijo asignado por el operador a la hora de generar la dirección.

- **Facilidad para la asignación de recursos**: lo que con IPv4 era el *type of service* ahora se denomina *traffic class*. También se tiene la posibilidad de marcar flujos individuales, lo que permite mucha más flexibilidad a la hora de marcar tráfico prioritario.

- **Capacidades de seguridad**: como actualmente la seguridad es un tema muy importante, IPv6 incluye características de autenticación y privacidad. Por defecto, IPv6 incluye funcionalidades nativas para la creación de redes privadas virtuales (VPN) mediante IPsec, un protocolo de cifrado de los datos en tiempo real que con IPv4 era opcional.

Ejercicios

15. Tenemos un PC con una tarjeta Ethernet con MAC 34:27:A4:6F:AE:53. El operador le proporciona el prefijo 2001:0A54:0039::/48. Indicad la dirección local y la dirección de autoconfiguración de este equipo.

Solución ejercicio 15

La dirección local vendrá dada por el prefijo *link-local*. La dirección será: FE80::3427:A46F:AE53. La dirección de autoconfiguración sale del prefijo y de la MAC. Por lo tanto: 2001:0A54:0039::3427:A46F:AE53.

3.2.3. Problemas de la migración a IPv6

Uno de los motivos principales por los que todavía se trabaja con IPv4 es la dificultad que supone la migración al nuevo protocolo. La incompatibilidad de las direcciones y de las cabeceras de ambos protocolos provoca que la actualización a la nueva versión no sea fácil. También debemos tener en cuenta que las aplicaciones existentes sólo soportan el sistema de direcciones de IPv4; para aceptar las nuevas direcciones se ha de cambiar el código de la aplicación y todas las llamadas al sistema de acceso a la red.

Además del nivel de aplicación, existe otro problema muy grave. Dada la gran diversidad de redes que configuran Internet, se encuentran funcionando equipos de comunicaciones muy variados y no todos ellos tienen soporte para el nuevo protocolo. Por lo tanto, se debe actualizar el sistema operativo de los encaminadores de la red, con el consecuente gasto económico y de tiempo que ello supone, circunstancia que muchas empresas no están dispuestas a asumir (especialmente las grandes corporaciones americanas, que son las que tienen direcciones suficientes).

Por último, a causa de la gran utilización que tiene actualmente Internet, es un gran problema la obligación de tener que paralizar todas las redes para realizar la migración. En ese sentido, el problema es el enorme gasto económico de las

empresas que controlan todas sus transacciones a través de la red, lo que fuerza a efectuar la migración de manera progresiva y transparente a los usuarios, sin dejar de ofrecer los servicios disponibles en ningún momento.

3.2.4. Mecanismos para asistir la transición

Una migración entre dos protocolos es extremadamente compleja cuando uno de ellos se está utilizando masivamente. Como hemos visto, normalmente las razones son económicas, ya que la gran mayoría de las aplicaciones en la actualidad sólo soportan IPv4 y llevarlas a la nueva versión no siempre es sencillo (por ejemplo, aplicaciones bancarias). Por otra parte, todo el equipamiento hardware que forma la columna vertebral[24] de la red, si bien está preparado para soportar IPv6, no siempre tiene la configuración correcta ni la asignación de direcciones elaborada. Con todo, se espera que haya una fase de coexistencia de los dos protocolos. En cualquier caso, a pesar de la coexistencia existen escenarios que obligan a diseñar un plan de migración controlado.

(24)En inglés, *backbone*.

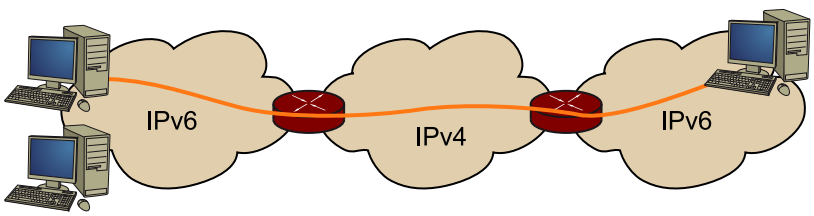
Así, los diferentes mecanismos de transición se pueden dividir en dos grandes grupos: mecanismos básicos y mecanismos para la interconexión de islas.

Se pueden distinguir dos mecanismos básicos, el conocido como *dual-stack*, en el que los equipos utilizan simultáneamente los dos protocolos y se conectan al que mejor se ajuste a las necesidades del momento, y el de *tunneling*, en el que dos equipos con *dual-stack* crean un túnel IPv4 entre ellos y por el que se comunican con IPv6.

Hay que señalar que un túnel es aquel mecanismo por el que se encapsulan dos protocolos de red dentro de un mismo datagrama. Por lo tanto, hay dos cabeceras de red consecutivas del nivel de red. IPv4 soporta el mecanismo de túnel mediante un valor especial en el campo protocolo que se encuentra en la cabecera.

Respecto a los mecanismos para conectar islas, éstos pretenden resolver el caso en el que varias máquinas interconectadas mediante IPv6 (isla IPv6) se quieren conectar con otra isla IPv6, pero por el camino existe una isla IPv4, tal como muestra la figura 9.

Figura 9. Redes IPv4 y redes IPv6



Estos mecanismos también están basados principalmente en túneles. Se pueden distinguir los siguientes tipos:

1) Túneles configurados: los extremos de los túneles entre las islas se configuran manualmente entre las dos redes. Los extremos deben ser *dual-stack*.

2) Túneles automáticos: se utilizan direcciones IPv4 mapeadas dentro de IPv6 y con un prefijo reservado, que es ::/96; por ejemplo, ::195.123.57.93, que el encaminador convierte en la dirección IPv4, mientras que en el otro extremo se vuelve a convertir a la IPv6. Los extremos no se dan cuenta del cambio y se pueden comunicar con IPv6 sin problemas.

3) Túnel *broker*: utiliza un gestor[25] que indica al cliente un guión[26] para ejecutar el túnel automáticamente. El cliente debe ser *dual-stack*, ya que la petición al gestor se realiza con IPv4, que contesta con un guión que permite conectar a un *tunnel-server* (que también es *dual-stack*) y éste, a su vez, conectarse a la red IPv6.

(25) En inglés, *broker*.

(26) En inglés, *script*.

4) 6to4: se asigna una dirección IPv4 compatible con el prefijo IPv6 a los encaminadores, que ejecutan un túnel. Por ejemplo, para la red 2001:d002:0507::/48, el encaminador tendría la dirección 208.2.5.7 (que sale de d002:0507). En el otro extremo se realizaría la operación análoga y se establecería el túnel.

3.3. Protocolos de soporte a IP

Tanto IPv4 como IPv6 son protocolos de red, pero ambos necesitan soporte de otros protocolos de la misma capa para poder llevar a cabo ciertas funciones que, de otra manera, sería difícil conseguir. A pesar de las diferencias estructurales dadas entre los dos protocolos, muchos de los servicios son diferentes y se engloban en esta sección para simplificar la comprensión, ya que si bien los protocolos divergen, su funcionalidad a menudo es muy similar.

3.3.1. ICMP

IP es un protocolo no orientado a conexión cuyo objetivo es enviar información, independientemente de la tecnología de niveles inferiores utilizada. Esto proporciona un entorno ideal para poder comprobar el estado de la red o enviar información de control en caso de que haya problemas en la red (por ejemplo, cuando un datagrama no puede llegar a su destino, o se produce congestión en un enlace, o ha expirado el TTL en un paquete). Todas estas situaciones requieren algún protocolo que, trabajando en la misma capa que IP, permita avisar automáticamente de cualquiera de estos eventos y para ello se diseñó el protocolo ICMP.

El protocolo ICMP[27] es el encargado de enviar mensajes de control (y de error) entre los diferentes equipos que forman la red.

(27)ICMP es la sigla de *Internet control message protocol.*

La siguiente tabla muestra todos los tipos de mensajes existentes con ICMP.

Los mensajes ICMP tienen varias utilidades: desde comunicar errores hasta depurar el estado de la red. Una de las herramientas más utilizadas para comprobar si un equipo está conectado a la red es *ping*. Otra funcionalidad es posible gracias al campo TTL de la cabecera IP, que ayuda a realizar otra tarea de depuración mediante una herramienta denominada ***traceroute***, que nos permite descubrir los encaminadores intermedios entre el origen y el destino de los datagramas. Para saber qué encaminadores cruza un datagrama, el *traceroute* envía paquetes IP consecutivos con un TTL de 1, otro de 2, otro de 3 y así sucesivamente hasta llegar al destino. Su efecto es que el primer encaminador, cuando recibe un paquete con TTL = 1, lo reduce, y al ser 0, lo descarta y envía de vuelta un paquete ICMP de TTL *expired*. En este punto, la aplicación sólo necesita saber quién ha enviado este paquete (IP origen) para conocer el encaminador. Evidentemente, con TTL = 2 sucederá lo mismo con el segundo encaminador, después con el tercero y de ese modo hasta el destino.

Descripción de los diferentes mensajes ICMP

Mensaje	Descripción
Destination unreachable	Indica que no se puede llegar al destino. Este mensaje tiene un campo de código que indica si es culpa de la red, del equipo en particular o del puerto. También distingue si no se puede llegar al destino o si el destino es desconocido.
Echo request/*Echo reply*	Estos dos mensajes son el de petición y el de respuesta. Cuando una estación activa recibe un *echo request*, debe responder con un *echo reply*. En general, es aconsejable que todos los equipos respondan a estas peticiones, aunque por seguridad muchas veces se filtran los mensajes. La aplicación por excelencia que utiliza los *echo request*/*reply* es el *ping*.
Source quench	Este paquete sirve para regular, indica que aquel enlace está sufriendo congestión. Actualmente, este tipo de paquete no se utiliza porque el control se realiza sobre todo en la capa de transporte.
Router advertisement	Este paquete de ICMP se envía a una dirección multidifusión en la que todos los encaminadores la escuchan por defecto. Con este mensaje es posible descubrir automáticamente la existencia de nuevos encaminadores en la red.
Router discovery	Es un paquete complementario al de *router advertisement*. Pero en este caso es un encaminador que acaba de entrar en una red el que pregunta por la existencia de otros encaminadores en la red.
TTL expired	Cuando el TTL de un paquete llega a 0, éste se descarta y el encaminador que lo ha descartado genera un paquete TTL *expired* en el origen del paquete descartado.
IP header bad	En el caso de que se detecte un error de la suma de comprobación en la cabecera de un paquete IP, éste se descarta y se avisa al origen con este paquete ICMP.

3.3.2. ARP

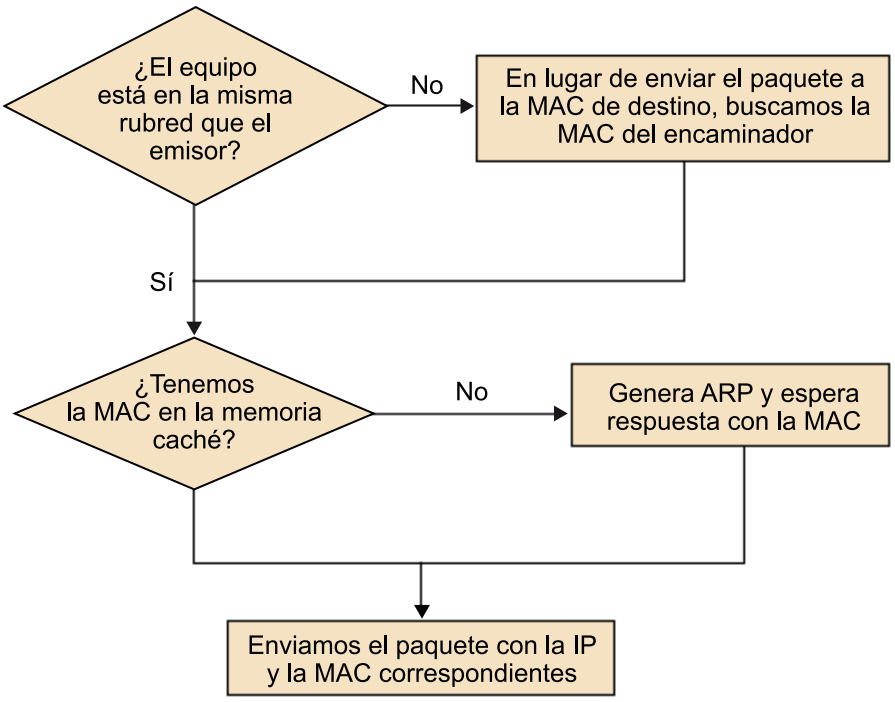
Al principio del módulo hemos visto que para enviar un datagrama IP a una estación de la misma red que el emisor, era necesario descubrir qué dirección del nivel del enlace de datos tiene esta estación, ya que las tecnologías de capas inferiores no entienden qué es una dirección IP.

Así, para que IP funcione, necesita interactuar con las capas inferiores y descubrir automáticamente cuál es la dirección de enlace de datos a la que responde un equipo para poder intercambiar información. Por ello el protocolo ARP[28] fue diseñado específicamente para IPv4.

[28]ARP es la sigla de *address resolution protocol*.

Para conseguir el descubrimiento de la dirección hardware de un equipo a partir de su IP, ARP emplea la funcionalidad que nos proporciona la capa de enlace de datos para enviar paquetes de difusión. Éste es el proceso para conseguir la dirección hardware (denominada MAC, como veremos en el módulo 4) y enviar el paquete, tal como se puede comprobar en el diagrama de la figura 10.

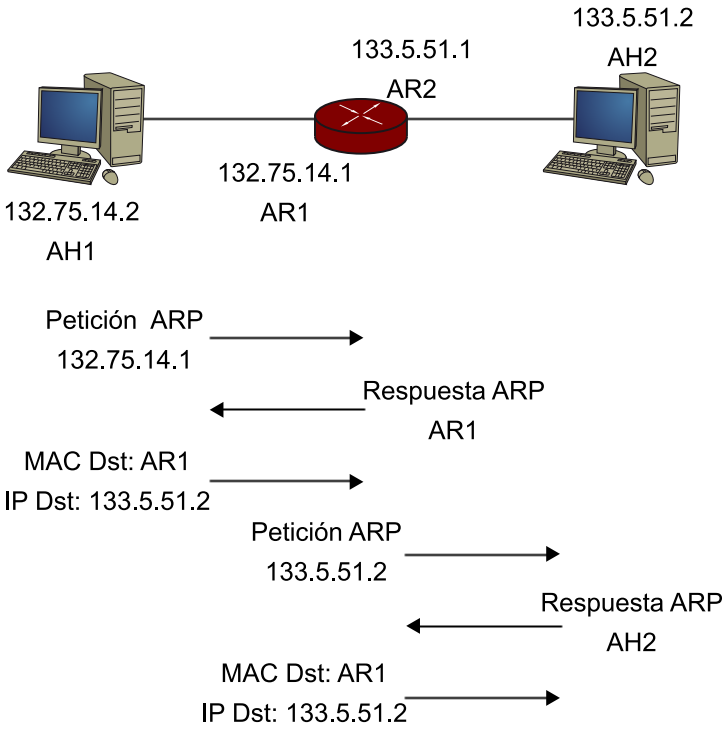
Figura 10. Diagrama de secuencia para el envío de un paquete IP



El proceso de ARP, ilustrado en la figura 11 con un ejemplo, sigue una serie de pasos para descubrir la dirección. Lo primero que se debe conseguir es la dirección MAC del próximo salto, para lo que se envía un paquete ARP a la dirección de difusión de nuestra red local. Este ARP buscará la IP del destino o la del encaminador, en función de si la estación forma parte de la subred o no. Una vez se obtiene la dirección MAC, se construye un paquete cuya dirección

MAC es la obtenida por ARP y cuya IP destino es la original (es decir, en el caso de que se envíe el paquete al encaminador esta IP será la del destino final, no la del encaminador).

Figura 11. Ejemplo de petición por ARP



Hemos visto que ARP nos sirve para descubrir qué dirección MAC corresponde a una IP, así como la importancia de esta operación. Para completar sus funciones, ARP tiene una variante denominada RARP que nos permite realizar la operación inversa, es decir, desde una dirección MAC averiguar a qué IP corresponde. RARP no es exactamente un protocolo de la capa de red, ya que incluye muchas funcionalidades de la capa de enlace de datos, pero dada la estrecha relación con ARP, se suele considerar conjuntamente. RARP ya no se utiliza, pues existen protocolos como BOOTP y DHCP que nos ofrecen ésta y otras funcionalidades, como veremos a continuación.

3.3.3. NDP

ARP es un protocolo que fue diseñado específicamente para IPv4. Los avances de las redes actuales han probado que es insuficiente para determinados servicios. Por ello, cuando apareció IPv6 se decidió que era necesario un protocolo más completo. Así surgió el NDP[29].

[29]NDP es la sigla de *network discovery protocol.*

NDP es un protocolo que permite descubrir a los vecinos existentes en una red local. El modo de operar es muy parecido al que de IPv4: se envían *neighbour solicitations* y se reciben *neighbour advertisements*. Sin embargo, la gran diferencia con ARP es que se utiliza tráfico multidifusión en lugar de difusión. Además, NDP forma parte de un protocolo mayor denominado ICMPv6, que es la extensión en IPv6 del protocolo ICMP.

Como hemos visto, cuando un nodo IPv6 se da de alta, escucha un conjunto de direcciones multidifusión, y una de ellas es *solicited-node*. Para automatizar este procedimiento, la dirección multidifusión *solicited-node* de una estación se construye de la siguiente manera: se eligen los últimos tres bytes de la dirección unidifusión y se añade al principio el prefijo multidifusión FF02::1:FF00:0000/104. Por ejemplo, la dirección multidifusión *solicited-node* para 2001:630:1310:FFE1:02C0:4EA5:2161:AB39 sería FF02::1:FF61:AB39. En ese momento, la estación comienza a escuchar al grupo multidifusión para responder con la dirección hardware del equipo al que va dirigida la petición.

Ved también

Podéis ver el IPv6 en el subapartado 3.2 de este módulo didáctico.

Esto nos da la versatilidad de que no todos los nodos reciben los anuncios, y en el caso de que no vayan dirigidos a ellos, ni los llegan a ver, con la reducción en el empleo de recursos que esto supone.

3.3.4. BOOTP

El protocolo BOOTP[30] se utiliza para obtener automáticamente una dirección IPv4 desde un servidor que está situado en la misma subred que el cliente. Normalmente, se utiliza en equipos finales durante el proceso de puesta en marcha, lo que permite que la imagen del núcleo para arrancar el sistema se obtenga a través de la red.

(30)BOOTP es la sigla de *bootstrap protocol*.

El funcionamiento del protocolo BOOTP es muy simple: el cliente envía un paquete de difusión del tipo *bootrequest*, indicando la dirección hardware y, si la sabe, su dirección IP. El servidor le contesta con un *bootreply* con los datos necesarios y un enlace a la imagen del núcleo preconfigurada por aquel equipo.

En la actualidad, este protocolo está en relativo desuso, ya que han aparecido alternativas más modernas, como el PXE[31], propuesto por Intel. El PXE da más versatilidad a la hora de configurar qué sistema operativo debe arrancar.

(31)PXE es la sigla de *preboot execution environment*.

3.3.5. DHCP

El siguiente protocolo de red que veremos (DHCP[32]) no es realmente un protocolo de red, sino un protocolo de aplicación. En cualquier caso, dado que se utiliza para configurar la red, se explica en esta sección. El protocolo DHCP es el que utilizan los dispositivos para obtener información de la configuración de los parámetros de red para un equipo IPv4 automáticamente.

(32)DHCP es la sigla de *dynamic host configuration protocol*.

El administrador de la red configura un prefijo de red, junto con un subrango de direcciones destinadas a la autoconfiguración (este rango de direcciones se denomina *pool*). Cuando se recibe una petición, el protocolo comprueba si el cliente está autorizado, y si lo está, o bien se le asigna una IP preconfigurada, que se obtiene de una base de datos a partir de la dirección hardware del equipo, o bien se le asigna una al azar del *pool*, si la dirección hardware no se encuentra. Esta cesión de IP va controlada por un temporizador. Cuando este temporizador expira y no se ha recibido ninguna noticia del cliente, se devuelve la IP al *pool* de direcciones libres. Para evitar esto, el protocolo implementa un sistema de *keep-alive* que va enviando renovaciones de uso de la IP al servidor para evitar que ésta caduque. Para realizar todas estas tareas, DHCP utiliza UDP para enviar la información.

Detallemos un poco más. Cuando un cliente da de alta una interfaz, envía un DHCP *discovery*, que es un paquete difusión para descubrir servidores DHCP. El servidor, cuando ve el paquete, comprueba la validez del cliente (base de datos de MAC) y envía un DHCP *offer* con su IP. El cliente responde directamente con un DHCP *request* y el servidor acepta mediante un DHCP *acknowledgement*, que contiene la duración de la IP y la configuración específica que el cliente ha pedido al DHCP *request*. Por ejemplo: encaminador por defecto, servidor de DNS, etc.

3.3.6. DNS

Una funcionalidad muy importante, y muy utilizada actualmente, es el DNS[33]. Este servicio se creó para simplificar la identificación de los diferentes equipos de la red. Hasta ahora hemos visto que un nodo de la red se identifica mediante su dirección IP, pero, como se puede observar, un usuario puede tener dificultades a la hora de recordar direcciones IP y asociarlas al servicio que proporciona.

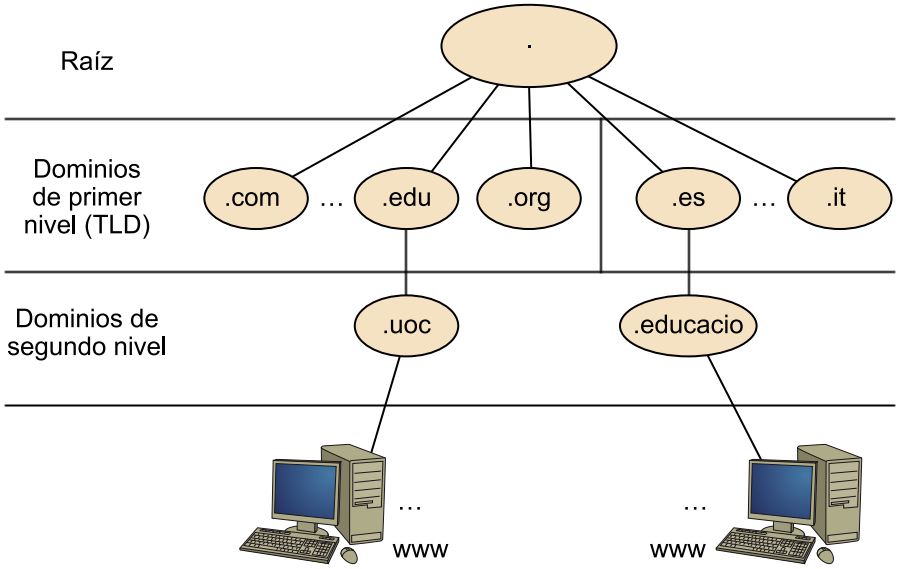
(33)DNS es la sigla de *domain name service*. En castellano, servicio de nombres de dominio.

El DNS nos permite realizar un mapeo de una dirección IP a un nombre fácil de recordar para una persona. Así, cada equipo o encaminador que tenga una IP puede tener un nombre asignado a ésta; el servicio de DNS nos permitirá resolver la IP que pertenezca a un nombre y viceversa.

DNS es una base de datos distribuida que utiliza TCP o UDP en el puerto 53 para ofrecer el servicio. Para simplificar la gestión dentro de una red tan grande como Internet, utiliza una aproximación jerárquica que permite distribuir la carga entre varios nodos. Concretamente, el DNS dispone de lo que se conoce como nodo raíz, desde donde proceden todos los nombres asignados. Este nodo raíz está compuesto actualmente por 13 servidores y en general se identifica con un ".", del que cuelgan los dominios de alto nivel (TLD[34]). Existen dos tipos de TLD, los genéricos (.com, .edu, etc.) y los geográficos (.es, .uk, .it, etc.), tal como se muestra en la figura 12. A su vez, los TLD delegan la resolución de nombres a los dominios de segundo nivel, de los que salen los de tercer nivel, y así sucesivamente hasta llegar a los equipos finales. En la figura 12 podemos ver dos ejemplos: www.uoc.edu y www.educacion.es.

[34]TLD es la sigla de *top level domains*.

Figura 12. Jerarquía del DNS



Los ejemplos anteriores nos permiten ver que el TLD .edu (perteneciente a las universidades a nivel mundial) tiene registrada una Universidad (.uoc), de la que cuelga una máquina (www), que por convenio normalmente hace referencia a un servidor web. La importancia de esta solución es que cuando se produce una consulta al DNS, ésta se realiza de manera recursiva. Por ejemplo, en el caso de www.uoc.edu, uno de los servidores raíz pasará la pregunta al dominio .edu (que está formado por varios servidores) y éste enviará la pregunta recursivamente al dominio .uoc. Posteriormente la consultará en su base de datos y devolverá la dirección IP correspondiente a la consulta. Cuando el cliente reciba la respuesta podrá establecer una conexión con esta dirección.

Un tipo de TLD que no se ha comentado hasta ahora es el .arpa. Este dominio se utiliza para llevar a cabo la denominada resolución inversa, es decir, obtener un nombre a partir de una dirección IP. Para ello se realiza una consulta al DNS invirtiendo la IP de la consulta y poniéndole el nombre de dominio

in-addr.arpa. Por lo tanto, para solicitar el nombre asociado a 193.45.15.48, se construirá una consulta de la forma 48.15.45.193.in-addr.arpa, que se resolverá de manera similar a la resolución directa de nombres vista hasta ahora.

4. Algoritmos y mecanismos de encaminamiento

Un algoritmo de encaminamiento es aquel proceso que permite el envío de un datagrama generado desde un nodo origen de la red a cualquier otro nodo de ésta. Generalmente, estos nodos estarán conectados a encaminadores diferentes, por lo que el datagrama deberá ir pasando por varios nodos de la red hasta llegar a su destino.

Actualmente, los algoritmos de encaminamiento se pueden clasificar en dos tipos fundamentales, los conocidos como estado del enlace (LS[35]) y los de vector distancia (DV[36]). Esta sección estará destinada a detallar los diferentes algoritmos de encaminamiento genéricos, que –como veremos posteriormente– se utilizan para protocolos de encaminamiento específicos en Internet.

[35] LS es la sigla de *link state*.

[36] DV es la sigla de *distance vector*.

La topología de Internet suele modelarse como un grafo, por lo tanto, a la hora de diseñar los algoritmos de encaminamiento se utiliza la teoría de grafos. Antes de describir cómo funcionan los algoritmos, recordaremos un poco qué es un grafo y qué herramientas nos proporciona para poder recurrirlo.

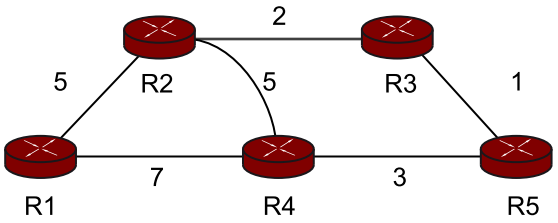
Un grafo $G = (N, A)$ es un conjunto de nodos N y una serie A de aristas que unen los nodos formando el grafo. Cada arista une un par de nodos de N. Debemos señalar que dentro de una red los nodos serán los encaminadores y las aristas los enlaces; desde el punto de vista del encaminamiento, los equipos finales no se consideran. En nuestro caso, asumiremos que todos los enlaces son bidireccionales, por lo que el grafo que resulte de la red siempre será no dirigido. Dentro de una red, al igual que dentro de un grafo, se conoce el nodo ***x*** como *vecino* del nodo ***y*** si hay una arista ***a*** que los una. Mientras que cada arista (enlace) tendrá un coste asociado para recurrirlo, el coste o la métrica de un enlace es una función que define "cuánto cuesta" enviar un datagrama a través de aquel enlace, por lo que generalmente nos interesará minimizar este coste. Se pueden definir muchas funciones de coste; por ejemplo, el precio que se debe pagar para enviar información mediante aquel enlace, o el retraso con el que la información llega al destino. Otra métrica posible es el ancho de banda disponible (que en este caso nos interesa maximizarla).

Históricamente, el coste se medía en función del retraso y la carga de los enlaces, ya que las diferencias de velocidades podían ser muy grandes, por ejemplo, de 56 Kbps de un puerto serie a 1,5 Mbps de una línea T1. En la actualidad, el valor de coste más utilizado en general es el número de saltos, ya que

se considera que cuantos más saltos se produzcan para llegar al destino más lento será el envío. En este sentido, cabe señalar que en el núcleo de la red todos los enlaces tienen anchos de banda y cargas comparables.

En cualquier caso, por ahora asumiremos que la función de coste de nuestra red es el coste de cada uno de los enlaces que lo forman. En la figura 13 se muestra un ejemplo de red con la lista de costes asociada.

Figura 13. Ejemplo de red con costes



Así, en una red cualquier algoritmo de encaminamiento tendrá por objetivo obtener la ruta con menos coste para llegar al destino. Por ejemplo, en el caso de la figura 13, la ruta de menos coste[37] para llegar de R1 a R5 es (R1, R2, R3, R5), a pesar de ser una ruta que necesita un salto más que la ruta más corta[38]: (R1, R4, R5).

(37)En inglés, *least cost path*.

(38)En inglés, *shortest path*.

Según el ejemplo anterior, la decisión de la ruta de menor coste ha asumido que todos los nodos disponen de información de toda la red[39], ya que para saber que el mejor camino para llegar a R5 pasa por R2 necesitamos información global de la red. Los algoritmos de estado del enlace que veremos un poco más adelante asumen este conocimiento global. En contraposición, se puede construir un algoritmo que esté descentralizado (es decir, que sólo dispongamos de información de los vecinos inmediatos[40]). Los algoritmos de vector-distancia forman parte de esta categoría.

(39)En inglés, *global routing*.

(40)En inglés, *decentralized routing*.

Para completar las diferentes categorías de algoritmos, otra clasificación posible es si el algoritmo es **estático** o **dinámico**. Los algoritmos estáticos son aquellos que, una vez configurados, cambian poco a lo largo del tiempo, y cuando lo hacen es, en general, porque alguien realiza el cambio de manera manual. En cambio, los algoritmos dinámicos (que en la práctica son los más utilizados en Internet) están pensados para ajustarse a los cambios topológicos de la red automáticamente.

4.1. Algoritmo de encaminamiento por la ruta más corta

Dada su sencillez, ésta es una de las técnicas más utilizada para desarrollar protocolos de encaminamiento. En general se combina con otros, como veremos más adelante.

Para poder entender bien este algoritmo, primero se debe tener claro qué significa "la ruta más corta". La ruta más corta puede cambiar en función de nuestra métrica. En este sentido, posibles métricas son:

- Mínimo número de saltos
- Menor distancia geográfica
- Mayor ancho de banda
- Menor carga del enlace

En la práctica, esto se traduce con un grafo y unos costes en sus aristas, como hemos visto en la figura 13. Ahora sólo debemos encontrar el camino menos costoso (más corto) para llegar al destino. Existen muchos algoritmos para conseguir los caminos cortos, pero el más utilizado es el algoritmo de Dijkstra, un algoritmo que fue presentado por Edsger Dijkstra en 1959.

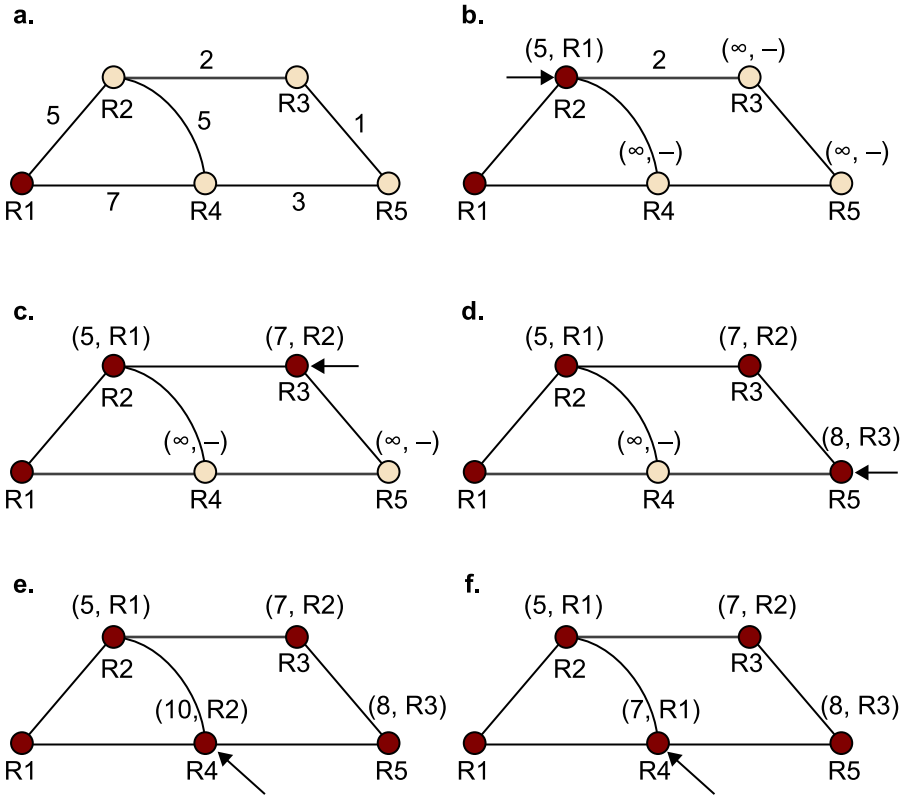
Lectura recomendada

Edsger W. Dijkstra (1959). "A note on two problems in connexion with graphs". *Numerische Mathematik* (núm. 1, pág. 269-271).

Básicamente, el algoritmo de Dijkstra es un algoritmo de búsqueda en grafos, que permite solucionar el problema de encontrar los caminos más cortos dentro de un grafo con costes positivos en los nodos. El algoritmo con k iteraciones es capaz de encontrar los caminos más cortos a k destinos con los menores costes.

En la figura 14 se muestra un ejemplo de ejecución del algoritmo de Dijkstra. La figura 14a muestra el grafo inicial con los costes asociados; los círculos llenos muestran los nodos tratados al menos una vez, y la flecha indica el nodo que estamos calculando en esta iteración. El objetivo es encontrar los caminos más cortos de R1 al resto de nodos de la red.

Figura 14. Ejemplo de ejecución del algoritmo de Dijkstra



Cada nodo tratado muestra entre paréntesis el coste total y el nodo previo para llegar a R1. El funcionamiento simplificado del algoritmo es:

1) Inicialmente todos los nodos tienen coste infinito y no tienen predecesor para llegar al nodo origen.

2) Entre todas las aristas del nodo inicial se elige la de menos coste (figura 14b). La arista nos lleva a un nuevo nodo *D*.

3) El resto de aristas se ponen en una lista *L*.

4) Se añaden a *L* todas las aristas que salen de *D*, excepto la que ya ha sido tratada.

5) Se elige la arista de coste mínimo. Esto nos lleva a un nuevo nodo *D*.

6) Si el coste es menor que el existente, se actualizan los valores con los nuevos y vamos al paso 3 del algoritmo mientras queden aristas en *L*.

7) En el caso de que sea mayor, se ignora y se va al punto 3 mientras queden aristas en *L*.

Al final, esto nos da la ruta más corta. Si repetimos esta operación en todos los nodos de la red, obtenemos unas rutas más cortas en éstos. Esta versión del algoritmo es estática, por lo que no contempla que pueda haber variaciones

en la red. Así, si hubiera algún cambio topológico, el sistema no se daría cuenta. Posteriormente, veremos alternativas dinámicas que permiten ajustarse a cambios topológicos.

Un punto importante que conviene tener en cuenta cuando se diseña un algoritmo de encaminamiento es el coste algorítmico, ya que es una operación que puede llegar a ejecutarse muchas veces y condicionar sensiblemente el tiempo de convergencia (tiempo que pasa desde que se empieza el cálculo de las rutas hasta que se obtienen todas). En el caso de Dijkstra, el coste es el siguiente:

- Supongamos que tenemos $n + 1$ nodos y queremos conocer el coste de calcular los caminos mínimos desde un nodo hacia los n restantes.

- Sabemos que en cada iteración se añade un nodo a la lista de nodos considerados, lo que nos lleva a realizar: $1 + 2 + 3 + ... + (n – 1) + n$ operaciones. Esto da un total de:

$$\frac{n(n+1)}{2} = O(n^2)$$

En cualquier caso, existen mecanismos para optimizar el algoritmo que nos permite reducir su coste a $O(n)$.

4.2. Inundación

Otro mecanismo para propagar la información dentro de una red, si bien no es exactamente un algoritmo de encaminamiento, es lo que se conoce por inundación.

> Este algoritmo se basa en el envío de la información a todas las interfaces del encaminador, exceptuando por la que llega. De este modo se consigue que todas las estaciones reciban la información que les queremos transmitir. En este caso no se propagan las rutas, sino que directamente se envía la información que hay que transmitir.

A simple vista se puede ver que esta solución tiene una serie de inconvenientes. El primero es la gran cantidad de mensajes duplicados que se generan y que llegan a las estaciones. El segundo es la repetición infinita de los mensajes, que se puede evitar de las siguientes maneras:

- Poniendo una fecha de caducidad a los paquetes. Cuando un paquete hace cierto tiempo que ha sido generado, se supone que todo el mundo lo ha recibido, que su validez ha expirado, y deja de ser difundido.

- También se puede limitar el número máximo de saltos que puede sufrir un paquete; de esta manera cada encaminador descuenta uno de un contador presente en la cabecera del datagrama, que cuando llega a 0 se descarta.

- La última manera propuesta, y la más utilizada, es poner un número de secuencia en los datagramas. Si una estación recibe un datagrama por el que ya ha hecho difusión previamente, este paquete no se reenvía por ninguna interfaz. Esta última alternativa debe considerar la posibilidad de que los números de secuencia se repitan; para evitar esto, se limita el número de mensajes que se pueden enviar por unidad de tiempo y para que no se produzcan repeticiones se destinan al contador suficientes bits. Por ejemplo, con 32 bits se pueden generar centenares de millones de números de secuencia sin sufrir colisiones.

Debido a su baja eficiencia, este sistema no se suele utilizar, a no ser que se halle en entornos en los que sea crítico que la información tenga que llegar al destino y en topologías muy cambiantes. Por ejemplo, en operaciones militares, en las que el riesgo de que los encaminadores sean destruidos no es negligible y siempre se necesita una ruta rápidamente alternativa.

Por su diseño, al igual que el algoritmo por la ruta más corta, el envío por inundación se considera un mecanismo estático, ya que no se ve afectado ni considera cambios topológicos.

4.3. Algoritmo de encaminamiento de estado del enlace

Como ya hemos apuntado, los algoritmos de estado del enlace basan su funcionamiento con encontrar el camino de coste mínimo entre dos nodos y en el hecho de que todos los nodos tienen conocimiento total de la topología y de los costes de los enlaces de toda la red. Así, el algoritmo estará dividido principalmente en dos partes: la primera consistirá en conocer la topología y la segunda en encontrar el mejor camino para llegar a los otros nodos.

Ejercicios

16. ¿Qué problemas creéis que puede suponer para este grupo de algoritmos la necesidad de conocer toda la topología en redes grandes?

Solución ejercicio 16

Conocer toda la topología implica guardarla en memoria, por lo que si la red está compuesta por muchos nodos (como Internet), se genera un grave problema de capacidad para conservar todo el grafo.

Éste es el primer algoritmo dinámico que vemos. Así, para considerar este dinamismo, los pasos que debemos seguir son más elaborados que antes. En este sentido, cuando cada encaminador ejecuta el algoritmo:

1) Averigua cuáles son los vecinos (ahora la lista no es estática).

2) Mide el coste del enlace que une el encaminador con cada uno de los vecinos.

3) Envía esta información a todos los encaminadores de la red.

4) Calcula la ruta más corta a los otros nodos a partir de la información recibida (similar a la que él ha enviado). Para ello, utiliza algún algoritmo de caminos más cortos, como Dijkstra.

Para poder adquirir conocimiento de toda la topología, cada nodo toma información de sus vecinos. Esto se puede hacer enviando un paquete por todas las interfaces de salida del encaminador y preguntando qué encaminadores hay en cada subred. Al recibir un paquete de este tipo, los encaminadores responderán con la dirección IP, que será almacenada en la base de datos del encaminador que ha enviado la petición.

Una vez obtenidos por todos los vecinos, se debe calcular el coste del enlace. Por esa razón se envían paquetes de prueba[41] a los vecinos. Cuando éstos responden (por ejemplo, se envía un *echo request* y se responde con un *echo reply*) se calcula el tiempo que ha transcurrido; de ese modo se obtiene una idea del retraso (es decir, el coste de los enlaces), y para mejorar la valoración se envían varios paquetes y se calcula la media de los resultados. Si se utilizan otras métricas, como el ancho de banda, el algoritmo puede tomar los datos directamente de la configuración del enlace, con lo que se ahorra los paquetes de prueba. Estos datagramas de prueba se envían periódicamente para ir refrescando el estado de los enlaces.

(41) En inglés, *probe packets*.

Ejercicios

17. Una red con 10 nodos y 32 enlaces en total tiene un tiempo de refresco t_r = 30 segundos. Se envían 3 paquetes de prueba por ronda, con una media de 64 bytes cada uno. Calculad cuál es el sobrecoste (*overhead*) que causa el protocolo LS debido a la valoración del retraso.

Solución ejercicio 17

Cada nodo enviará tres paquetes cada 30 segundos por todos sus enlaces. En la práctica, esto significa que cada enlace recibirá 3 × 2 = 6 paquetes de ida y las correspondientes 6 respuestas cada 30 segundos. Como existen 32 enlaces, se obtiene un total de (6 + 6) × 32 = 384 paquetes cada 30 segundos. Si tenemos en cuenta que se distribuyen uniformemente a lo largo de los 30 segundos, se alcanza la cantidad de 384 × 64 = 24.576 bytes. Que equivale a 196.608 bits cada 30 segundos, o a 196.608/30 = 6.553 ~ 6,5 kbps de sobrecoste a causa de los paquetes de prueba.

El siguiente paso es enviar esta información a los otros nodos de la red para que todos puedan llegar a saber la topología de toda la red; esta información se puede pasar de diferentes formas, por ejemplo, a través de inundación –como hemos visto anteriormente– o utilizando tráfico difusión.

Una vez se ha obtenido la topología, el último paso del algoritmo es encontrar el camino más corto desde cada encaminador al resto. Los algoritmos LS siguen el algoritmo de Dijkstra para encontrar los caminos más cortos. Una vez llegados a este punto, se puede decir que el algoritmo ha convergido. O, lo que es lo mismo, que ha llegado a un punto estable en el que todos los destinos son conocidos y accesibles.

Ved también

Podéis ver el algoritmo de Dijkstra en el subapartado 4.1 de este módulo didáctico.

Sin embargo, el trabajo del algoritmo no acaba aquí. En una red ideal, una vez el algoritmo ha convergido no sería necesario hacer nada más, pero en un caso real hay redes que dejan de existir, o bien fallos de hardware que provocan que un encaminador caiga, o bien algún problema en el cableado por el que un enlace deja de funcionar, etc. Por ello, los algoritmos LS deben considerar también el caso del cálculo de nuevas rutas. Por lo tanto, siempre que se produzca un problema en la red o aparezcan nuevos encaminadores, una vez detectado se debe informar a todos los nodos de la red de esta incidencia y ejecutar de nuevo el algoritmo en cada encaminador. El coste del algoritmo viene dado por el coste de Dijkstra, por lo tanto, es equivalente al de los caminos más cortos vistos antes.

4.4. Algoritmo de encaminamiento vector-distancia

El segundo algoritmo de encaminamiento dinámico que veremos en esta sección es el vector-distancia. Como ya hemos señalado, la mayor diferencia entre los algoritmos LS y DV es el hecho de que los segundos no tienen información topológica de toda la red, sólo de lo que aprenden mediante sus vecinos.

Ejercicios

18. ¿Qué ventaja creéis que puede aportar el hecho de no tener toda la información? ¿Y qué inconveniente?

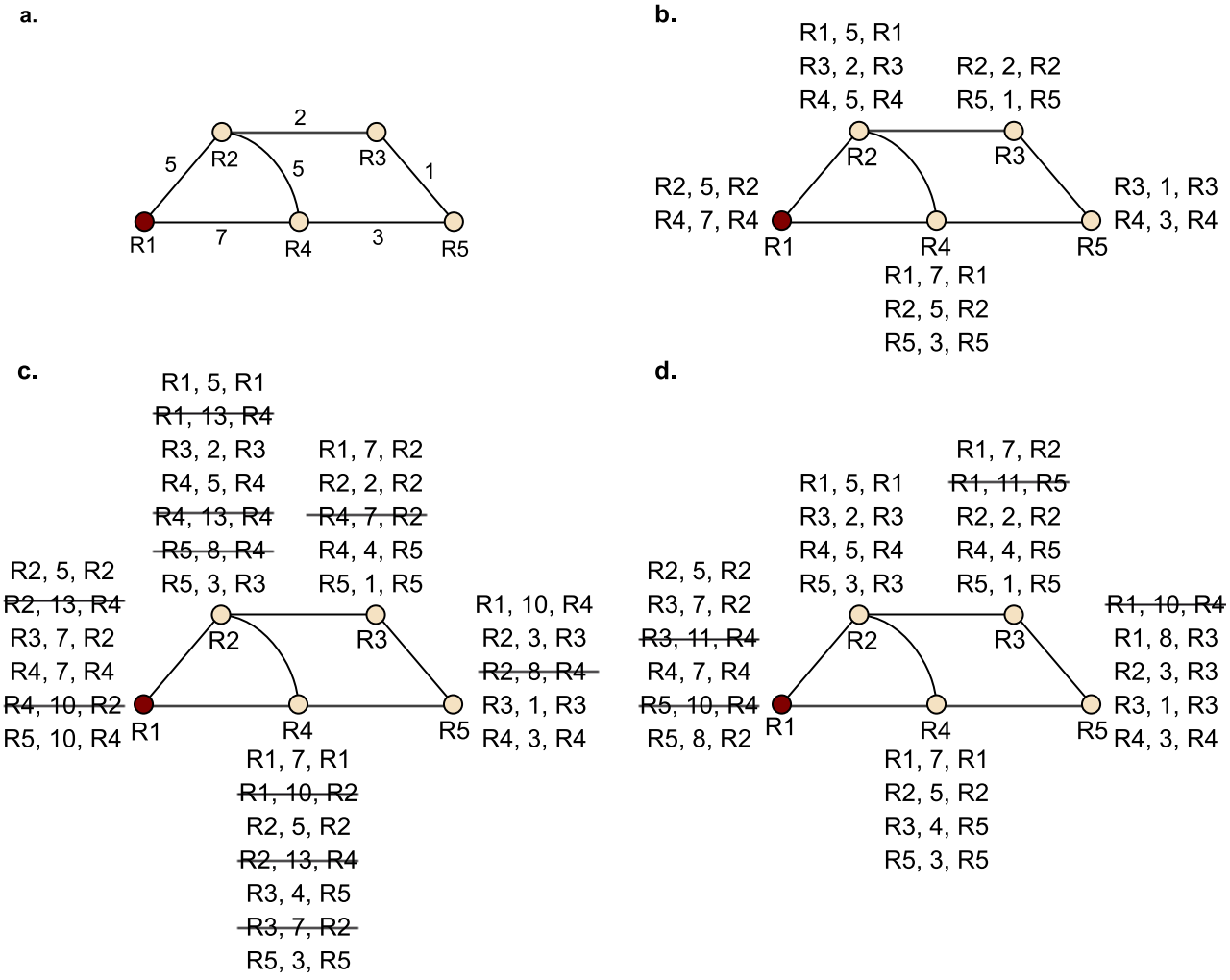
Solución ejercicio 18

La ventaja evidente de esta aproximación es el hecho de no tener que reservar en memoria toda la topología, con la consecuente reducción de recursos que supone en topologías grandes. El inconveniente es que no tener toda la topología no nos impide realizar optimizaciones, especialmente si existen varios caminos para llegar a los destinos.

Los algoritmos DV se basan en mantener una tabla, que se acaba tratando como un vector, en la que se informa de la mejor distancia conocida para cada destino. Esta información se va actualizando dinámicamente con los datos que se reciben de los múltiples vecinos de cada nodo.

Esta tabla, que es la tabla de encaminamiento, contiene la interfaz de salida, así como su destino y una métrica, que en este caso suele ser el número de saltos necesarios para llegar al destino especificado. Pero utilizar alguna otra información como retrasos se puede conseguir preguntando a los vecinos directos.

Figura 15. Ejemplo de ejecución del algoritmo DV



Para describir los pasos que realiza el algoritmo utilizaremos la misma red que para el caso de caminos mínimos, como muestra la figura 15a. Como los algoritmos DV se basan en no tener información global del sistema, todos han de trabajar juntos (de manera distribuida) para conseguir la convergencia final.

Los pasos que se deben seguir son:

1) Cada encaminador mira la métrica de los enlaces conectados a sus vecinos.

2) Propaga los datos de encaminamiento adquiridos a todos los vecinos de manera inteligente (no se envía información de un encaminador a sí mismo, y tampoco se envían las rutas conocidas que no son óptimas).

3) Cada encaminador recibe la información de los nodos vecinos y se queda con las rutas de menor coste (para un ejemplo completo, podéis ver la figura 15).

4) Se repite desde el paso 2 hasta que no quedan más rutas que transmitir.

Como se puede comprobar, este algoritmo es muy deseable, ya que nos proporciona las siguientes ventajas:

- Es autoconvergente: es decir, no es necesario especificar en qué momento debe pararse el envío de mensajes, sencillamente cuando conocen todas las rutas se para.

- Trabaja de manera asíncrona: si bien en el ejemplo de la figura 15 se ha discretizado el tiempo en pasos para simplificar la explicación, el algoritmo funciona independiente del orden en el que se reciben los mensajes y del momento específico en el que se envían.

- Es muy eficiente en recursos: no necesitamos conocer toda la topología, basta con tener la tabla de encaminamiento.

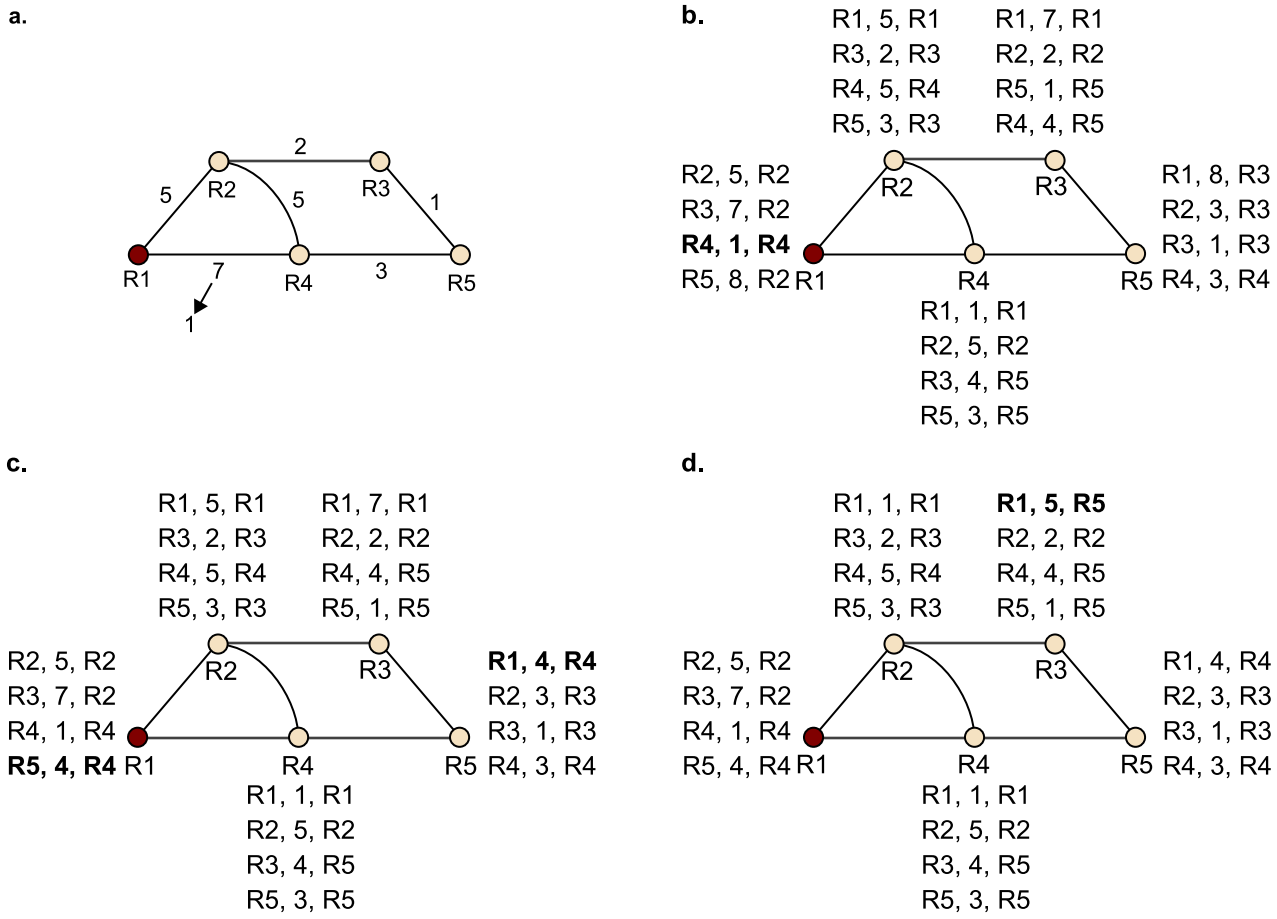
Una vez que conocemos el funcionamiento del algoritmo, explicaremos por qué funciona de manera óptima. Originalmente, este algoritmo fue diseñado por Bellman y Ford en 1957, por lo que se conoce como el algoritmo de Bellman-Ford. Según este algoritmo, el camino más corto entre dos puntos que involucran varios segmentos (saltos) está compuesto por subcaminos que a su vez son mínimos. Con otras palabras, si el camino óptimo para ir de R1 a R5 es R1-R4-R3-R5, el camino óptimo para ir de R4 a R5 será lógicamente R4-R3-R5. La ecuación recursiva que modela este comportamiento es: $d_x(y) = min_z\{c(x,z) + d_z(y)\}$, donde se quiere encontrar la menor distancia para ir de x a y, y donde $c(x,z)$ es el coste de ir desde el nodo $\boldsymbol{x}$ al próximo salto $\boldsymbol{z}$ para todos los enlaces de x hacia el resto de vecinos (z).

Como ya se ha comentado, los algoritmos DV están descentralizados y son dinámicos, lo que supone que cuando se produce algún cambio topológico se deben tomar medidas para mantener la coherencia en las tablas de encaminamiento de todos los encaminadores. Con los algoritmos LS esto era más o menos sencillo por el hecho de que todos los nodos esperan tener conocimiento de toda la topología. Sin embargo, en el caso de los algoritmos DV la cosa se complica un poco más, sobre todo cuando algún enlace pasa a tener un coste mayor.

Cuando un nodo detecta un cambio a un enlace (bien en el coste, bien porque ha dejado de funcionar, bien porque vuelve a funcionar), el algoritmo valora si eso le provoca algún cambio en su camino de menos coste hacia algún destino. En el caso de que sea así, informa del cambio a sus vecinos, que a su vez volverán a ejecutar el algoritmo de convergencia visto antes para ver si se han producido cambios y anunciarlos si es necesario.

Analicemos los casos en los que un enlace reduce e incrementa su coste. Para ello utilizaremos la misma red ejemplo que hasta ahora. Asumiremos que el algoritmo ya ha convergido (figura 15d) y que, por lo tanto, en t_0 el coste del enlace R1-R4 pasa de valer 7 a valer 1. La evolución del algoritmo se puede ver en la figura 16. Primero se dan cuenta del cambio R1 y R4, que a continuación pasan a propagar este nuevo estado. Como se puede ver en la figura 16b, las rutas desde R1 a R5 y a R3 no son correctas, pero como el enlace involucrado no es el que ha cambiado, el algoritmo todavía no puede saber que existe una mejor ruta. Para simplificarlo, en este caso no se muestran las rutas que se desestiman, sólo las nuevas en negrita en cada paso.

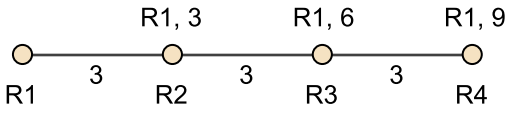
Figura 16. Reducción del coste de un enlace con algoritmos DV



Teniendo en cuenta los resultados, se podría pensar que cuando un enlace incrementa su coste, la convergencia también es rápida, pero vamos a ver que eso no es así. Para ilustrarlo necesitamos una red un poco más sencilla, como la de la figura 17, en la que en la parte superior se muestran los costes para llegar

a R1. Ahora supondremos que el enlace entre R1 y R2 se desconecta (tiene coste infinito). El proceso sería entonces: R2 avisa a R3 diciéndole: el coste de llegar a R1 ya no es 3, sino infinito. Y R3 respondería: yo tengo una ruta muy buena que dice que para llegar a R1 el coste es 6. Pensad que, en este caso, R2 no puede saber que la ruta que le llega de R3 pasa por R2. En este sentido, R2 actualizaría su tabla de encaminamiento a R1, 9, y R3 vería que existen dos rutas con el mismo coste (9) para llegar a R1, por lo que elegiría una al azar e incrementaría el coste a 12, una acción que se repetiría hasta el infinito. El problema radica en que la información que se pasa es estrictamente de coste y no del camino que se ha de seguir; por lo tanto, los encaminadores no pueden saber si ellos mismos forman parte del camino anunciado.

Figura 17. Ejemplo de cuenta al infinito



Con el fin de resolver este problema se propuso lo que se conoce como la cuenta hasta el infinito. Así, cuando un camino alcanza un coste de "infinito" se supone que el destino no es accesible, y este infinito tendrá un valor numérico que depende de la implementación. Debemos señalar que esta cuenta hasta el infinito sólo evita un bucle infinito, pero no resuelve el problema del tiempo de convergencia. Se han propuesto soluciones como *poisoned reverse*, pero no se acaba de resolver el problema general. En cambio, en secciones posteriores veremos cómo lo resuelven los algoritmos utilizados en Internet.

Para acabar esta parte haremos una pequeña comparativa de los dos algoritmos de encaminamiento en los que se basan los protocolos de encaminamiento utilizados en Internet. Así, las principales diferencias son:

- Como LS conoce toda la topología, no tiene el problema de la cuenta hasta el infinito; por lo tanto, siempre convergirá mediante el algoritmo de Dijkstra con coste $O(n^2)$.

- Los algoritmos LS siempre encontrarán una ruta igual o mejor que los algoritmos DV.

- Cuando se produce un cambio en la red, los algoritmos LS necesitan volver a ejecutar todo el algoritmo de convergencia, mientras que los algoritmos DV sólo propagan la ruta en el caso de que sea un nuevo camino de menor coste, con una complejidad algorítmica mucho más baja.

- El número de mensajes que se ha de enviar también es diferente. En el caso de LS se debe informar a todos los nodos de la red del cambio, mientras que los algoritmos DV sólo informan a los vecinos, que propagarán el cambio sólo si es necesario.

- Los algoritmos LS tienen un grave problema de escalabilidad cuando la red es muy grande, por lo tanto, no pueden ser utilizados en toda la red de Internet.

Como se puede comprobar no hay un claro ganador; cada alternativa tiene sus ventajas. Así, como veremos, lo que se acaba haciendo en Internet es utilizar dos alternativas.

Ved también

Podéis ver los protocolos de encaminamiento en Internet en el apartado 5 de este módulo didáctico.

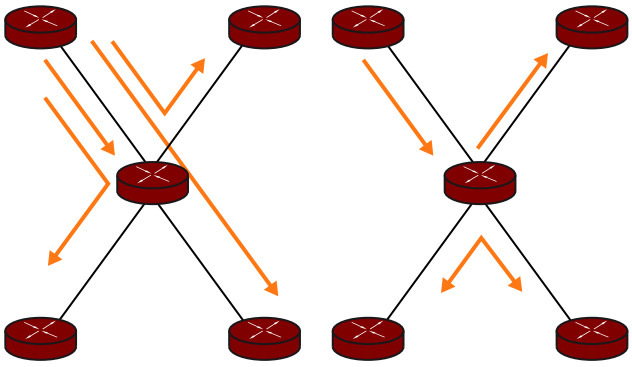
4.5. Encaminamiento basado en difusión

Como ya hemos visto antes, un modo de enviar información a una red es mediante inundación. Esta técnica, a pesar de ser muy poco eficiente, puede aportar algún beneficio por el hecho de ser robusta. El envío de datos por inundación no envía información de encaminamiento, sino directamente los datos.

El encaminamiento basado en difusión más que un algoritmo de encaminamiento es un mecanismo para transferir información de encaminamiento usando tráfico difusión. Como ya hemos visto, el tráfico difusión es aquel en el que se envía un datagrama que puede recibir e interpretar todo el mundo de la subred.

Las ventajas de esta solución respecto a la inundación se pueden comprobar en la figura 18.

Figura 18. Comparación de envío por unidifusión y por difusión



Como se puede observar, las dos alternativas generan un datagrama destinado a todos los encaminadores. Pero si nos fijamos, la diferencia es que con el caso unidifusión el encaminador central debe reencaminar paquetes por todas sus interfaces, mientras que con el caso difusión (a la derecha de la figura) el encaminador central se limita a enviar un paquete nuevo de difusión a todos sus vecinos con la información de encaminamiento. Esta manera de enviar información de encaminamiento es utilizada por los algoritmos LS para pasar la información entre los nodos. En cualquier caso, este mecanismo se ve afec-

tado por los mismos problemas que la inundación: ciclos en el grafo y sobrecoste por el exceso de mensajes en la red, que se soluciona con la utilización de números de secuencia, como hemos visto antes.

El modo más utilizado de evitar que un encaminador reciba la misma información más de una vez desde encaminadores diferentes es el que se conoce como árbol de expansión[42]. El árbol de expansión contiene todos los nodos del grafo, pero con la particularidad de que garantiza que no hay ciclos y, por lo tanto, que no se enviarán más paquetes que los estrictamente necesarios.

(42)En inglés, *spanning tree*.

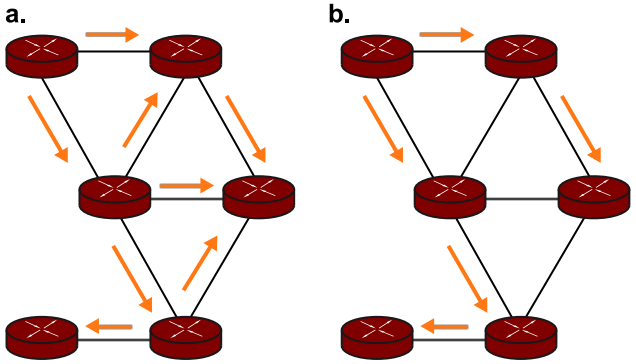
Lectura complementaria

En el siguiente artículo se pueden encontrar ejemplos de varias técnicas para construir árboles de expansión:

F. C. Gartner (2003). "A survey of self-stabilizing spanning tree construction algorithms".

La figura 19 muestra un ejemplo de árbol de expansión. La parte de la izquierda ofrece el intercambio con difusión, mientras que en la derecha vemos un árbol de expansión y el intercambio de mensajes en este caso. La parte más compleja de esta técnica es la creación y el mantenimiento (cuando existen cambios topológicos) del árbol.

Figura 19. Ejemplo de árbol de expansión



Lo que hace del árbol de expansión una opción interesante es que, independientemente del nodo al que envíe la información, nunca se enviarán más paquetes de los estrictamente necesarios, con el fin de que todo el mundo reciba la información exactamente una vez.

4.6. Encaminamiento basado con multidifusión

Al igual que el encaminamiento basado con difusión, el encaminamiento basado en multidifusión no se utiliza por sí mismo, sino que nos proporciona un mecanismo para poder encaminar a través de Internet el tráfico multidifusión. El principal objetivo de este mecanismo de encaminamiento es permitir seleccionar al receptor de la información de encaminamiento de una manera óptima, a diferencia de la solución basada con difusión, que no permite hacer distinciones y todo el mundo recibe toda la información. Evitar la difusión se debe a motivos de seguridad en la transferencia de los datos de encaminamiento. De este modo sólo reciben la información los encaminadores con suficientes privilegios.

En el encaminamiento basado con multidifusión cada encaminador de un grupo multidifusión crea un árbol de expansión que parte de él hacia el resto de los encaminadores multidifusión de la subred (cabe señalar que estos encaminadores han de tener soporte multidifusión; no puede ser cualquier encaminador). Este árbol incluye todos los encaminadores, tanto si están suscritos a algún grupo como si no lo están.

Cuando se recibe un datagrama dirigido a un grupo concreto, el encaminador realiza una "poda" del árbol, de manera que sólo le quedan aquellos nodos que forman parte del grupo. La poda de un árbol de expansión también es un *árbol de expansión*, pero de un conjunto menor de nodos. Una vez realizada la poda sólo falta enviar los datagramas a los vecinos, que a su vez se encargarán de procesarlos hacia los nodos suscritos y de enviarlos a sus vecinos del árbol multidifusión. El árbol se va ajustando dinámicamente, en función de las nuevas suscripciones al árbol, o de las bajas, en el caso de que un nodo deje de formar parte de aquél.

El problema principal de multidifusión, y el motivo por el que prácticamente no se utiliza en Internet, es que cada encaminador debe mantener en memoria los n árboles que representan los n grupos en los que están suscritos sus equipos, y eso, para redes con muchos nodos, supone un grave problema de memoria y un tiempo de proceso en el encaminador, además de que no todos los encaminadores de la red tienen el soporte multidifusión activado.

5. Protocolos de encaminamiento en Internet

Antes de empezar la descripción de los diferentes protocolos de encaminamiento presentes en Internet, es conveniente ver un poco más en detalle cómo se estructura la red.

Hasta ahora hemos visto que Internet se separa por redes, que a su vez están estructuradas en subredes, pero en realidad, para simplificar la gestión todavía existe un nivel más de abstracción, el de los *dominios*. Un dominio es el conjunto de todas las redes que forman parte de una misma unidad administrativa. Los dominios también se conocen como sistemas autónomos (AS[43]). Un sistema autónomo indica una única política de gestión de los recursos (direcciones IP, separación con subredes, etc.), de una manera interna y transparente al resto de Internet. Cada sistema autónomo se comunica a través de los encaminadores fronterizos[44] con otros sistemas autónomos, tal como muestra la figura 20.

[43] AS es la sigla de *autonomous systems*.

[44] En inglés, *border routers*.

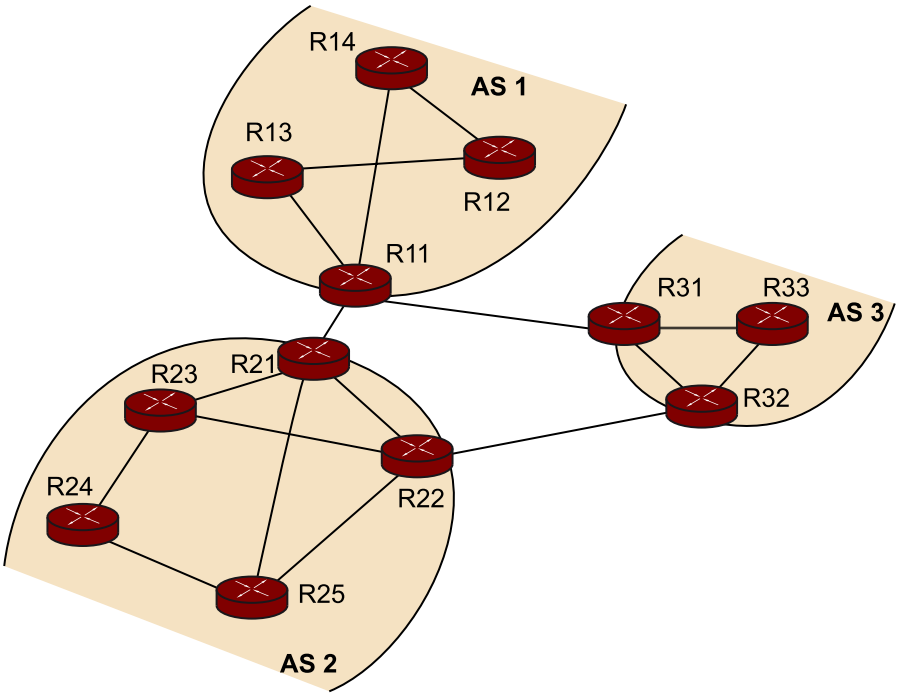
Algunos autores, como Tanenbaum o Kurose, denominan el encaminamiento de nivel de AS como **encaminamiento jerárquico**.

Lecturas recomendadas

Andrew S. Tanenbaum (2003). *Redes de computadores* (4.ª ed.). Pearson.

James F. Kurose; Keith W. Ross (2005). *Computer networking: a top-down approach featuring the Internet*. Addison-Wesley.

Figura 20. Ejemplo de jerarquía con AS



En la figura 20 se pueden observar tres AS, donde AS1 tiene un encaminador fronterizo, y AS2 y AS3 tienen dos. Con esta solución jerárquica, los encaminadores fronterizos se limitan a anunciar a los AS vecinos sólo las rutas que gestiona el AS. Generalmente, en Internet esto se lleva a cabo mediante un

[45] BGP es la sigla de *border gateway protocol*.

Ved también

Podéis ver el protocolo BGP en el subapartado 5.3 de este módulo didáctico.

protocolo denominado BGP[45]. Este tipo de encaminamiento se conoce como encaminamiento interdominio, dado que involucra conexiones entre dominios (AS). En el caso de que un AS tenga más de un encaminador fronterizo, se dirá que este AS tiene conectividad *multihoming* en Internet, es decir, que podrá ser accedido desde Internet mediante diferentes puntos.

Así como existe el encaminamiento interdominio, también contamos con el encaminamiento intradominio, que es el que gestiona internamente la conectividad entre los diferentes encaminadores de un mismo dominio. En este caso se pueden utilizar protocolos de encaminamiento como RIP[46], OSPF[47] o una variante de BGP conocida como IBGP[48].

(46)RIP es la sigla de *routing information protocol*.

(47)OSPF es la sigla de *open shortest path first*.

(48)IBGP es la sigla de *intra domain border gateway protocol*.

Para poder enviar los datagramas al próximo salto, un encaminador basa su funcionamiento en las denominadas tablas de encaminamiento. Una tabla de encaminamiento indica para cada prefijo destino cuál es el siguiente encaminador en el camino. Un posible ejemplo de tabla de encaminamiento puede ser:

Destination	Gateway	Genmask	Metric	Iface
183.128.13 . 0	0 . 0 .0 .0	255.255.255.0	2	eth0
147.83 .120. 0	183.128.13.2	255.255.255.0	2	eth0
147.83 . 0 . 0	131.10 .4 .2	255.255. 0 .0	2	eth2
0 . 0 . 0 . 0	150.18 .87.1	0 . 0 . 0 .0	2	eth1

Cabe señalar que el formato de la tabla puede cambiar en función del fabricante del encaminador, pero la información mínima contenida es siempre la misma. Si detallamos por columnas tenemos:

- ***Destination***: indica el prefijo destino de los datagramas.
- ***Gateway***: es la IP del próximo salto. Cuando es 0.0.0.0 significa que estamos en el último salto y se aplican las técnicas ARP vistas anteriormente para el envío de datagramas dentro de la misma subred.
- ***Genmask***: es la máscara de red que indica el conjunto de IP que corresponde al destino.
- ***Metric***: indica el coste de elegir esta ruta.
- ***Iface***: especifica la interfaz física por la que saldrán los datagramas.

En cualquier caso, como se puede ver en la tabla anterior, hay prefijos que se solapan, en concreto el 147.83.120.0/24 está incluido en el 147.83.0.0/16, que a su vez se encuentra en el 0.0.0.0/0.

Con el fin de deshacer esta ambigüedad, los encaminadores usan la técnica *longest prefix match* o, lo que es lo mismo, elegir la ruta por la que el destino tiene más bits en común (se ve reflejada más parte de la IP en la ruta). Por esta razón, en general las rutas se ordenan por máscaras, de la máscara mayor a la más pequeña, de modo que la primera ruta que corresponde al destino es la que se utiliza. Así, si nos llega un datagrama destinado a la dirección 147.83.121.3, el encaminador comprobará que la 183.128.13.0/24 no corresponde, tampoco la 147.83.120.0/24 ni la 147.83.0.0/16; pero sí lo será cuando el encaminador elegido para reenviar el paquete sea el 131.10.4.2, que está directamente conectado a la interfaz eth2 de nuestro encaminador.

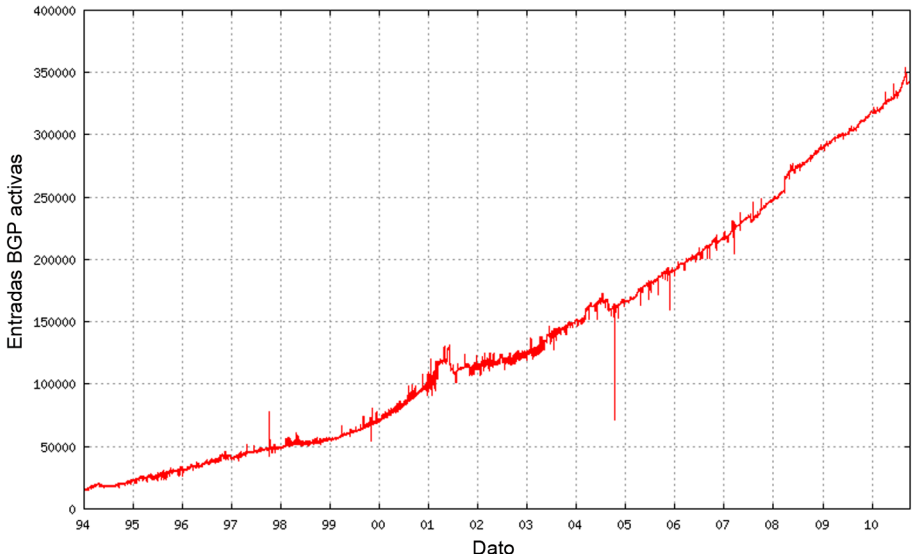
Si la IP destino hubiera sido 120.134.23.235, se hubiese elegido la última ruta. Cabe señalar que 0.0.0.0/0 indica la ruta por defecto[49], que es la ruta por la que se envían los datagramas que no tienen otra ruta más específica. Actualmente, uno de los grandes problemas de Internet es que los encaminadores del núcleo de la red[50] tienen lo que se conoce como DFZ[51], por lo que no tienen ninguna entrada a la tabla de encaminamiento que sea por defecto. Esto implica que conocen el camino hacia todas las subredes, por lo que sus tablas de encaminamiento están formadas por centenares de miles de entradas. La figura 21 muestra un ejemplo de la evolución desde 1994 hasta principios del 2010 en número de entradas del tamaño de la tabla de encaminamiento a la DFZ.

(49) En inglés, *router of last resort*.

(50) En inglés, *core routers*.

(51) DFZ es la sigla de *default free zone*.

Figura 21. Ejemplo del número de entradas a la tabla de encaminamiento de un encaminador del núcleo



Fuente: http://bgp.potaroo.net/as6447/

5.1. RIP

RIP es uno de los primeros protocolos diseñados por ArpaNET. Está basado en los mecanismos de encaminamiento del tipo DV y aún se aplica en entornos intradominio.

Existen una primera versión y una segunda compatible con aquélla.

El modo de funcionamiento del protocolo es prácticamente el mismo que el empleado por el algoritmo DV teórico. Las únicas particularidades que tiene RIP respecto a DV son que:

Ved también

Podéis ver el algoritmo de encaminamiento vector-distancia en el subapartado 4.4 de este módulo didáctico.

- El coste de los enlaces viene contado por el número de saltos, lo que significa que cada enlace tiene un coste de 1. Se cuentan todos los saltos y la subred destino. Después veremos un ejemplo de ello.

- Se sitúa el infinito en 15 saltos, lo que implica que la red controlada por RIP no puede tener un diámetro mayor de 15 saltos.

- Los encaminadores RIP se pasan información cada 30 segundos para refrescar la información de encaminamiento.

- Si no se reciben actualizaciones de los vecinos durante 180 segundos, se supone que el vecino se ha caído, se actualizan las rutas y se propaga la información al resto de los vecinos.

Por lo tanto, RIP genera dos tipos de mensajes: mensajes de anuncio[52] y mensajes de respuesta[53]. Por su parte, los mensajes de anuncio se utilizan con el fin de anunciar la presencia del encaminador y para avisar a los vecinos de que se ha recibido correctamente una respuesta.

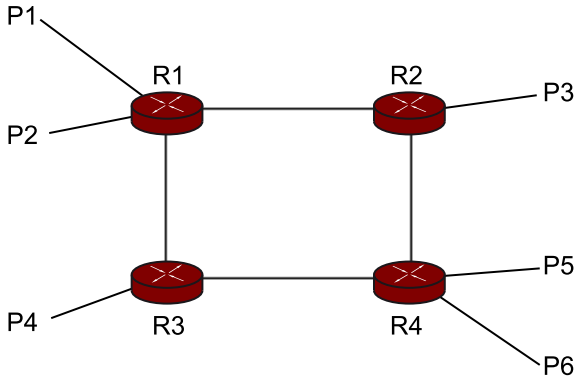
(52)En inglés, *RIP advertisements*.

(53)En inglés, *RIP response messages*.

Los mensajes de respuesta contienen las rutas conocidas por el encaminador (hasta 25 por paquete). En concreto, se envían los prefijos conocidos y la distancia a la que se encuentran. El receptor del mensaje de respuesta añadirá esta información a su tabla de encaminamiento mediante la técnica descrita por los algoritmos DV.

Para ver un ejemplo de funcionamiento de RIP, observemos la red de la figura 22. Queremos ver la evolución de los anuncios enviados a R1. En esta figura, los encaminadores se muestran con R# y los diferentes prefijos que anuncia cada red mediante P#. Debemos señalar que RIP, como todos los protocolos de encaminamiento, utiliza prefijos (IP) para los anuncios, pero no encaminadores concretos.

Figura 22. Ejemplo red RIP



En el instante 0, la tabla de encaminamiento de R1 será:

Subred destino	Siguiente encaminador	Número de saltos
P1	-	1
P2	-	1

Al cabo de 30 segundos se recibirán los anuncios de R2 y de R3, que configuran la red:

Subred destino	Siguiente encaminador	Número de saltos
P1	-	1
P2	-	1
P3	R2	2
P4	R3	2

Por último, al cabo de 30 segundos más (60 segundos desde el inicio), el algoritmo convergirá en una tabla:

Subred destino	Siguiente encaminador	Número de saltos
P1	-	1
P2	-	1
P3	R2	2
P4	R3	2
P5	R2	3
P6	R2	3

En este caso se elige R2, pero como el coste es el mismo, se hubiera podido elegir R3 como próximo salto para llegar a P5 y P6.

Supongamos ahora que el enlace que une R2 y R4 cae. Al cabo de 180 segundos, R2 comprobará que no se reciben actualizaciones de R4 y, por lo tanto, avisará de ello. A partir de ahí la tabla de R1 quedará de nuevo:

Subred destino	Siguiente encaminador	Número de saltos
P1	-	1
P2	-	1
P3	R2	2
P4	R3	2

Esto sucede porque las rutas que no se utilizan se descartan automáticamente, y cuando llegue la siguiente actualización de R3, el protocolo convergirá por R1 con:

Subred destino	Siguiente encaminador	Número de saltos
P1	-	1
P2	-	1
P3	R2	2
P4	R3	2
P5	R3	3
P6	R3	3

Lógicamente, esta información llegará a R2 al cabo de 30 segundos y se añadirán las entradas:

.	.	.
.	.	.
P5	R1	4
P6	R1	4

5.2. OSPF

Por encima de RIP, OSPF es uno de los protocolos intradominio más utilizados en la actualidad. Cuando se suele utilizar RIP en redes relativamente pequeñas, OSPF se utiliza internamente en AS de tamaños mayores. Por ello sorprende que OSPF sea un protocolo del tipo LS.

La versión 2 de OSPF tiene un funcionamiento bastante intuitivo: todos los nodos de la red envían información topológica mediante trafico difusión (aunque existen otras extensiones que permiten utilizar tráfico unidifusión o incluso multidifusión). Una vez que la topología es conocida por todo el mundo, se ejecuta el algoritmo de Dijkstra, tal como ya hemos visto. Por robustez, cada 30 minutos el algoritmo intercambia información topológica, aunque no haya cambiado nada en la red.

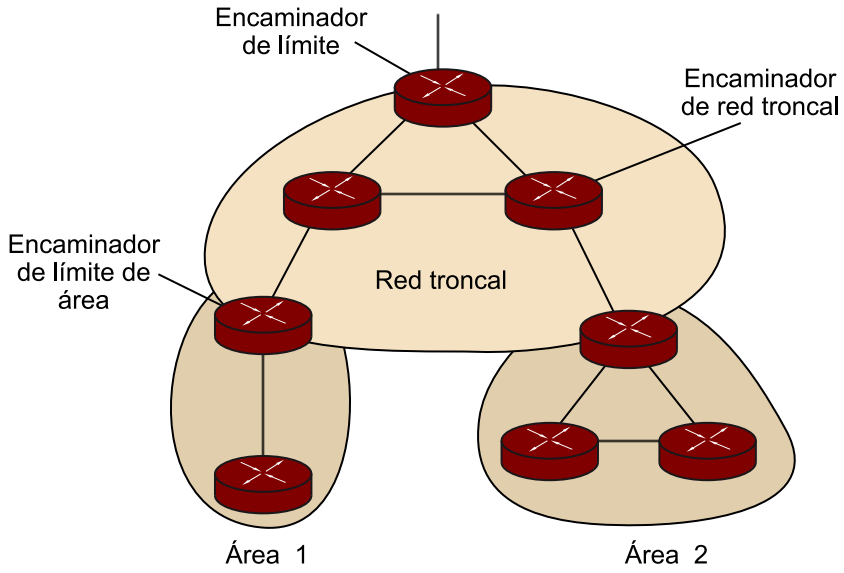
Ved también

Podéis ver el algoritmo de Dijkstra en el subapartado 4.1 de este módulo didáctico.

Como OSPF está pensado para redes bastante mayores que RIP, con el fin de proporcionar un protocolo escalable, permite la división de la red en áreas. Cada área es un conjunto de encaminadores que intercambian información, y entre los que cuenta OSPF. Las áreas tienen un encaminador de área fronterizo[54] que agrega toda la información del área al resto de la red. A su vez, cada encaminador de área fronterizo está conectado a la columna vertebral del AS, que se encargará, utilizando también OSPF (o algún otro protocolo), de interconectar las diferentes áreas. La figura 23 muestra un ejemplo.

(54)En inglés, *area border router*.

Figura 23. Ejemplo de división por áreas en OSPF



Un punto importante que debemos considerar es que OSPF deja al administrador de la red la tarea de especificar cuáles son los costes de los enlaces. De este modo se puede optar por que sean todos 1 para la política de menor número de saltos, así como priorizar los enlaces con más ancho de banda si se quiere.

Dado que OSPF fue pensado para sustituir a RIP, era obligatorio que tuviera mecanismos de seguridad, por ello OSPF permite autenticar las peticiones OSPF con el fin de evitar ataques malintencionados de anuncios de redes falsas. De este modo, los encaminadores OSPF de la red utilizan una clave secreta (decidida por el administrador) que permite cifrar el *hash* generado a partir de la petición para garantizar que nadie que no tenga la clave pueda generar peticiones falsas.

El último punto de innovación de OSPF sobre RIP fue la introducción de lo que hoy en día se conoce como *load-balancing*. Así, si hay dos rutas con el mismo coste para el mismo destino, el sistema permite que el tráfico oscile entre los enlaces de igual coste.

5.3. BGP

Los protocolos de encaminamiento vistos hasta ahora comparten un problema: el hecho de que en Internet haya demasiados nodos para poder encaminarlos a partir de la dirección IP. Esto provoca que los encaminadores del núcleo de la red deban tener billones de entradas en su tabla de encaminamiento. Para resolver esta situación se utilizan las direcciones por prefijos (CIDR), como ya hemos visto antes. Sin embargo, la información de encaminamiento continúa siendo demasiado elevada para enviarla directamente. Por esta razón aparece el BGP.

De todos los protocolos de encaminamiento presentes en Internet el más complejo, pero también el más utilizado, es BGP. Fundamentalmente, este protocolo se utiliza para intercambiar información de encaminamiento entre dominios diferentes (entre sistemas autónomos distintos).

Por lo tanto, BGP anunciará prefijos de red entre sistemas autónomos vecinos, lo que implica que la abstracción que se hace de la red sea a nivel del sistema (AS) y no a nivel de encaminador, como habíamos visto hasta ahora. También se reduce mucho la cantidad de información que se debe transmitir entre los diferentes dominios.

AS

El sistema autónomo (AS, por la sigla inglesa de *autonomous system*) es un conjunto de redes y encaminadores IP administrados por una sola entidad (o, a veces, más de una) que comparten una política de encaminamiento común.

Antes de realizar una descripción más exhaustiva, es importante tener claros algunos conceptos que convierten BGP en el estándar de facto en Internet. Los protocolos de encaminamiento vistos hasta ahora guardaban el identificador del próximo salto (normalmente la IP), lo que nos permitía enviar el paquete hacia el destino. Por el contrario, BGP guarda todo el camino, pero en lugar de guardar las direcciones IP conserva lo que se conoce por AS-Path, que es la lista de AS que se debe cruzar para llegar al destino. Por el hecho de tener toda la lista de dominios que hay que cruzar, BGP puede evitar con eficiencia bucles en los caminos, y como todas las conexiones interdominio suelen contar con anchos de banda muy grandes, se puede utilizar la política de saltos mínimos para llegar al destino. Debemos señalar que cuando se habla de saltos mínimos con BGP nos referimos a los saltos en el nivel de AS, y no de encaminador físico, lo que nos permite obviar los diferentes encaminadores internos que tenga el AS que recibe nuestros paquetes.

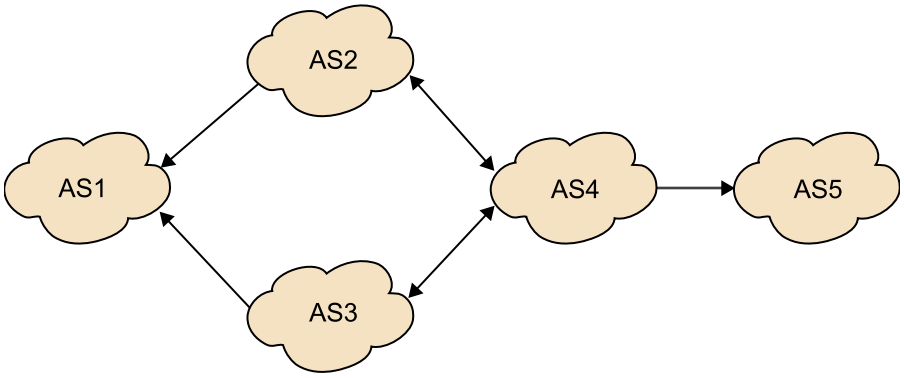
Si todo el trabajo que tuviera que hacer BGP fuera el expuesto hasta ahora, el protocolo sería bastante sencillo, pero el hecho de que BGP se utilice en conexiones interdominio (eBGP[55]) provoca que unidades de administración diferentes (compañías que pueden ser competencia) deban intercambiar información interna, como qué conexiones se tienen hacia el exterior. Esto ha provocado que BGP, además de intercambiar información de encaminamiento, haya de diseñarse, para evitar dar más información de la necesaria, con las denominadas *forwarding policies*. Por lo tanto, BGP anunciará las rutas a sus vecinos en función de las relaciones comerciales que éstos tengan. Así, se distinguen tres tipos de relaciones entre dominios: clientes, proveedores y pares[56].

(55)eBGP es la sigla de *external BGP*.

(56)En inglés, *peers*.

Un cliente es un AS que paga a un proveedor para que ofrezca tráfico de su información al resto de la red. Un proveedor es aquel que ofrece sus servicios (y conexiones) a sus clientes. Por último, un par hace referencia a aquella relación existente entre dos dominios, por los que sólo se da tráfico de un subconjunto de rutas.

Figura 24. Relaciones entre diferentes AS



Para entender mejor estos conceptos, la figura 24 muestra una red de ejemplo, en la que AS1 es cliente de AS2 y AS3. AS2 y AS4 son pares (doble flecha), al igual que AS3 y AS4. Por su parte, AS4 es proveedor de AS5 (o AS5 es cliente de AS4).

En vista de esta red se deben tener en cuenta las siguientes consideraciones: AS1 está pagando tanto a AS2 como a AS3 para tener un servicio, lo que significa que AS1 debe evitar enviar tráfico de AS2 hacia AS3 y viceversa, ya que de esta manera AS2 tiene tráfico gratuito (*free-ride*) hacia AS3. Por su parte, AS2 y AS4 tienen una relación de pares[57], es decir, que han llegado a un acuerdo económico para compartir los gastos del enlace que los une. Esto implica que AS4 querrá enviar tráfico entre AS2 y AS5 de manera gratuita, pero no entre AS2 y AS3, porque los dos son pares y no clientes (es otro caso de *free-ride*).

(57)En inglés, *peering*.

En resumen, las políticas de anuncio de rutas vienen dadas por las relaciones comerciales entre los vecinos, con las siguientes normas básicas:

- Un proveedor anunciará a sus clientes todas las rutas conocidas por él.

- Un proveedor nunca anunciará las rutas de un par (o de otro proveedor) a otro par (o proveedor), sólo las de sus clientes.

- Un cliente nunca anunciará a un proveedor y a un par las direcciones de sus proveedores, sólo las anunciará a sus clientes.

Sin embargo, en Internet nos encontramos tres niveles de jerarquía: aquellos AS que no tienen proveedores y que se conocen como Tier-1; aquellos que tienen clientes, proveedores y pares, denominados Tier-2 y Tier-3, según lo bien conectados que estén dentro de la jerarquía; y los Stub AS, que son los que no tienen clientes (la mayoría) y que no generan tráfico, es decir, que son origen o final de la comunicación.

Debemos señalar que las interconexiones lógicas entre AS no siempre corresponden a enlaces físicos, lo que significa que en muchas ocasiones habrá varios enlaces entre los dos AS, y otras veces encaminadores intermedios que harán tráfico a nivel IP entre los AS.

Una vez que un AS ha aprendido una ruta enviará la información a los otros encaminadores internos en el dominio para que sepan cómo llegar a los diferentes destinos. Esto se realiza mediante sesiones BGP a nivel intradominio (iBGP), mientras que al mismo tiempo los encaminadores fronterizos, en función de las políticas de encaminamiento presentes, avisarán a los AS vecinos de los nuevos destinos aprendidos.

Lecturas recomendadas

Andrew S. Tanenbaum (2003). *Redes de computadores* (4.ª ed.). Pearson.

James F. Kurose; Keith W. Ross (2005). *Computer networking: a top-down approach featuring the Internet*. Addison-Wesley.

Resumen

Este módulo ha descrito la capa de red. El objetivo de esta capa es dar conectividad extremo a extremo entre dos puntos cualesquiera, independientemente de la tecnología utilizada. Así, en el nivel de red se definen dos entidades: los encaminadores y los equipos finales. Los primeros tienen la tarea de hacer llegar la información a los segundos. Esto se puede conseguir mediante un protocolo. En la actualidad, IP es el protocolo por excelencia en Internet y permite el intercambio de información dentro de la red. IP requiere que cada equipo tenga una dirección única a nivel global. La versión IPv4 del protocolo define esta dirección con 32 bits, y la nueva versión IPv6 lo hace con 128. Dada la gran cantidad de direcciones y de nodos existentes en la red, IP necesita simplificar la gestión a través de las redes y las subredes, que son conjuntos de direcciones IP pertenecientes a la misma unidad administrativa. De este modo se distribuye la gestión de la red.

Aparte de identificar los nodos de la red, también es necesario tener mecanismos con los que hacer llegar los datagramas entre dos puntos distantes de la red. Por esta razón IP se ayuda de los protocolos de encaminamiento, que definen la política que se ha de seguir para el envío de la información. En el nivel interdominio, el protocolo de encaminamiento más utilizado es BGP, que –mediante los sistemas autónomos– da un nivel suficiente de abstracción de la red que hace viable realizar el encaminamiento globalmente. A una escala más pequeña se encuentran los protocolos basados en el estado del enlace, como OSPF, que se encarga de efectuar el encaminamiento a nivel intradominio. De este modo se completan los mecanismos de encaminamiento y se permite el intercambio de información dentro de Internet de manera transparente a los usuarios finales.

Bibliografía

Dijkstra, Edsger W. (1959). "A note on two problems in connexion with graphs". *Numerische Mathematik* (núm. 1, pág. 269-271).

Gartner, F. C. (2003). "A survey of self-stabilizing spanning tree construction algorithms".

Kurose, James F.; Ross, Keith W. (2005). *Computer Networking: a Top-Down Approach Featuring the Internet.* Addison-Wesley.

Tanenbaum, Andrew S. (2003). *Redes de computadores* (4.ª ed.). Pearson.