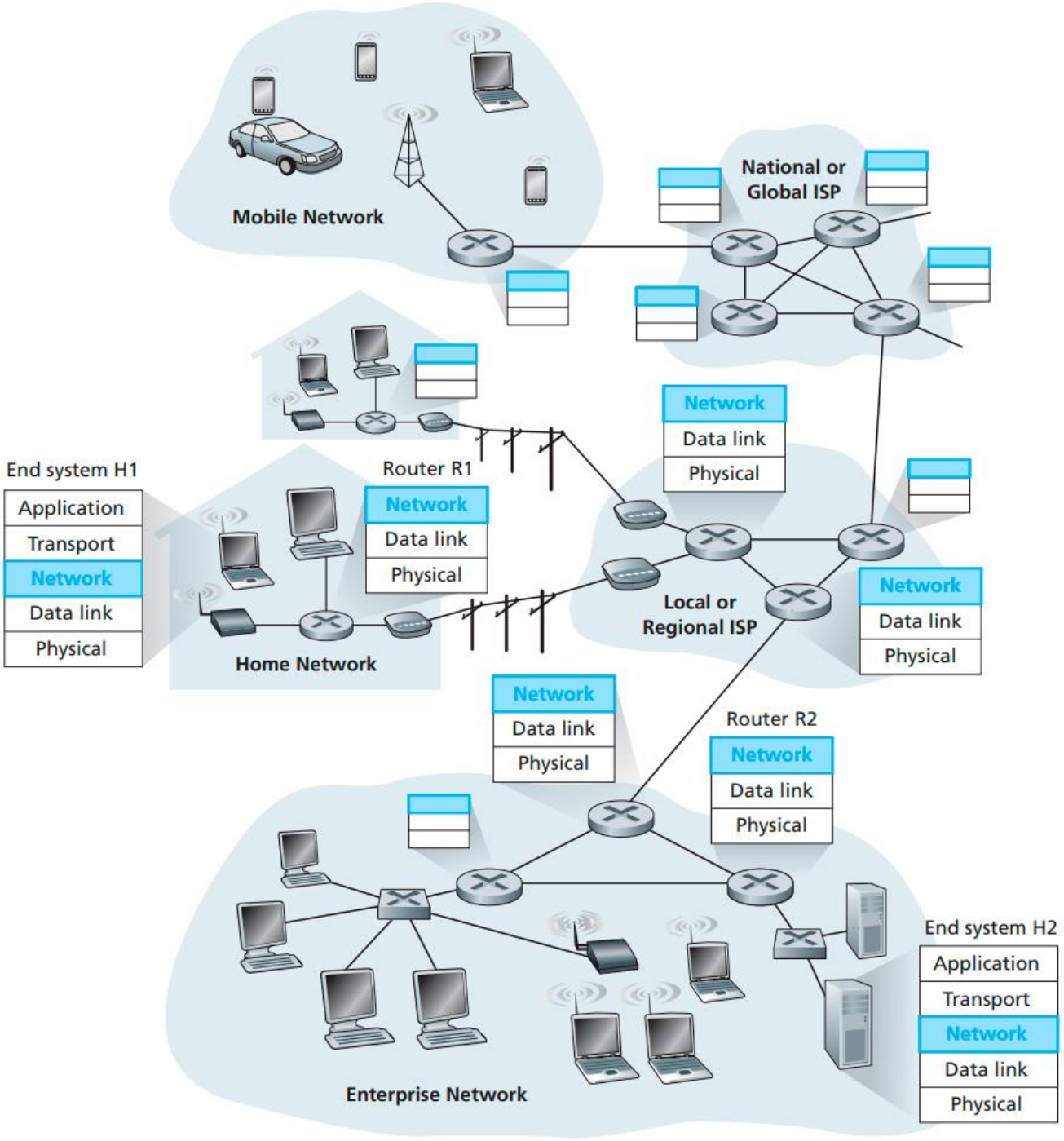
Capa de Red

1. Introducción

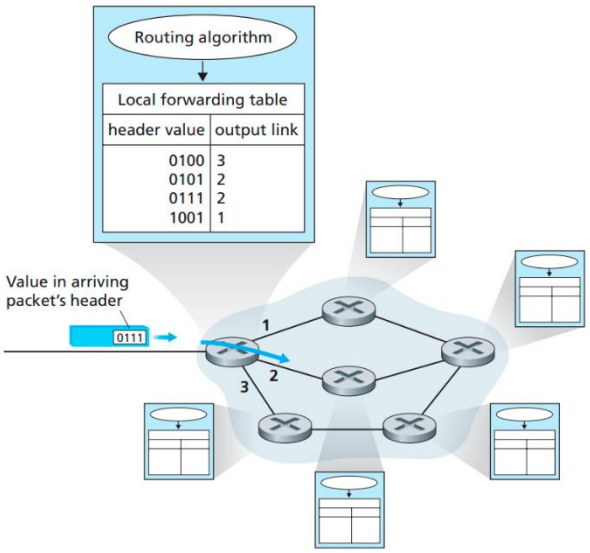


Este es un ejemplo de una red. En este ejemplo se pueden ver dos hosts H1 y H2. Suponemos que H1 trata de enviar información a H2, el host H1 coge la información que quiere enviar desde la capa de transporte y la **encapsula** convirtiendo cada segmento de la información en un **datagrama** (un datagrama no es más que un paquete de red, contiene cabecera física, de enlace y de red). Este datagrama es enviado al router R1, el router con la cabecera de red **reenvía** el datagrama y lo manda a la dirección correspondiente. Podemos ver que la actividad principal de los routers es reenviar datos, también se puede ver que los routers no disponen de más capas a partir de la capa de red.

1.1 Reenviar y enrutar

Estas son las actividades principales de los routers:

- **Reenviar**: Cuando un paquete llega a la entrada al enlace de un router, el router debe enviar este paquete por la salida del enlace de este con la dirección de enlace correcta (MAC ADDRESS). Esto se usa más de manera local.
- **Enrutar**: La capa de red del router se encarga de determinar la dirección correcta (el camino) que debe usar el paquete. Para esto se utilizan **algoritmos de enrutamiento**. Esto se usa en el internet.



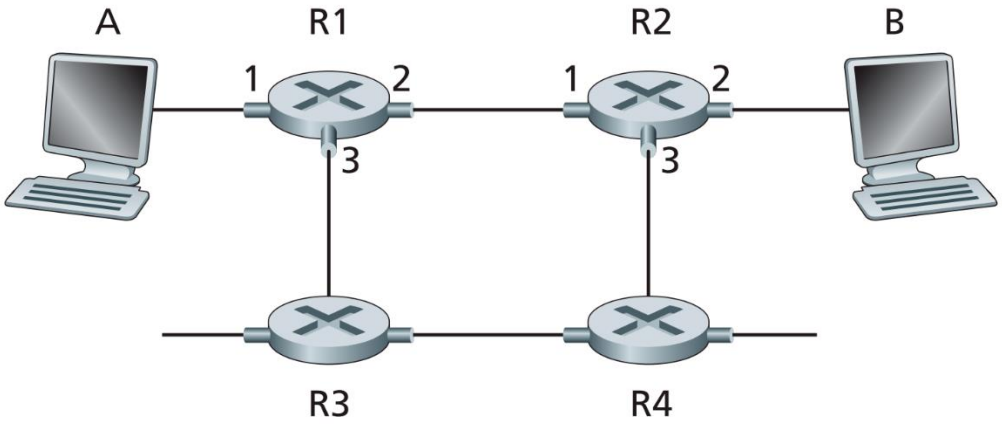
Todos los routers tienen una **tabla de reenvíos**, el router usa la cabecera de red del paquete que le llega, de ahí coge la dirección de destino y luego la compara con los valores en esta tabla, de aquí saca la dirección de reenvío del paquete.

Podemos ver en esta imagen como llega un paquete a un router y este a través de su tabla de reenvíos y de la dirección de destino del paquete decide que output usar.

1.2 Tipos de redes

Tenemos las **redes de datagramas** y las redes de **circuitos virtuales** VC:

- Circuitos virtuales: Los circuitos virtuales consisten en un camino (una serie de links y routers) entre el emisor y el receptor. En estos caminos existen números (*VC numbers*) donde cada link perteneciente a un camino tiene uno de estos números. Por último, en los circuitos virtuales también hay entradas en la tabla de reenvíos de cada router.
  Aquel paquete que pertenezca a un circuito virtual llevara en su cabecera un *VC number*, este VC number es cambiado por los routers, cuando el paquete llega al router este cambia el número dependiendo en función a su tabla de reenvíos.



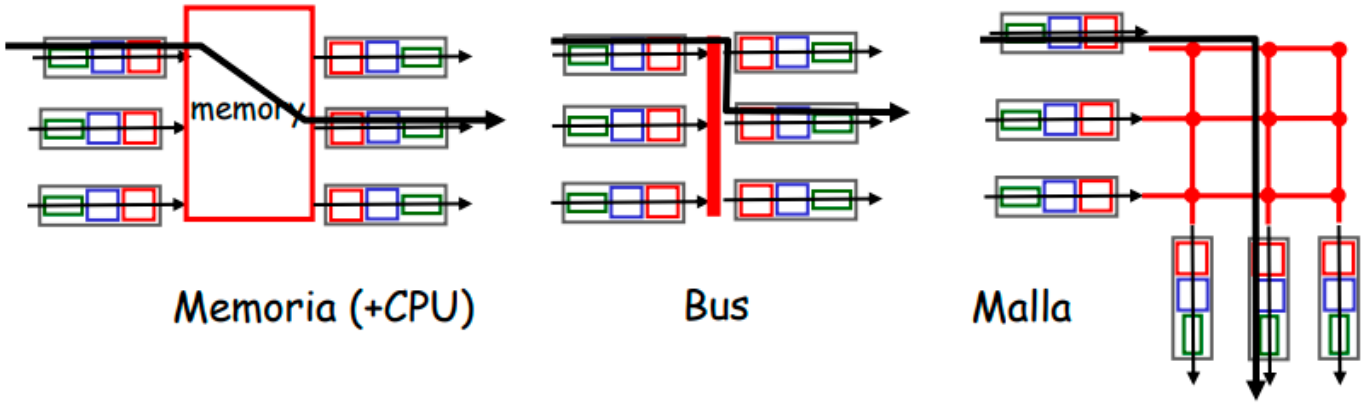
Este tipo de red es adecuada para llamadas telefónicas y es menos robusta ante fallos.

- Redes de datagramas: Este tipo de red es el usado en internet. Cuando se transmite un paquete desde un origen a un destino, este pasa por ciertos routers, con reenvíos y enrutaciones. En este tipo de red el servicio es flexible, y con datagramas se pueden ejecutar controles sobre el rendimiento y recuperarse de errores.

1.3 ¿Qué hay dentro de un router?

Hay 3 partes principales dentro de un router:

- Puertos de entrada: Un puerto de entrada ejecuta varias funciones clave. Ejecuta las funciones de la capa física que traducen señales, también ejecuta las funciones de la capa de enlace para comunicarse con otros equipos en la misma red.
  El puerto de entrada tiene otras funcionalidades a parte:
  - Dada una dirección de destino buscar el puerto de salida adecuado.
  - Debe completar el proceso a tasa de línea (*line speed*).
  - Encolado: Si llegan más datagramas de los que se pueden procesar, se crearan colas en buffers de entrada.
- Malla de conmutación: Se encarga de conectar los puertos de entrada con los puertos de salida. Hay varios tipos de malla de conmutación, se usan unos u otros dependiendo de la velocidad de conmutación, que es la tasa a la que los paquetes pueden ser transferidos desde el puerto de entrada. Los tipos de malla son:



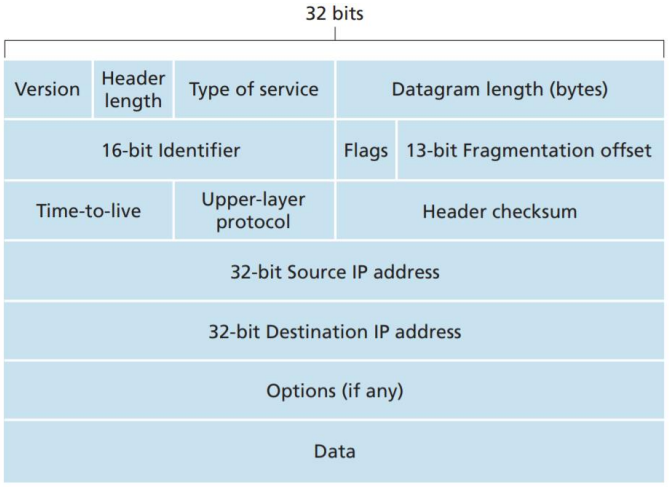
- Puertos de salida: Estos reciben los paquetes de la malla de conmutación y ejecutan las funciones necesarias de las capas de red y enlace para enviar el paquete. Se requieren buffers y tiene que haber un planificador de paquetes para en caso de haber una cola, determinar que paquete transmitir en cierto momento. ¿Cuánto buffer? Puede ser C = 10 Gpbs links: 2.5 Gbit buffer, siendo C la capacidad de enlace y RTT (250ms):

$$\frac{RTT \cdot C}{\sqrt{N}}$$

  Para N flujos.

1.4 IP

- Formato



- Fragmentación y reensamblado

Dependiendo de la interfaz de red, existe un **MTU** (máximum transfer unit – tamaño máximo a transportar por el nivel), esto significa que si se quiere enviar un paquete mayor que el MTU de la interfaz se debe **fragmentar**.

El receptor debe **reensamblar** todos los paquetes, pero surgen ciertos problemas ¿Cuándo sabemos que hemos recibido el último paquete? ¿Cuáles tengo que reensamblar? … Para resolver estos problemas asumiremos que el protocolo IP hace ya todo esto.

Pongamos un ejemplo donde se quiere enviar un paquete de 4000 Bytes (3980 + 20 cabecera), el MTU del enlace es de 1500 bytes, por lo tanto, habrá que fragmentar el paquete. Lo fragmentamos en 3, 1º de 1500B (1480+20), el 2º de 1500B (1480+20) y el último de 1040B (1020+20), hay que sumar la parte útil de los paquetes (hay que descartar el header) y así obtendremos el tamaño original de nuestro paquete. Para ordenarlo incluimos un offset que nos indique la posición del paquete fragmentado con respecto el original, este offset debe ser un múltiplo de 8, por lo tanto, dividimos su posición entre 8. Con posición me refiero a, el primer paquete sería 0/8, el segundo paquete sería 0+1480/8 (justo después del primer paquete) y el tercer paquete sería 0+1480+1480/8 = 2960/8 (justo después del segundo paquete).

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

La fragmentación nos permite enviar paquetes sin tener que preocuparnos del **MTU** de las interfaces en la red, pero el proceso de reensamblar es costoso para los extremos en la comunicación ya que hay que fragmentar y reensamblar. También esto permite hacer ataques en la red ya que se podrían enviar paquetes sin final o con numeración incorrecta.

- Direccionamiento

Los bits altos representan a la red que se quiere llegar, y los bits bajos representan a los hosts que se quieren llegar. Con red nos referimos a las interfaces que contienen la misma parte alta en su dirección. Existen varios tipos de redes, entre estas está la red local, en la cual se pueden comunicar elementos de esta sin la intervención del router.

Para que haya coherencia en las redes se usan diferentes estrategias de enrutamiento como:

- **CIDR**: a.b.c.d/x, donde x es el número de bits que pertenecen a la red. Por ejemplo: 11001100 10001010 00001000 00000000 Red – Interfaz. 204.138.8.0/23
  Se puede ver que se usan 9 bits para la interfaz, 32-9=23, por lo tanto, hay 23 bits útiles para definir una red. En esta red puede haber 512 interfaces, 2^9.
  La máscara de la red es aquella en la cual todos los bits de la parte de red están a 1.
  11111111 11111111 1111110 00000000, 255.255.254.0/23 ó ==/23

  En esta estrategia existen restricciones como que no se pueden asignar a las interfaces direcciones que empiecen por 127, ya que esta es la dirección de loopback.
  No se pueden asignar direcciones donde el campo de la interfaz esta todo a 1's o 0's. La dirección todo a 1's corresponde a la dirección de broadcast, y la dirección todo a 0's identifica los rangos.

  También existen direcciones privadas tales como 10.0.0.0 – 10.255.255.255, 172.16.0.0 – 172.31.255.255, 192.168.0.0 – 192.168.255.255.

- **DHCP**

Las direcciones IP se pueden conseguir mediante **DHCP** (Dynamic Host Configuration Protocol), este protocolo permite obtener dinámicamente una dirección IP para un equipo, es decir, si nosotros nos conectamos a un servidor DHCP, este nos asignara una IP al momento de conectarnos.

El proceso para conectarnos a un servidor DHCP tiene los siguientes pasos:

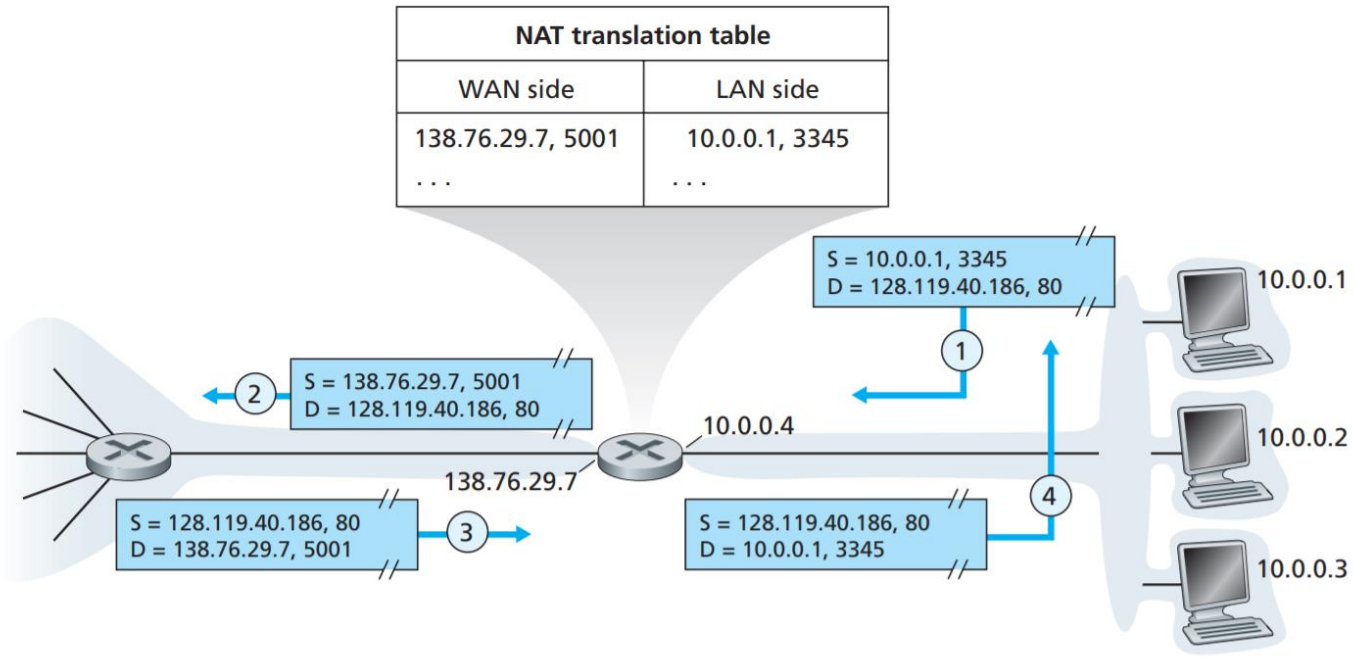
1. El cliente envía al servidor un **DHCP discover** a la dirección broadcast.
2. Los servidores DHCP de la red envían un **DHCP offer**, al cliente.
3. El cliente elige el servidor enviándole un **DHCP request**.
4. Por último, el servidor envía la dirección IP asignada a través de **DHCP ACK**.

La **ICANN** es la institución sin ánimo de lucro encargada de asignar direcciones. Cede el uso de sus direcciones con la condición de que sean activas y que el solicitante sea socio, para ser socio hacen falta 2000$.

- **NAT: Network Address Translation**

Todos los elementos de una red tienen una dirección IP, esta suele ser una dirección local, por lo tanto, cuando se envía un paquete fuera de la red local, el router NAT cambia la dirección origen del paquete por otra (la del propio router) asignándole un puerto. El router tiene una **tabla de direcciones NAT** donde guarda las asignaciones que hace.

Por ejemplo, si el equipo con dirección 10.0.0.1 de una red local envía un paquete fuera de la red, este paquete tendrá como origen 10.0.0.1. Al salir del router, este la cambiará por la suya, pero le asignará un puerto, por lo tanto, la dirección origen del paquete ahora será la del router. En la tabla de traducciones estará guardada la relación entre ambas direcciones. Cuando un paquete de fuera trate de comunicarse con 10.0.0.1, en su destino tendrá la dirección del router y el puerto que el router le asigno, de manera que el router cuando reciba este paquete consultara su tabla de reenvíos para enviar el paquete de manera correcta.



Pero, y si un cliente quiere conectarse a un servidor (o cliente) cuya dirección IP es 10.0.0.1 (privada), para esta situación hay varias maneras de afrontarla:

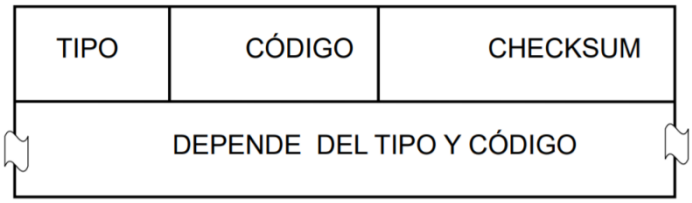
- **Estáticamente**: El router redirige toda la conexión entrante a un puerto dado a una dirección concreta.
- **NAT transversal simple**: En el caso de que el receptor no esté detrás de un router NAT se puede utilizar un host intermedio, este host intermedio debe ser accedido por el receptor. En resumen, el emisor envía un paquete a través de un host intermedio al cual el receptor (que no está detrás de un NAT) debe conectarse.
- **NAT transversal simétrico (STUN)**: En el caso en el que ambos extremos estén detrás de un router NAT, ambos routers deben añadir una entrada en la tabla y ambos deben s que se ha añadido esa entrada, para esto se necesita un servidor intermedio.

- **NAT transversal con retransmisores (TURN)**: Clientes con NAT establecen conexiones a nodos de retransmisión. El otro extremo también se conecta al nodo retransmisor y el retransmisor hace de puente entre ambos nodos.
- **Internet Gateway Device (IGD) Protocolo del conjunto de protocolos Universal Plug and Play (UPnP)**: Permite gestionar y hacer pública la tabla de traducciones. Es como la configuración estática, pero de forma automática y dinámica, ya que modifica la configuración de los puertos NAT.

- **ICMP: Internet Control Message Protocol**

Este protocolo es usado por hosts y routers para comunicar información a nivel de red.

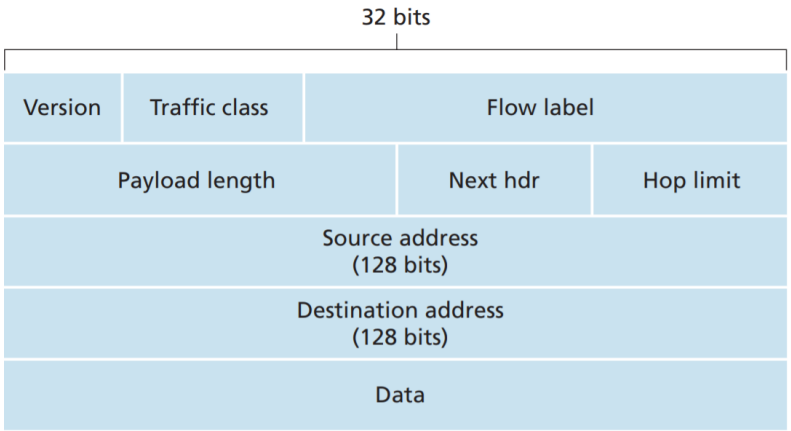
ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad



- **IPv6**

El protocolo IPv6 se creo porque en un futuro las direcciones de 32 bits estarán todas ocupadas.

En este protocolo la cabecera es de 40 bytes y la **fragmentación no está permitida**.

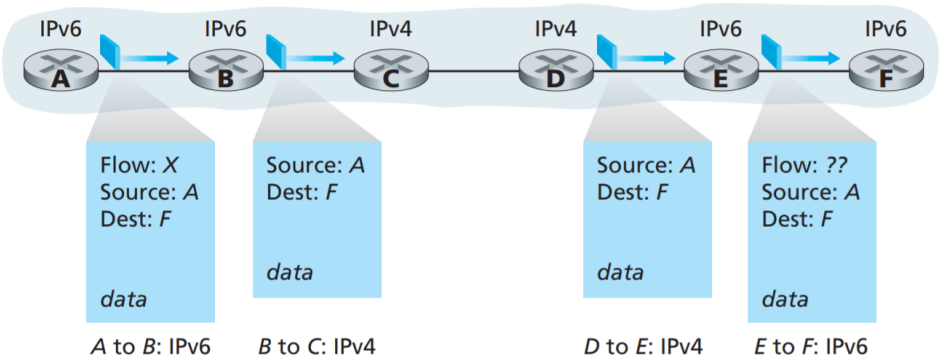


- **Traffic class/Prioridad**: Identifica prioridades entre los datagramas de un flujo.
- **Flow label**: Identifica datagramas en el mismo flujo.
- **Next header**: Identifica protocolo superior para datos.

Existen otros cambios en la cabecera con respecto a IPv4:

- **Checksum**: Eliminado para reducir el costo de procesado por salto.
- **Options**: Permitidas, pero fuera de la cabecera, indicado por el campo “Next Header”.
- **ICMPv6**: Nueva versión de ICMP con nuevos mensajes y funciones de gestión de grupos.

Como no todos los routers están actualizados a IPv6, los paquetes con este protocolo son “tunelados”, esto significa que los routers IPv4 transportan estos paquetes como payload y datos.



C a D envía un paquete el cual contiene el paquete IPv6, en los pasos B-C y D-E se hace la conversión.

1.5 Algoritmos de enrutamiento

Para elaborar algoritmos de enrutamiento se suelen usar grafos. Los grafos resultan útiles en el mundo de las redes. Una de las razones por las que son útiles es porque se pueden asignar costes entre nodos, de manera que se pueden construir algoritmos que busquen el camino más corto y con menos coste.

Al coste se le debe asignar una **métrica** la cual pueden ser la distancia, la longitud media de las colas, retardo…

Todas estas métricas se pueden usar en los algoritmos de manera **global** o **descentralizada**:

- Global: Todos los routers conocen la topología completa y el "coste" de los enlaces (se usan algoritmos de estado de enlace).
- Descentralizado: Los routers solo conocen a sus vecinos y el coste de los enlaces con estos. Se debe usar un proceso iterativo de calculo que implica el intercambio de información con los vecinos (se usan algoritmos basados en vector distancia).

Las métricas también pueden ser **dinámicas o estáticas**:

- Estático: La información en los routers cambia muy poco a lo largo del tiempo (pero cambia).
- Dinámico: La información de los routers cambian más rápido, por lo que se producen actualizaciones periódicas las cuales responden a los cambios en el coste de los enlaces.

- **Estado de enlaces**

El algoritmo de **Dijkstra** tiene en cuenta la topología de la red y el coste de los enlaces son conocidos por todos los nodos.

- **Vector distancia**

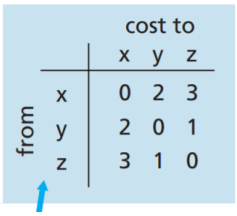
La ecuación de Bellman-Ford nos devuelve el coste de la ruta con menor coste de x a y.

$$d_x(y) = min_v\{c(x,v) + d_v(y)\}$$

La idea del algoritmo de encaminamiento es que periódicamente cada nodo manda su propio vector de distancias (estimado) a sus vecinos. Cuando los vecinos reciben este vector, actualizan el coste al emisor en su tabla de distancias.

El algoritmo es:

- Iterativo: Continúa mientras los nodos se intercambien la información y para de manera automática.
- Asíncrono: No es necesario que los nodos intercambien información en unos momentos determinados ni en un orden fijo.
- Distribuido: Cada nodo intercambia información sólo con sus vecinos inmediatos (los que están justo al lado).
- Estructura de datos, tabla de distancias: Cada nodo tiene su propia tabla donde cada destino ocupa una fila y cada vecino ocupa una columna.



from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

*Ejemplo de tabla

Si un nodo cambia el coste en un camino, este actualiza su información de enrutado y se lo notifica a sus vecinos.

Comparación de algoritmos: Estado de enlace (EE) vs. vector de distancias (VD)

Complejidad de los mensajes

- EE: con n nodos, E enlaces, O(nE) mensajes enviados cada ciclo
- VD: intercambio de mensajes entre vecinos únicamente

Velocidad de convergencia

- EE: Algoritmo $O(n^2)$ que requiere O(nE) mensajes
- VD: tiempo de convergencia variable
  - Puede haber bucles de encaminamiento
  - problema de cuenta al infinito

Robustez: ¿qué pasa si hay errores en el router?

EE:

- Un nodo puede enviar costes de *enlace* erróneos
- Recuperables en la siguiente ejecución
- Cada nodo calcula sólo su propia tabla (errores en el cálculo solo dañinos de forma local)

VD:

- Un nodo puede enviar costes de *rutas* incorrectos
- Los errores se propagan por la red y difícil recuperarse
- La tabla de un nodo se usa para configurar el resto de las tablas (errores cálculos)

- **Enrutamiento jerárquico**

En los algoritmos de enrutamiento jerárquico se hacen grupos de routers formando **sistemas autónomos (SA)**. Los routers en un mismo SA ejecutan los mismos algoritmos de enrutado, mientras que los que están fuera del SA, pueden ejecutar diferentes algoritmos.

Existen **pasarelas/puertas de enlaces (gateway)** que permiten conectar redes, estas pasarelas se encuentran en los bordes de los SA.

Existen dos protocolos de enrutado **intra-AS**, para la parte interna de los SA, e **inter-AS**, para conectar SA's.

El algoritmo inter-AS debe encargarse de encargarse de (en el caso en el que haya que enviar un paquete fuera del SA) enviar un paquete a la pasarela correcta. Para eso de be saber que destinos se pueden alcanzar por las diferentes pasarelas.

Cuando hay que escoger entre varios SA's, se puede usar el algoritmo de la patata caliente (hot potato routing), el cual, envia el paquete hacia la puerta de enlace de menor coste tal y como se ha definido en el algoritmo intra-routing. También se puede hacer al azar, por precio …

1.6 Enrutamiento en internet

Pasaremos ahora a definir el **Intra-AS**. Los protocolos Intra-AS más comunes son:

- RIP (Routing Information Protocol): Se basa en el algoritmo de vector de distancias. La métrica que se usa es la del número de saltos (máximo 15 saltos). Los vectores de distancia son cambiados cada 30 segundos (aprox.) y cada anuncio de un cambio es informado hasta 25 subredes.
  Si durante 180 segundos no se reciben anuncios, el vecino y el enlace con este se consideran muertos, lo que significan que las rutas a través de ese vecino son invalidas. De esto se informa a los otros vecinos, y se emplea envenenamiento inverso.
- OSPF (Open Shortest Path First): Se usa un algoritmo de estado de los enlaces. Se envían paquetes de inundación a todo el SA, de este modo se recupera la topología y los costes. Las rutas se computan usando Dijkstra.

Pasaremos ahora con protocolos **Inter-AS**:

- BGP (Border Gateway Protocol): Este es el protocolo aceptado por todos como algoritmo de enrutado inter-AS. Permite a cada SA:
  - eBGP: Obtener la alcanzabilidad de red de sus SA's vecinos.
  - iBGP: Propagar esta información a todos los routers del SA.
  - Determinar rutas.

  Esto permite a las redes darse a conocer. Si los routers tienen más de una opción para un SA destino, la selección en BGP es en este orden:
  1. Política establecida por el gestor.
  2. Menor número de SA's.
  3. Puerta de enlace con ruta más corta dentro del SA's (patata caliente).
  4. Criterios adicionales.

  Las diferencias entre Intra-AS y Inter-AS son políticas, en Inter-AS la administración puede controlar el trafico que la atraviesa y en Intra-AS solo hay una administración, de escala, depende de la cantidad de nodos en un SA es más razonable un algoritmo de Vector Distancia, de rendimiento, Inter-AS prioriza los términos políticos e Intra-AS el objetivo final.

1.7 Tipo de relaciones

- Peering: Ambos extremos se intercambian rutas de sus clientes, en principio sin coste.
- Cliente-proveedor: El proveedor permite servir como transito para el tráfico entrante saliente de sus clientes.