Nivel red

Tema 5

Alcaraz, S., Roig, P.J.

Fundamentos de Redes de Telecomunicación
Grado en Ingeniería de Tecnologías de la Telecomunicación

Area de Arquitectura y Tecnología de Computadores
Departamento de Física y Arquitectura de Computadores
Universidad Miguel Hernández

{salcaraz,proig}@umh.es

05/01/2025

Objetivos de aprendizaje

Después de abordar la unidad, el estudiante será capaz de:

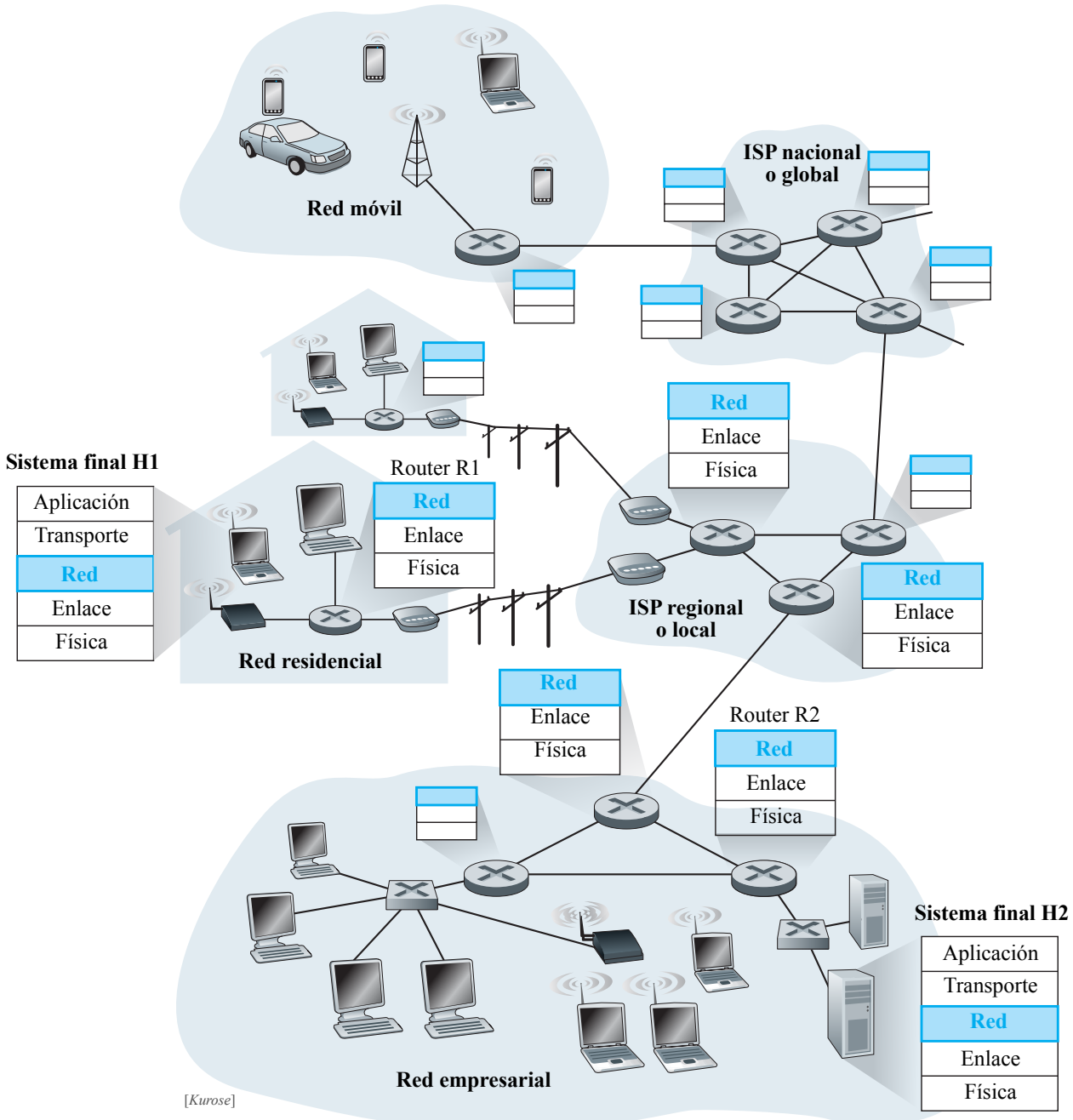
Comprender las funciones del nivel de red.
Comprender el servicio orientado a la conexión
Distinguir las estrategias de conmutación de paquetes y conmutación de circuitos.
Comprender el espacio de direcciones IPv4, tanto *classfull* como *classless*.
Conocer la notación CIDR.
Identificar la necesidad de subnetting así como las diferentes técnicas.
Conocer el concepto de tabla de reenvío.
Comprender la agregación de rutas
Entender la fragmentación a nivel red.
Conocer otros protocolos a nivel red: ARP e ICMP
Entender el concepto de enrutamiento
Entender la visión de redes como grafos.
Comprender el concepto de árbol de mínimo coste.
Conocer los algoritmos de enrutamiento por vector distancias y estado del enlace.
Conocer los protocolos RIP y OSPF.
Entender los algoritmos de Dijkstra y Bellman-Ford.

Nivel de red

1. Introducción al nivel de red
2. Conmutación de paquetes
3. Direccionamiento IPv4
4. Segmentación de redes
5. Reenvío de paquetes IP
6. Fragmentación
7. Otros protocolos de nivel de red
8. Enrutamiento

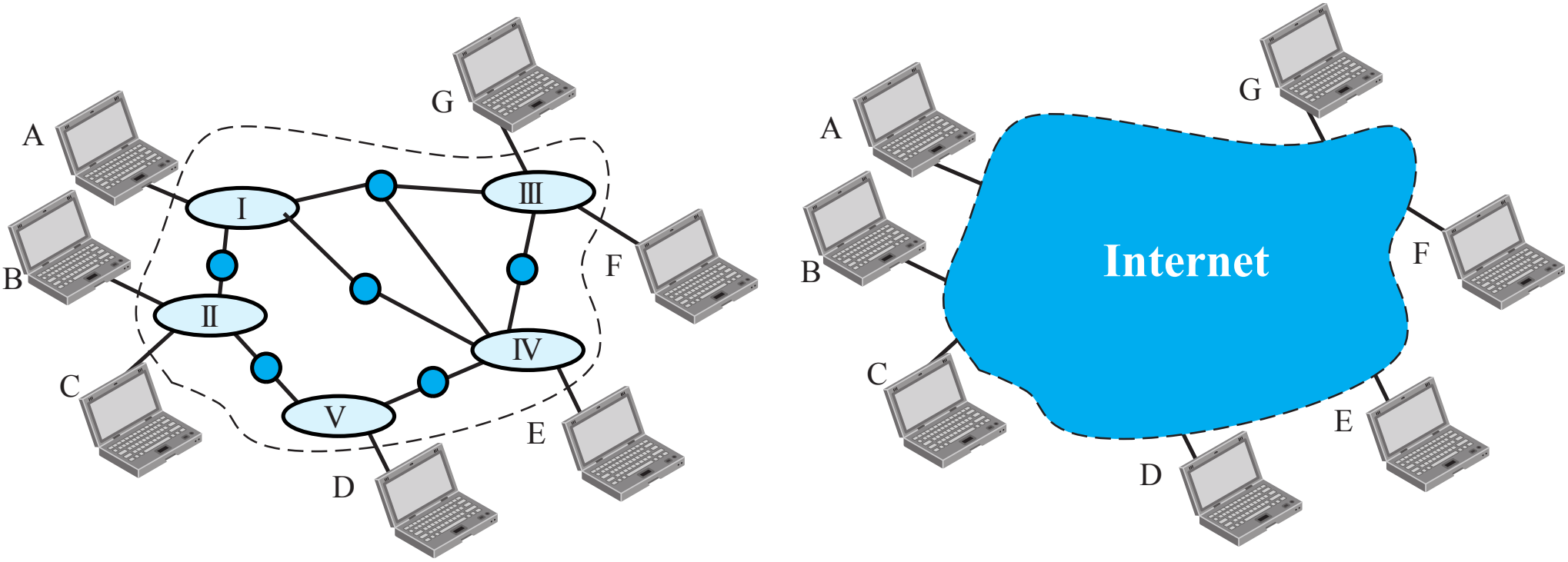
Nivel de red

Nivel de red



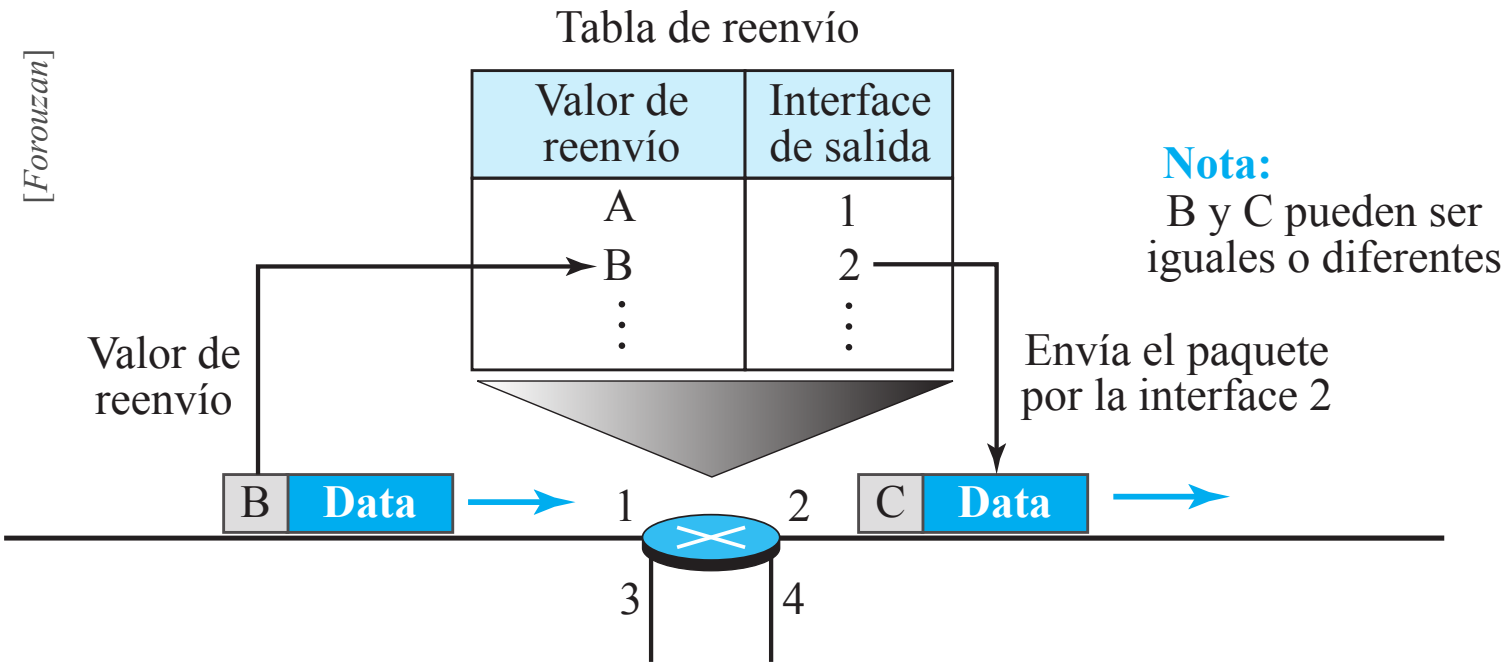
[*Kurose*]

Visión global de Internet (TCP/IP)



Desde el punto de vista de TCP/IP, Internet proporciona conectividad extremo a extremo, ocultando la estructura interna de la red.

Servicios a nivel red



Enrutamiento: el nivel de red es el responsable de encaminar los paquetes desde el origen hasta el destino, a través de la mejor ruta entre las posibles.

Reenvío: proceso por el que un enrutador recibe un paquete, y a partir de la dirección destino del paquete y la tabla de reenvío (tabla de enrutamiento), obtiene el mejor camino para alcanzar el destino, por lo que reenvía el paquete por la interface correspondiente.

Otros servicios a nivel red

Paquetización:

- El nivel de red recibe los datos de nivel superior (*carga*), lo encapsula en un paquete IP y envía al nivel inferior.
- El host destino recibe el paquete IP y entrega la carga al nivel superior.
- Si el paquete es un fragmento, es el responsable de esperar a que lleguen todos los fragmentos, reensamblar y entregar, el paquete completo, al protocolo de nivel superior.
- Los routers no desencapsulan el paquete IP, sólo inspecciona las direcciones IP para reenviar el paquete de forma correcta. No cambia ninguna dirección.

Control de errores: no hay control de errores a nivel red.

Control de flujo: tampoco proporciona control de congestión.

Control de congestión: si el routers se congestiona, descarta paquetes (sin avisar).

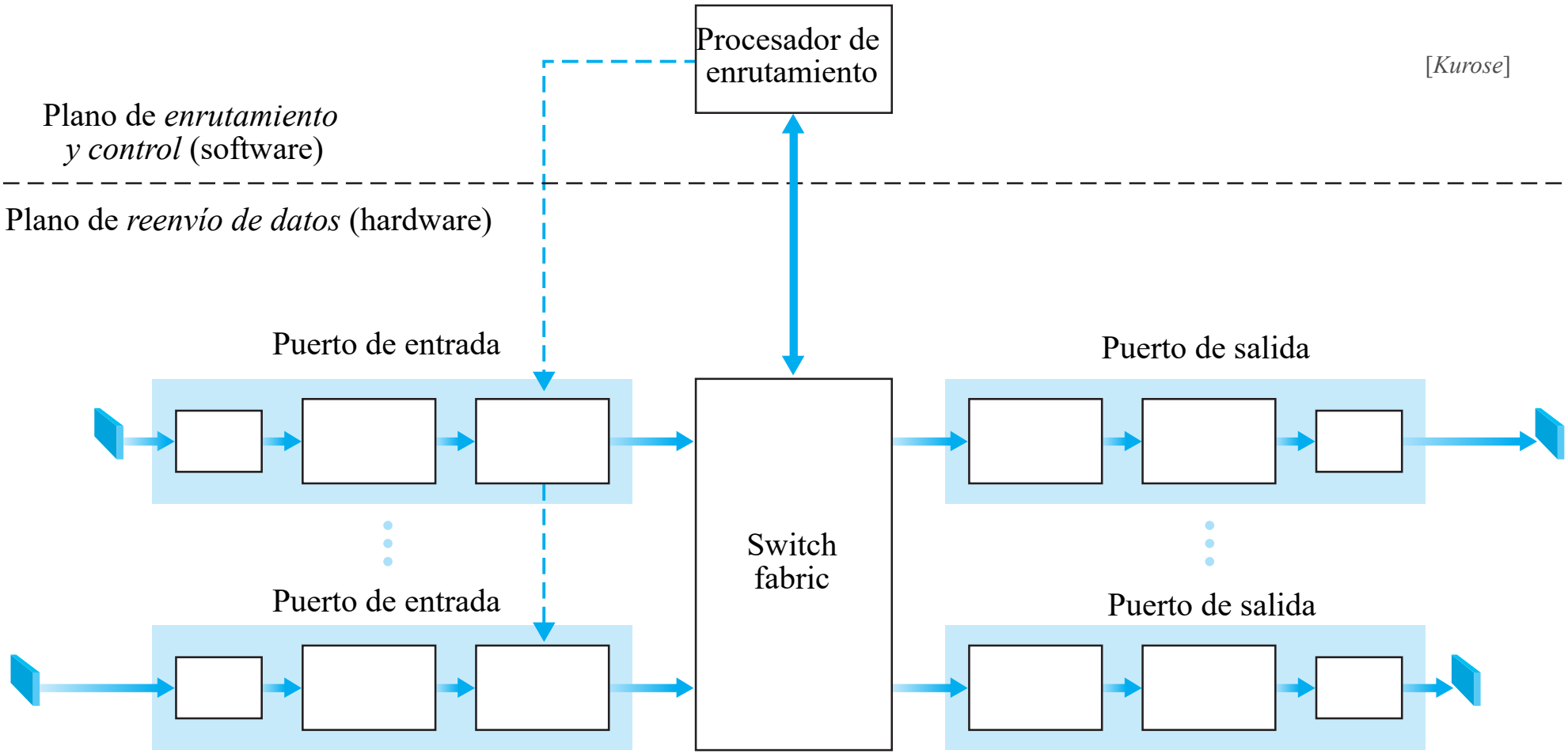
Calidad de servicio: alguna (pero pobre) característica de QoS.

Seguridad: no proporciona servicios de seguridad.

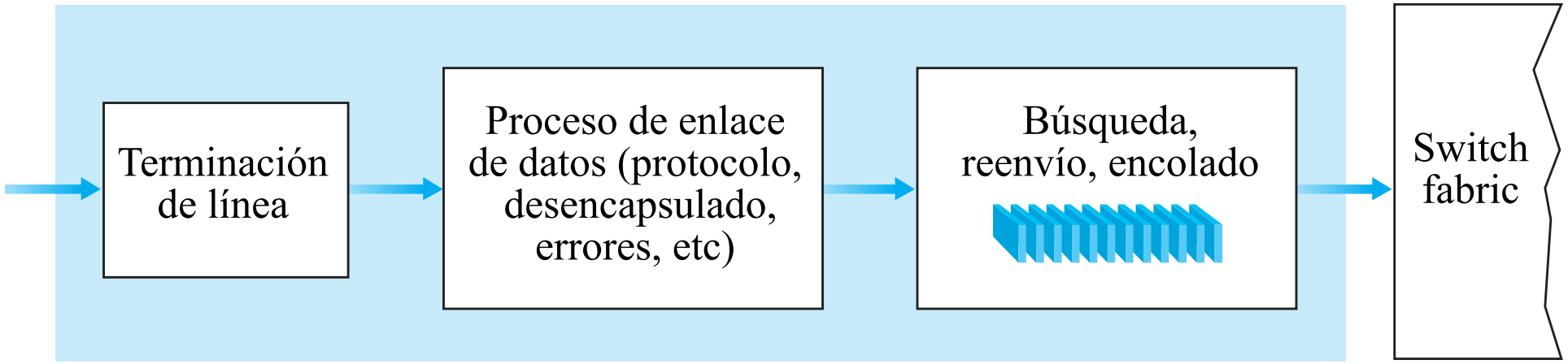
Nivel de red

1. Introducción al nivel de red
2. ***Conmutación de paquetes***
3. Direccionamiento IPv4
4. Segmentación de redes
5. Reenvío de paquetes IP
6. Fragmentación
7. Otros protocolos de nivel de red
8. Enrutamiento

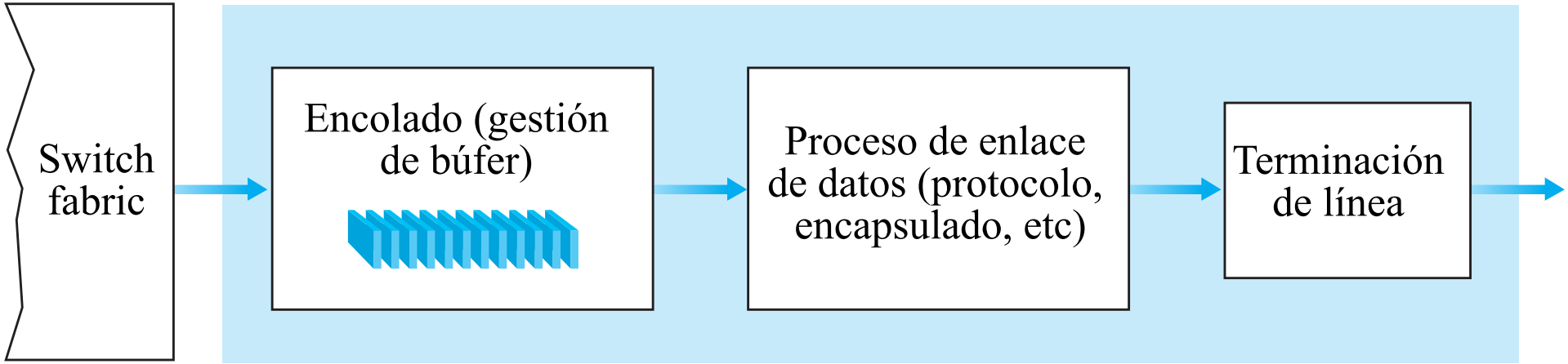
Arquitectura del router



Procesamiento de puertos E/S



(a) Procesamiento en puerto de entrada

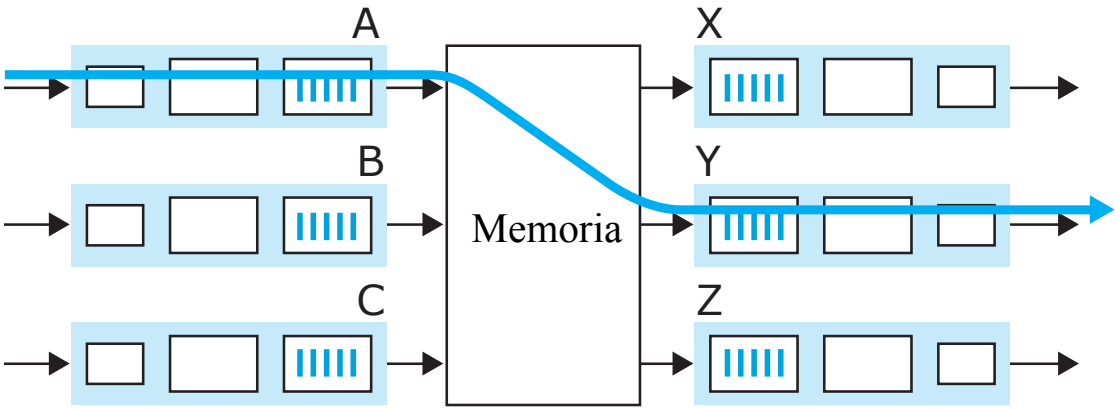


[*Kurose*]

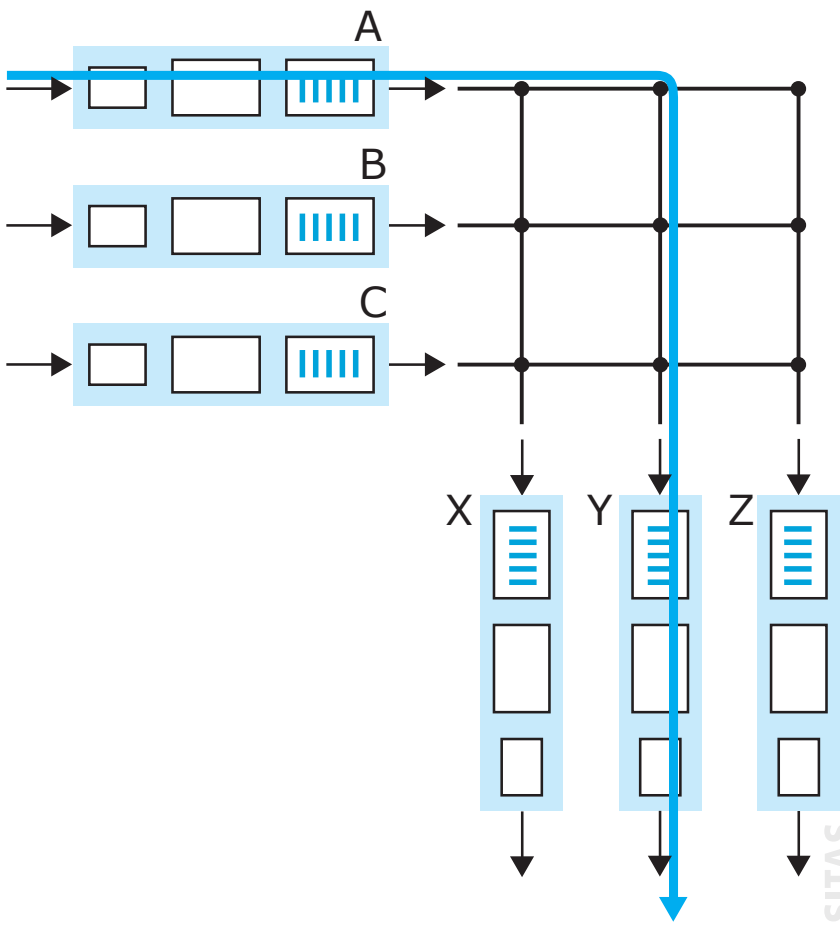
(b) Procesamiento en puerto de salida

Métodos de conmutación

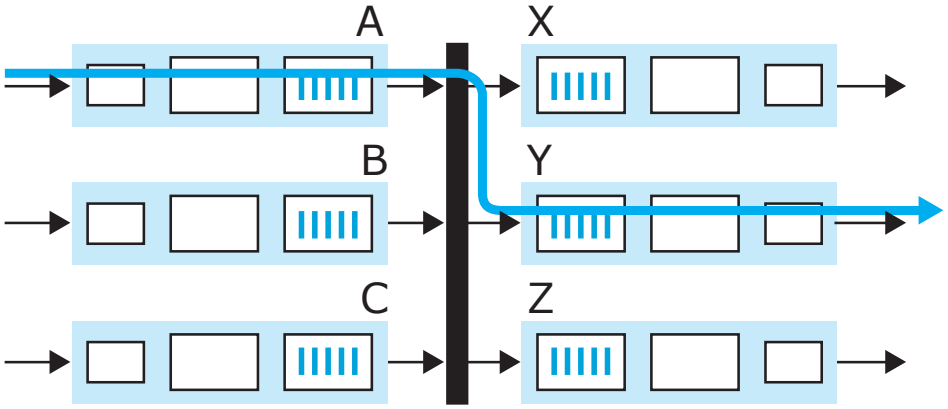
Memoria



Barras cruzadas



Bus



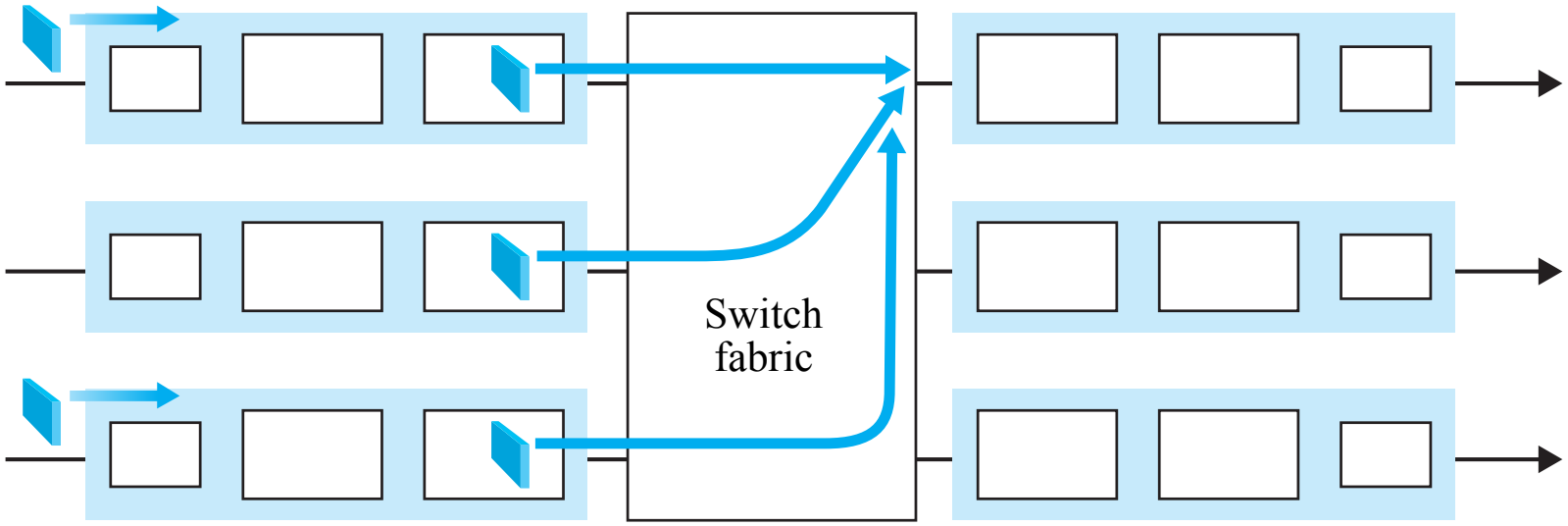
Leyenda:

[*Kurose*]

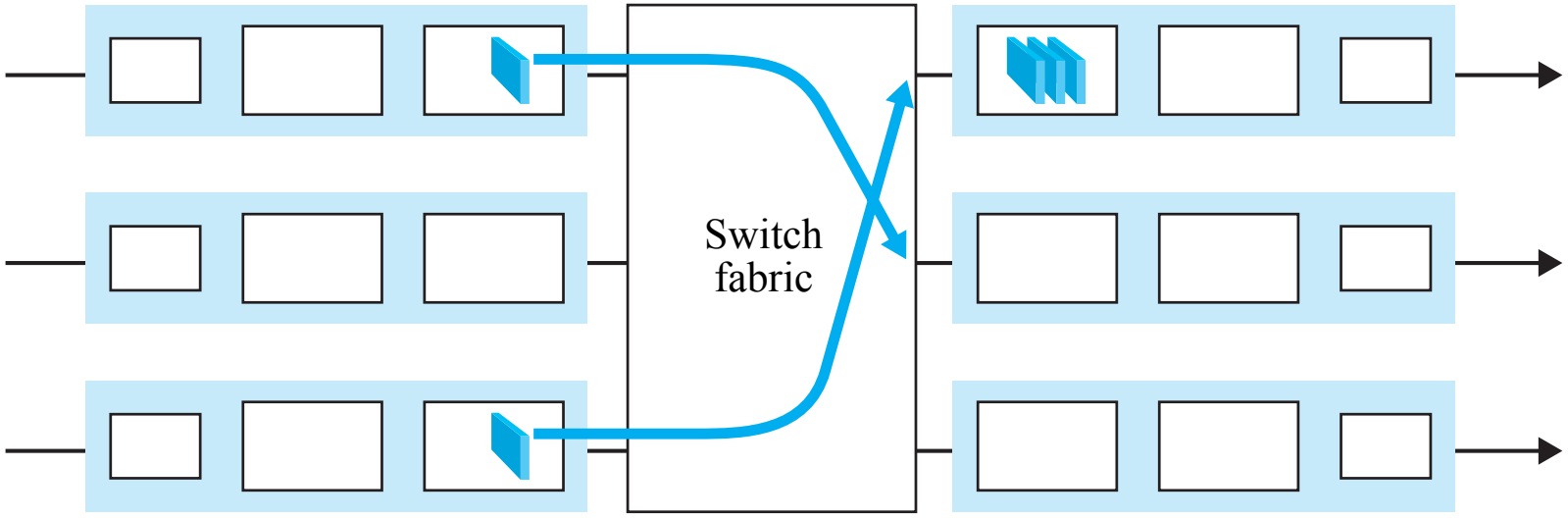
Encolamiento en colas de salida

Contención en puerto de salida en instante *t*

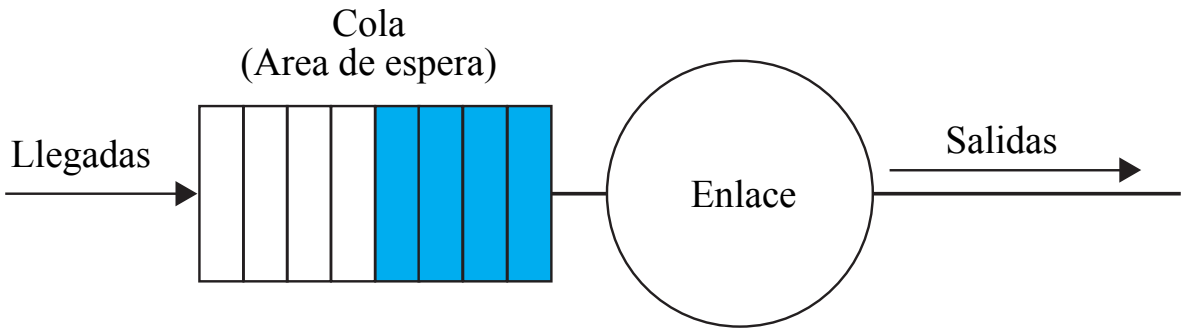
[*Kurose*]

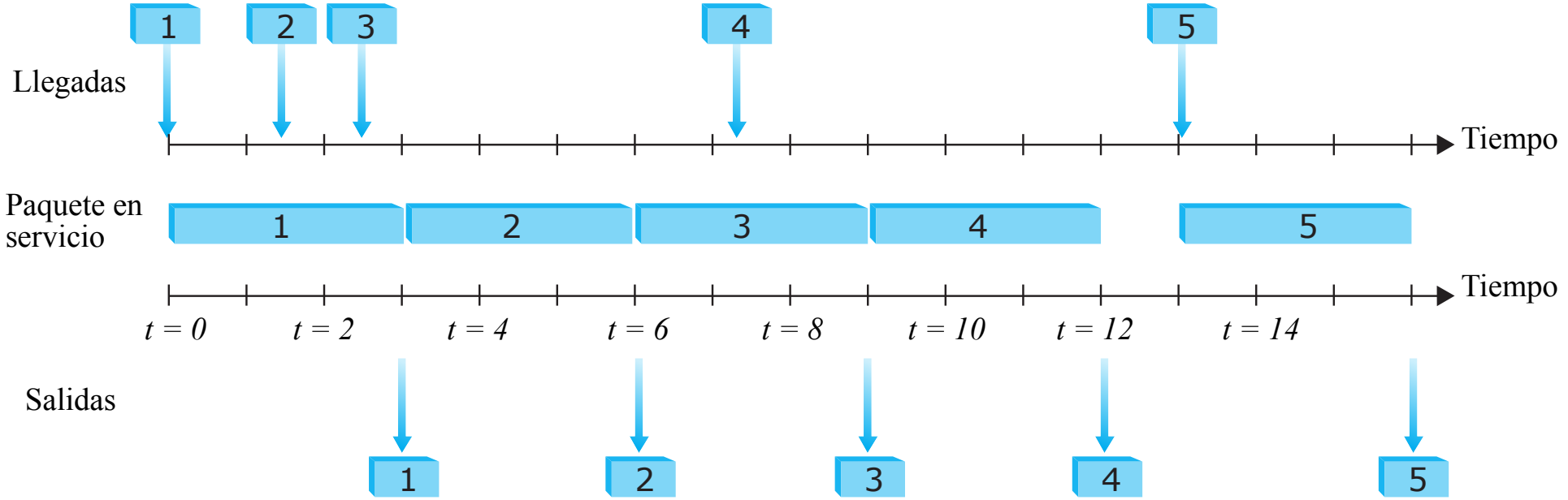


Un paquete un instante más tarde

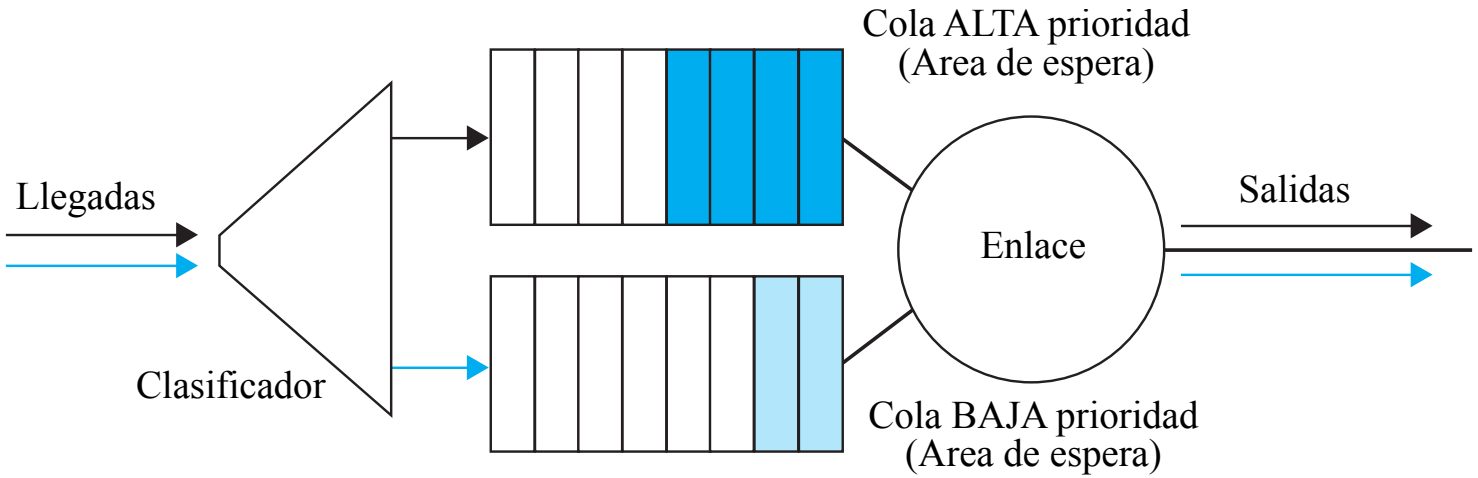


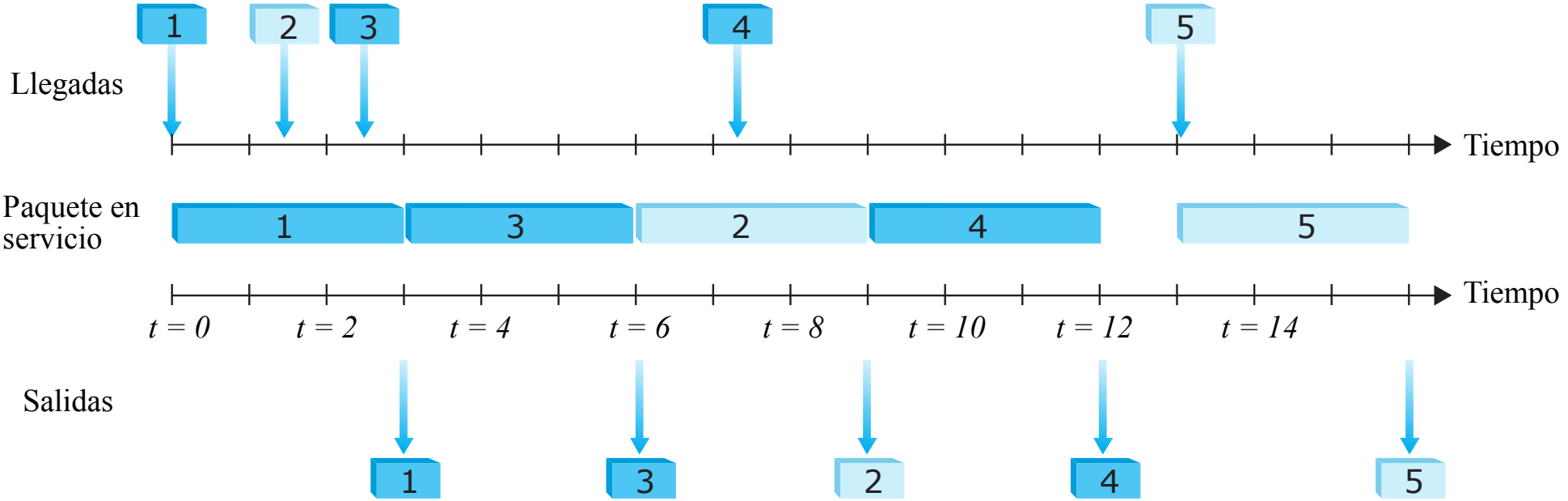
Estrategias de planificación: FIFO



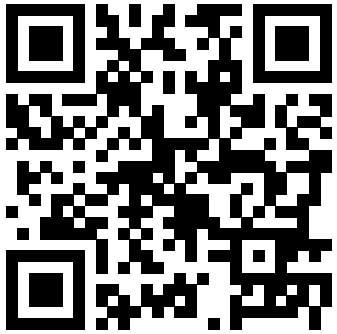


Estrategias de planificación: colas de prioridad

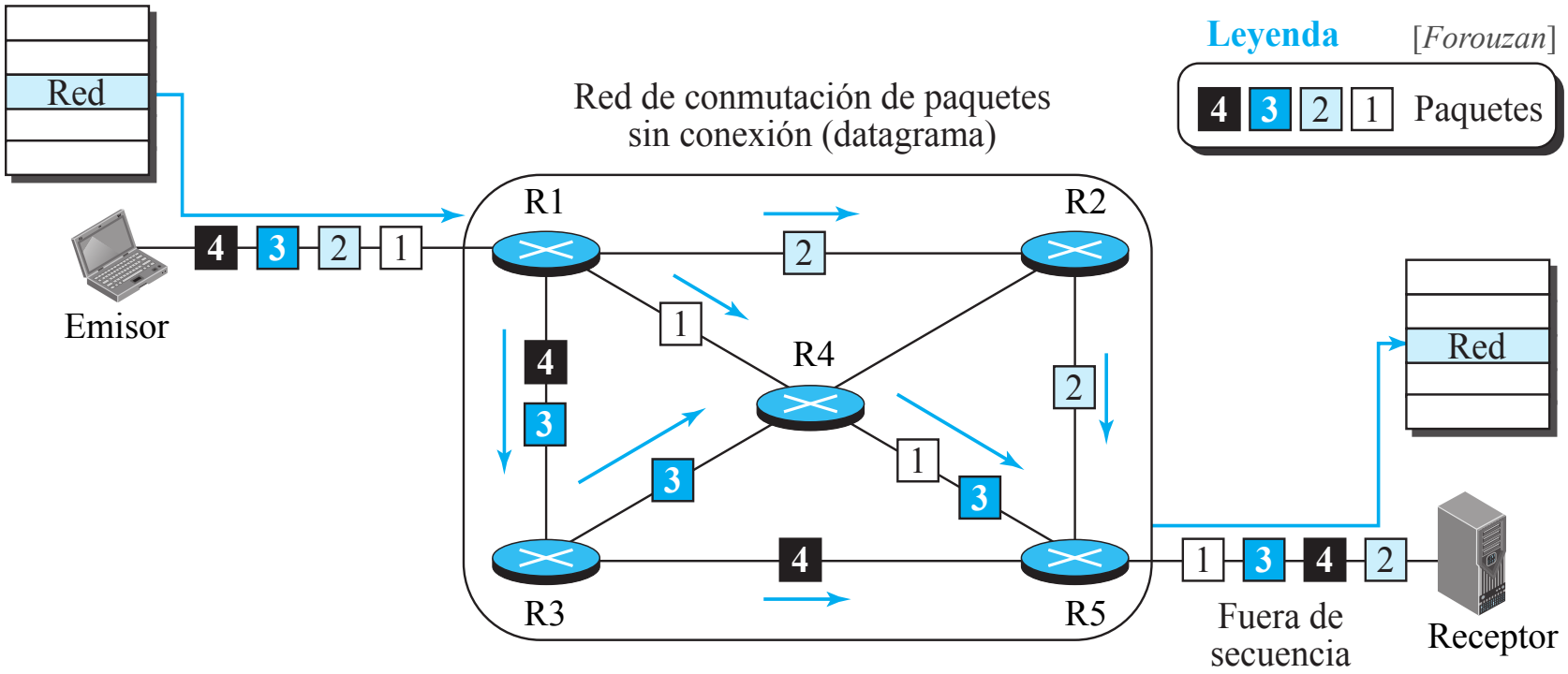




Datagramas



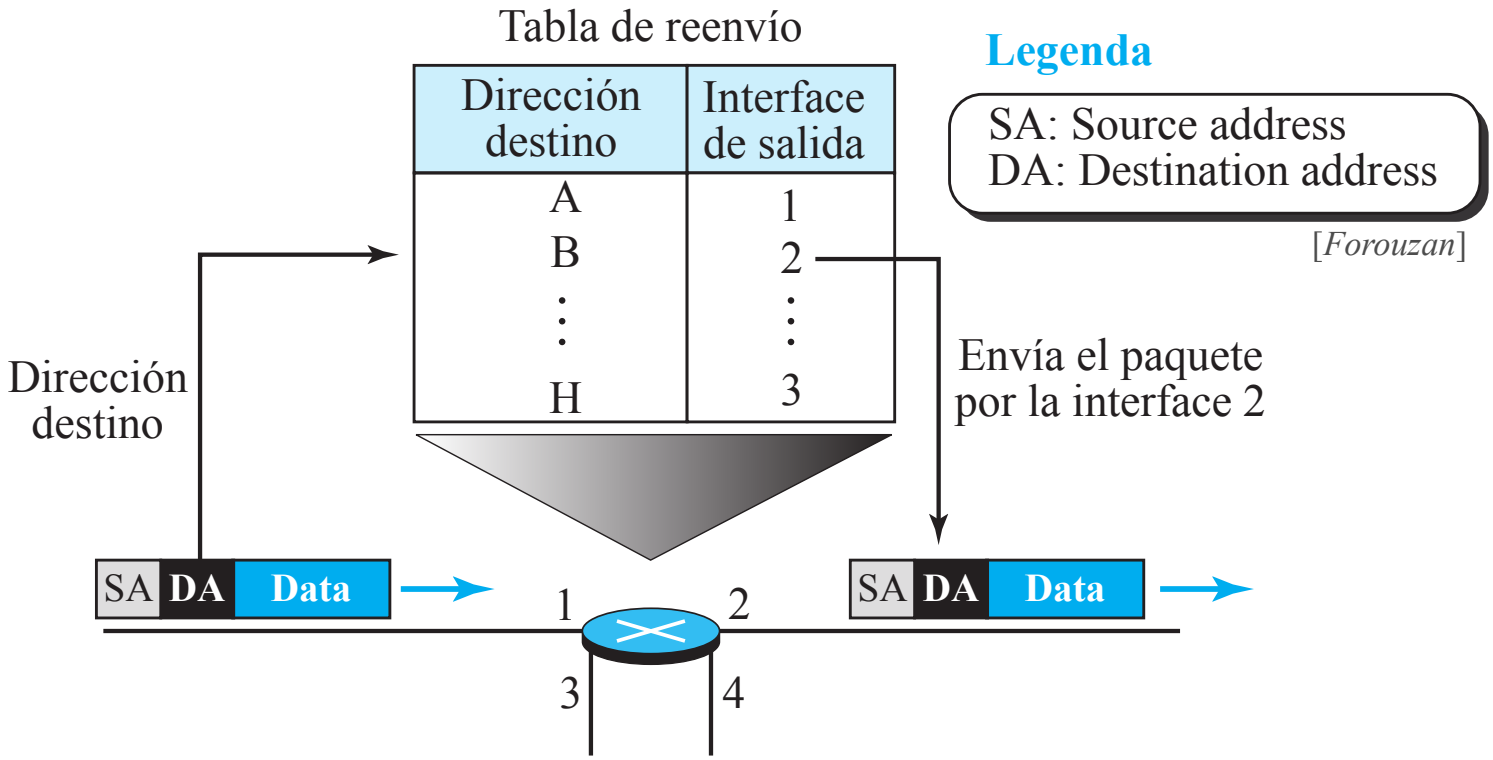
Red de conmutación de paquetes sin conexión



En **redes de conmutación de paquetes**, el nivel de red proporciona un servicio con las siguientes características:

- **Sin conexión**, donde cada paquete se trata de forma independiente.
- Cada paquete procedente de un mismo mensaje, puede seguir caminos diferentes.
- Los conmutadores en este nivel se denominan **routers**, **enrutadores** o **encaminadores**.

Red de conmutación de paquetes sin conexión: reenvío

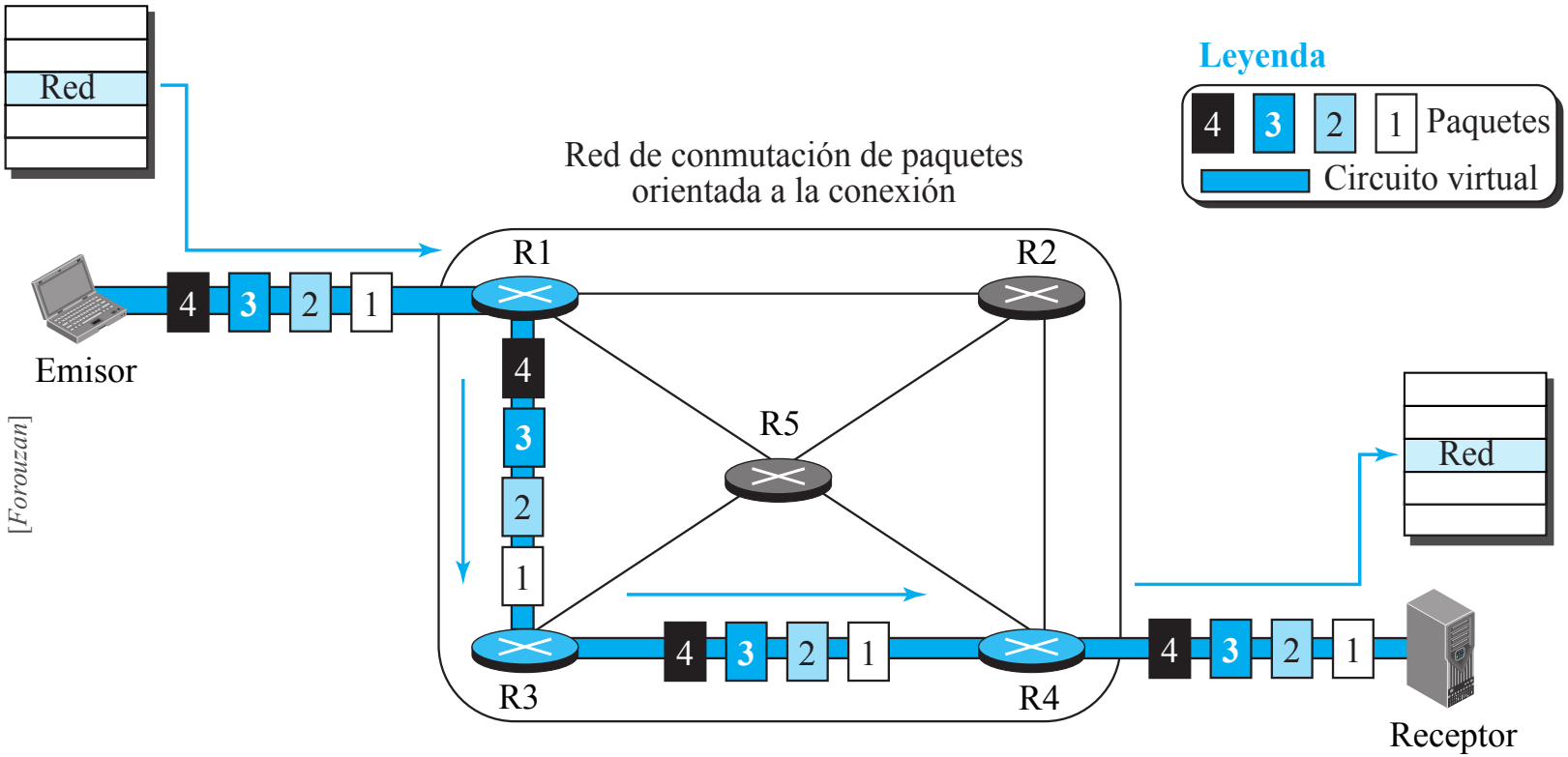


Cada paquete es enrutado a partir de la información contenida en su cabecera: dirección **origen** y **destino**.

La decisión de reenvío se obtiene a partir de la **dirección destino** del paquete y la tabla de reenvío.

Circuitos virtuales

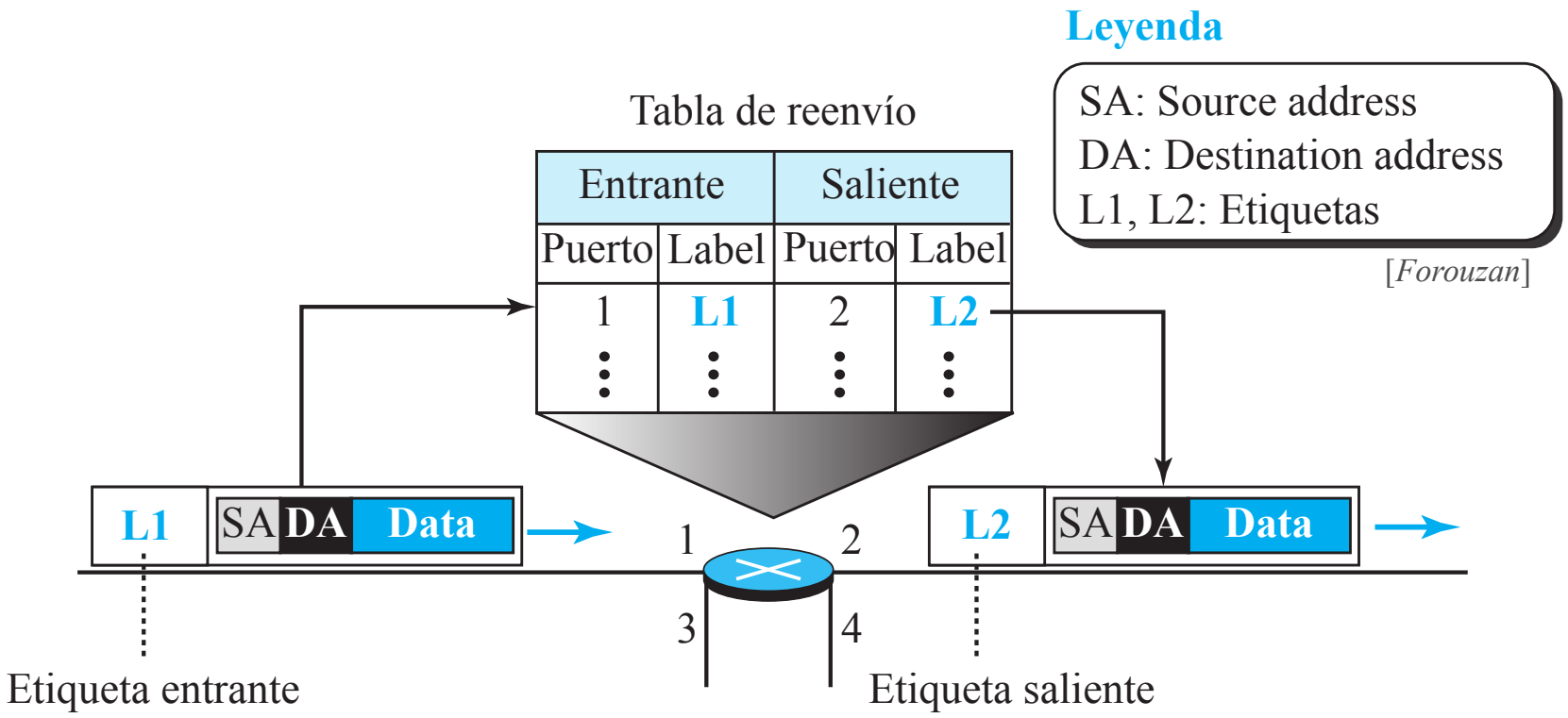
Servicio orientado a la conexión



En **servicio orientado a la conexión** (también se denomina **estrategia circuitos virtuales**) el nivel de red proporciona un servicio con las siguientes características:

- Antes de enviar los paquetes, se establece un circuito virtual que define el camino para los datagramas.
- Después de la conexión, todos los datagramas de un mensaje siguen el mismo camino.
- El datagrama, además de contener las direcciones origen y destino, debe contener las etiquetas de flujo, que define el camino virtual que debe seguir el paquete.

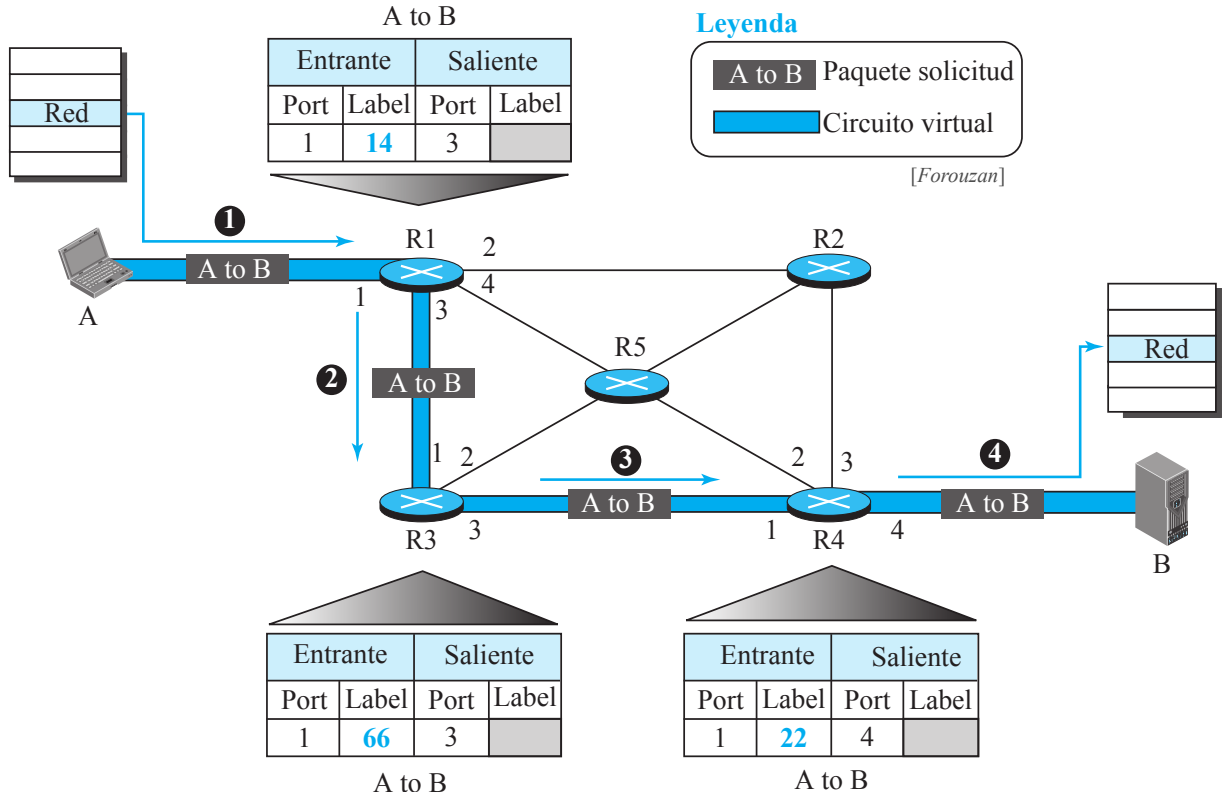
Servicio orientado a la conexión: reenvío



Cada paquete se reenvía a partir de la etiqueta contenida en dicho paquete.

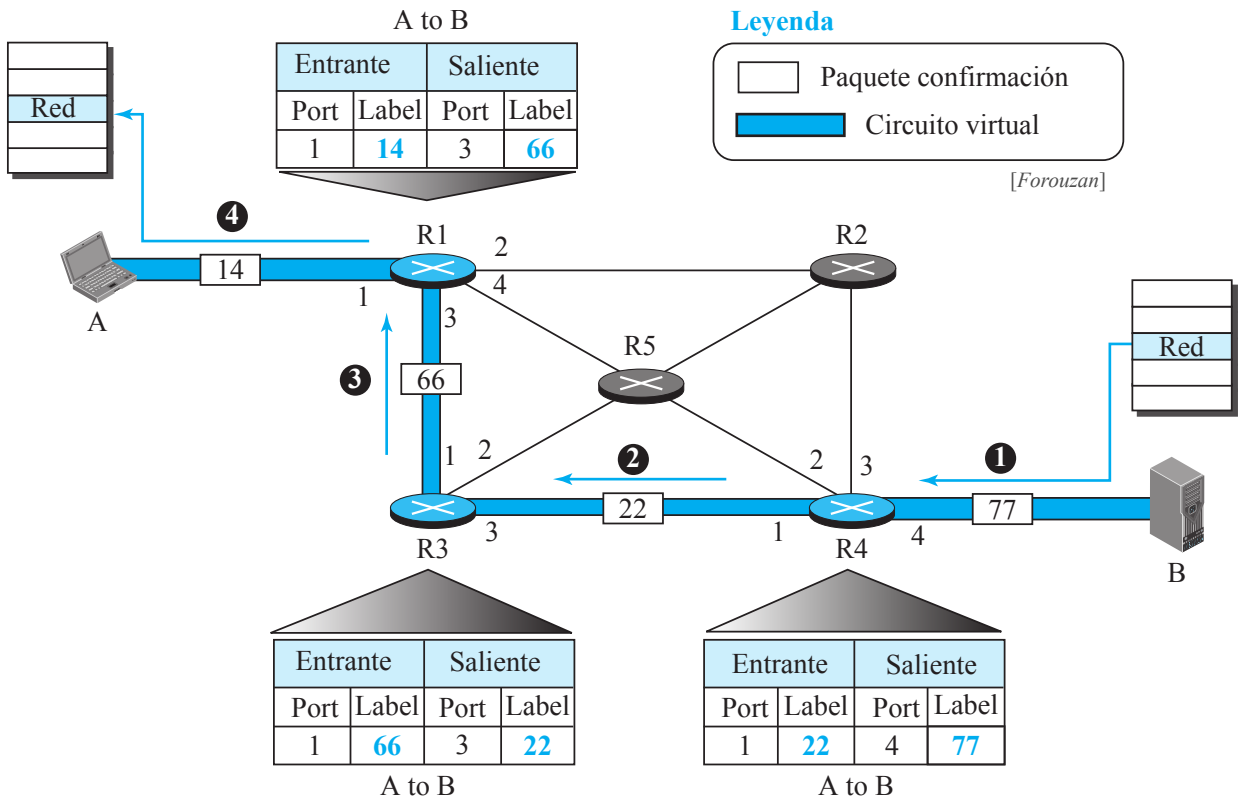
Fases: **establecimiento**, **transferencia** y **liberalización**.

Paquete solicitud en red circuitos virtuales



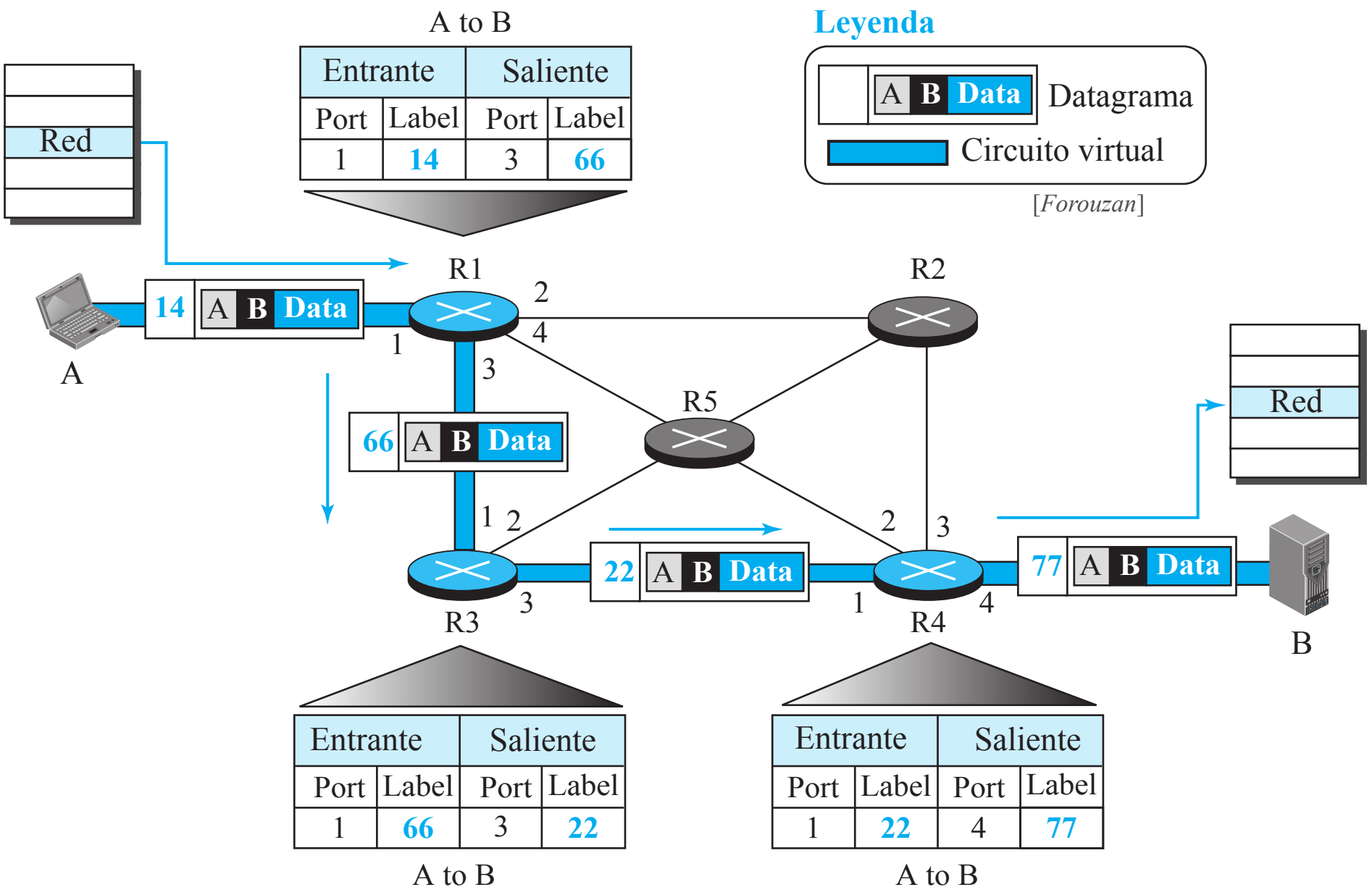
1. La fuente A envía un paquete de solicitud al router R1.
2. R1 sabe que un paquete procedente de A hacia B va a través del puerto 3 (tabla de reenvío). Crea una entrada en la tabla para este circuito virtual (falta por completar la última columna). Asigna el puerto entrante (1), etiqueta entrante (14) y puerto de salida (3). Todavía no conoce la etiqueta de salida, que será establecida en la fase de confirmación. El router envía el paquete a través del puerto 3 hacia R3.
3. Igual que en R1, el router R3 establece: puerto entrante (1), etiqueta entrante (66) y puerto saliente (3).
4. Router R4 establece: puerto entrante (1), etiqueta entrante (22) y puerto saliente (4).
5. El destino B recibe el paquete de configuración, y si está en disposición de recibir paquetes de A, le asigna una etiqueta a los paquetes entrantes procedentes de A, en este caso, 77 (siguiente figura). Esta etiqueta sirve para identificar, en el destino, los paquetes procedentes de A.

Paquete confirmación en red circuitos virtuales



1. El destino envía una confirmación a R4. La confirmación transporta las direcciones globales origen y destino, por lo tanto, el router sabe qué entrada de la tabla debe ser completada. El paquete también contiene la etiqueta 77, elegida por el destino como etiqueta entrante para los paquetes de A. R4 utiliza esta etiqueta para completar la columna de etiqueta saliente (77) para esta entrada.
2. R4 envía la confirmación a R3 que contiene su etiqueta entrante, elegida en la fase de establecimiento. R3 la utiliza como etiqueta saliente.
3. R3 envía una confirmación a R1 que contiene su etiqueta entrante. R1 la utiliza como etiqueta saliente.
4. R1 envía su confirmación a la fuente A que contiene su etiqueta entrante, elegida en la fase de establecimiento.
5. La fuente la utiliza como etiqueta saliente para los paquetes de datos que serán enviados al destino B.

Flujo de un paquete en un circuito virtual establecido



1. Introducción al nivel de red
2. Conmutación de paquetes
3. ***Direccionamiento IPv4***
4. Segmentación de redes
5. Reenvío de paquetes IP
6. Fragmentación
7. Otros protocolos de nivel de red
8. Enrutamiento

Direccionamiento IPv4: notación

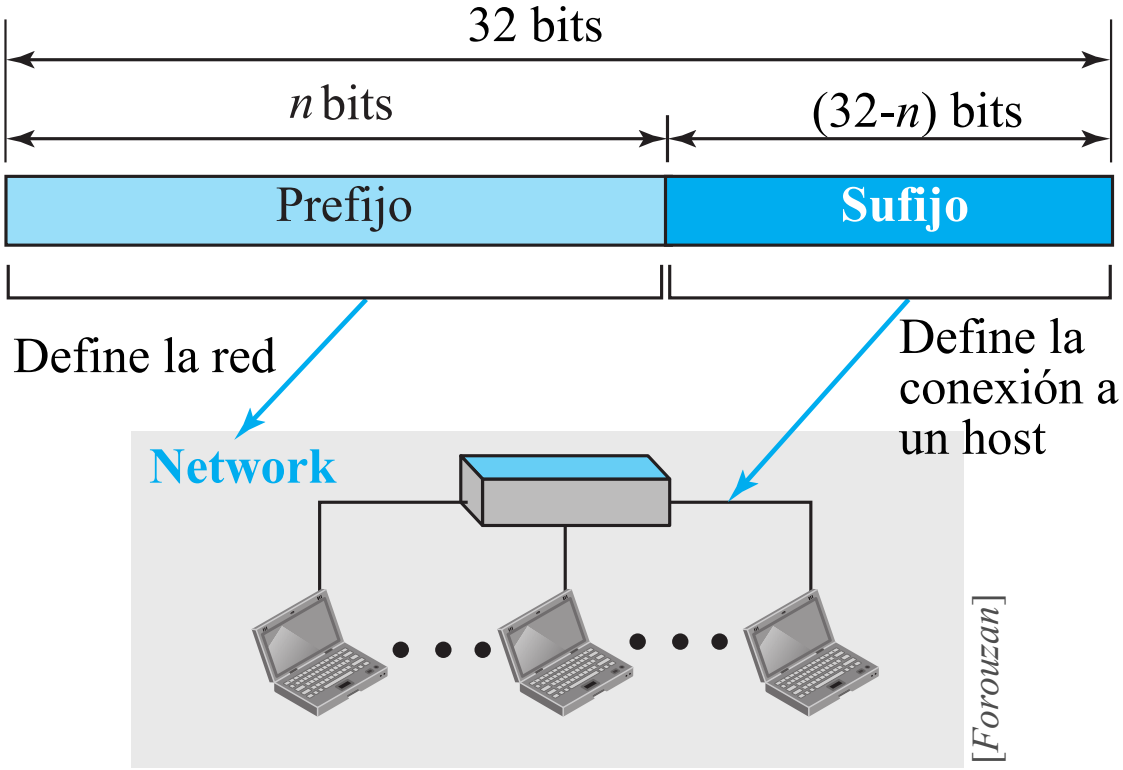
Una dirección IP consta de cuatro bytes (32 bits) que definen una conexión de la estación (host) a la red

[*Forouzan*]

Identificador de red (***prefijo***) Identificador de estación (***sufijo***)

Base 16	C193885F	Notación hexadecimal
Base 2	11000001 10010011 10001000 01011111	Notación binaria
Base 256	**193 . 147 . 136 . 95**	Notación decimal-punto

Espacio de direcciones IPv4



Un protocolo, como IPv4, define un **espacio de direcciones**.

Un espacio de direcciones es el número total de direcciones utilizado por el protocolo.

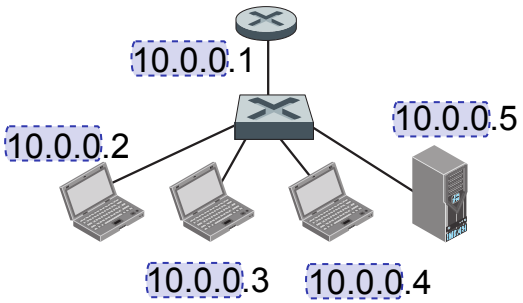
En general, un protocolo utiliza b bits para definir una dirección, entonces, el espacio de direcciones es 2^b.

IPv4, utiliza 32 bits, por lo tanto, su espacio de direcciones es:

$$2^{32} = 4\,294\,967\,296 \text{ direcciones}$$

Tipos de direcciones IPv4: dirección de red

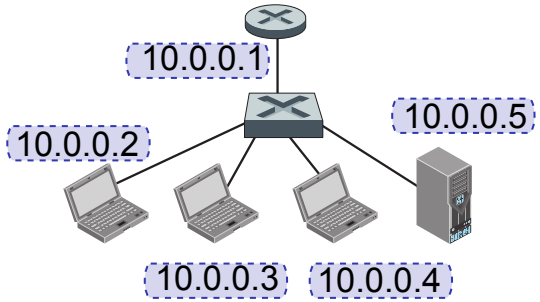
	Red			Host
Dirección *red*	10	0	0	0
	00001010	00000000	00000000	00000000
	10	0	0	255
	00001010	00000000	00000000	11111111
	10	0	0	5
	00001010	00000000	00000000	00000101



- También se denomina **prefijo**.
- **Dirección de red** es aquella contiene 0s en el sufijo.
- Es la porción de bits que comparten todos los hosts de una red.
- Es una dirección reservada (no se puede utilizar como dirección de host).

Tipos de direcciones IPv4: dirección de broadcast

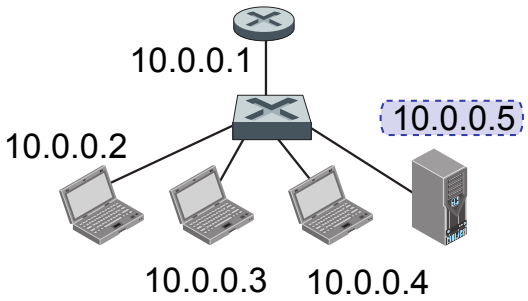
	Red			Host
	10	0	0	0
	00001010	00000000	00000000	00000000
Dirección *broadcast*	10	0	0	255
	00001010	00000000	00000000	11111111
	10	0	0	5
	00001010	00000000	00000000	00000101



- **Dirección broadcast** es aquella que tiene 1s el sufijo.
- Dirección reconocida por todos los hosts de una red.
- Es una dirección reservada (no se puede utilizar como dirección de host).

Tipos de direcciones IPv4: dirección de host

	Red			Host
	10	0	0	0
	00001010	00000000	00000000	00000000
	10	0	0	255
	00001010	00000000	00000000	11111111
Dirección *host*	10	0	0	5
	00001010	00000000	00000000	00000101



- **Dirección de host** es la que se asigna a un host (dispositivo, interface, etc).
- Debe de ser única.
- Corresponde al **sufijo**.

Asignación de direcciones IP

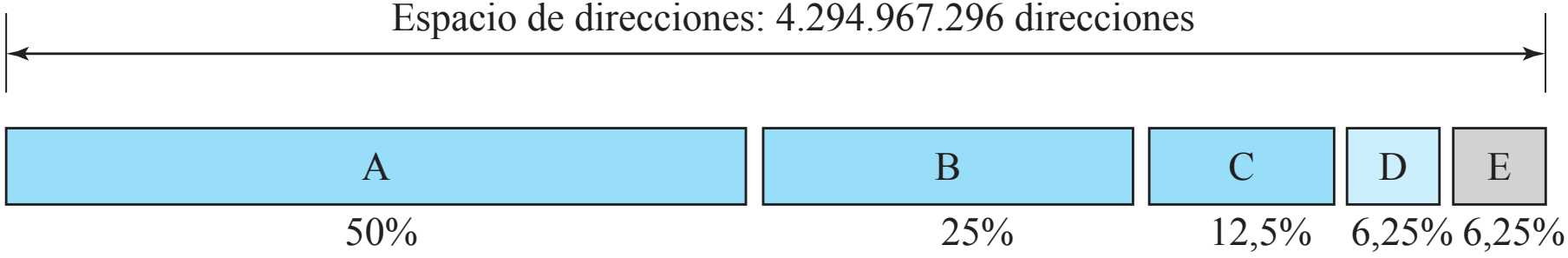
En la planificación de asignaciones de direcciones IP hay que considerar ciertos aspectos:

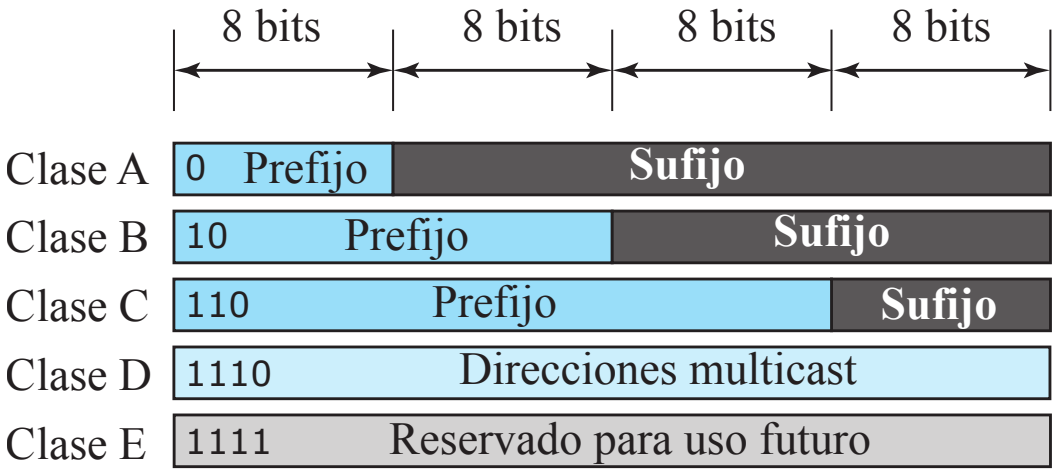
- Evitar duplicación de direcciones.
- Proporcionar y controlar el acceso.
- Controlar seguridad y rendimiento.
- Segmentación de redes.

Las direcciones IP (útiles) se utilizan en los siguientes dispositivos:

- Dispositivos finales para usuarios: host, impresoras, teléfonos IP, etc.
- Servidores y periféricos.
- Hosts a los que se accede desde Internet.
- Dispositivos intermediarios (routers, una IP por interface).

Direccionamiento IP (*Classfull*)



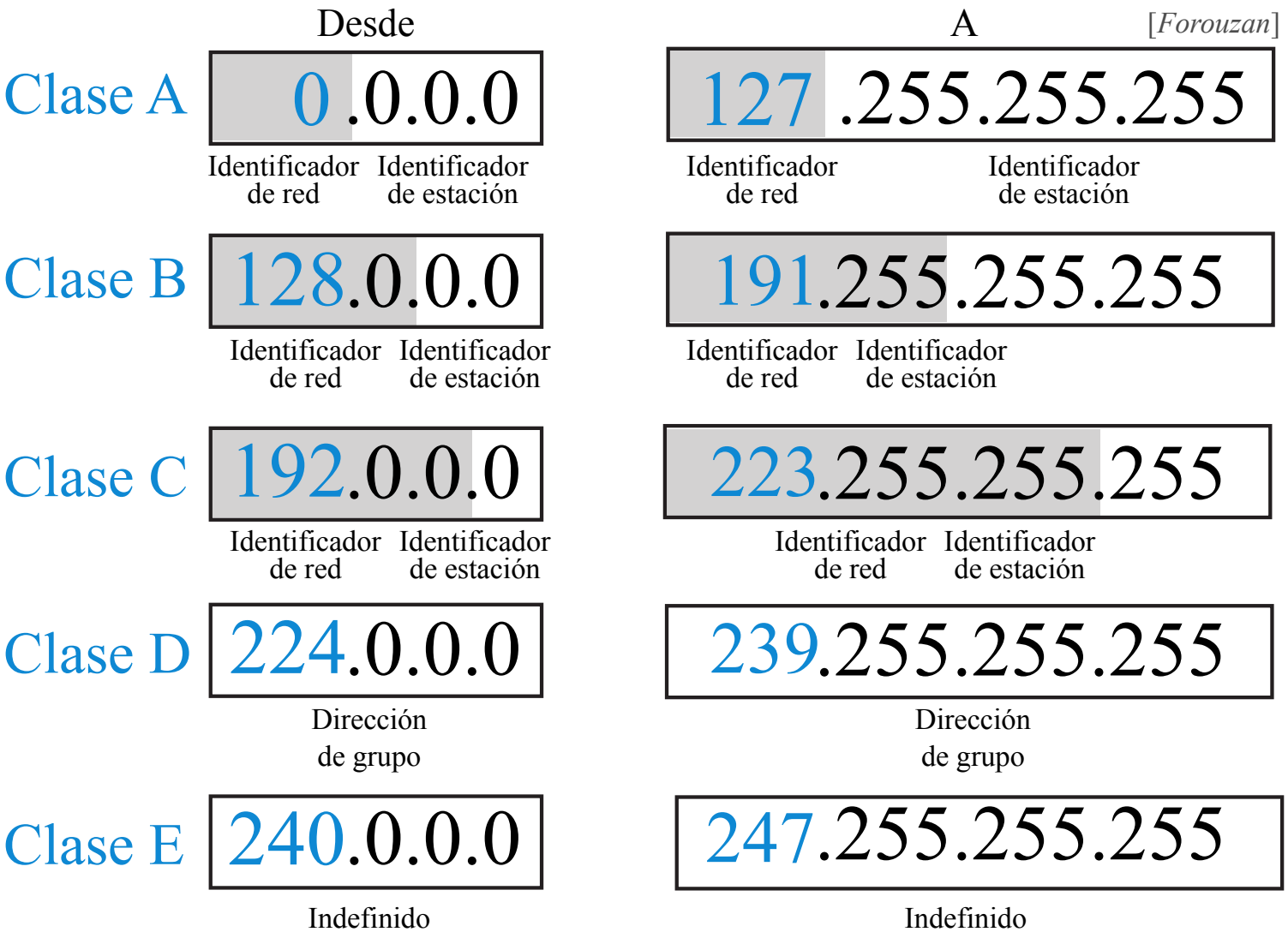


Clase	Prefijo	Primer byte
A	$n = 8$ bits	0 a 127
B	$n = 16$ bits	128 a 191
C	$n = 24$ bits	192 to 223
D	No aplicable	224 a 239
E	No aplicable	240 a 255

[*Forouzan*]

En el modo de direccionamiento con clase *Classfull*, las clases estándar vienen definidas por sus rangos y tienen una máscara de red por defecto

Direccionamiento IP (*Classfull*). Rango de direcciones



Direccionamiento IP (*Classfull*): consideraciones

La razón por la que el direccionamiento con clase está en desuso es por el **agotamiento** de direcciones:

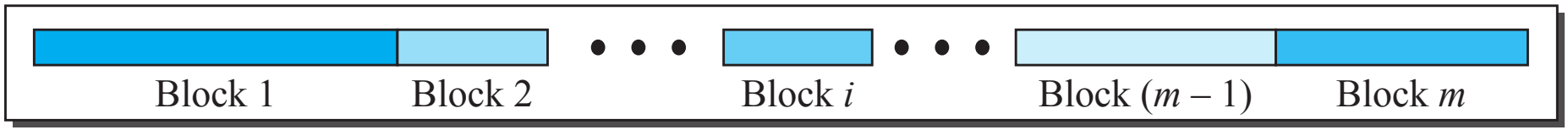
- En redes de Clase A, sólo pueden ser asignadas a 128 organizaciones en el mundo, y cada red de esta clase estará formada por 16.777.213 estaciones.
- Las direcciones de Clase B fueron diseñadas para organizaciones de tamaño medio, no obstante, muchas de estas direcciones permanecían sin utilizar.
- En las redes de Clase C se pueden utilizar hasta 256 direcciones, por lo que en ocasiones, es demasiado pequeño el conjunto de direcciones.

La principal ventaja del direccionamiento con clase es que para una dirección determinada, su clase está predeterminada, y por lo tanto, el prefijo asociado, por lo que no es necesario añadir información adicional para extraer el prefijo y sufijo.

Para aliviar el problema del agotamiento de direcciones, se propusieron varias soluciones:

- Subnetting
- Direcciones privadas + NAT.

Direcciones *Classless*



Espacio de direccionamiento

En 1996 se establece una nueva arquitectura en el espacio de direcciones IP, denominado **Classless**.

Elimina las restricciones de *classfull*, de forma que los prefijos y sufijos ya no vengan definidos, sino que pueden ser establecidos según necesidad.

El total del espacio de direcciones queda dividido en bloques de longitud variable.

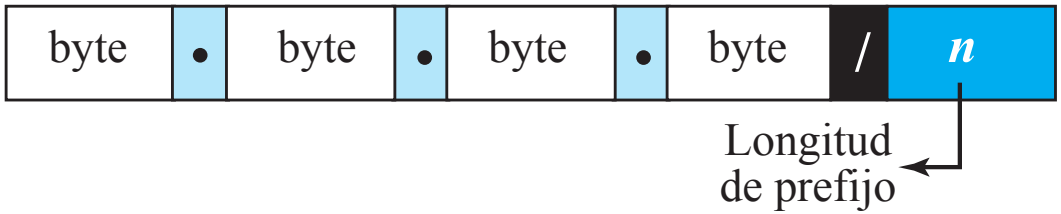
El prefijo en una dirección define el bloque (red) y el sufijo define el nodo (dispositivo).

Teóricamente se pueden definir bloques de $2^1, 2^2, \cdots, 2^{32}$ direcciones.

El número de direcciones en un bloque necesita ser potencia de 2

La mayor ventaja es que ahora, los bloques de direcciones pueden ser de longitud variable, adaptando los bloques a las necesidades de direccionamiento

Direcciones *Classless*: notación CIDR



Ejemplos:
12.24.76.8/**8**
23.14.67.92/**12**
220.8.24.255/**25**

¿Cómo encontrar la longitud de prefijo para una dirección?

Como ahora la longitud de prefijo no es inherente a la dirección, es necesario proporcionarlo de forma separada.

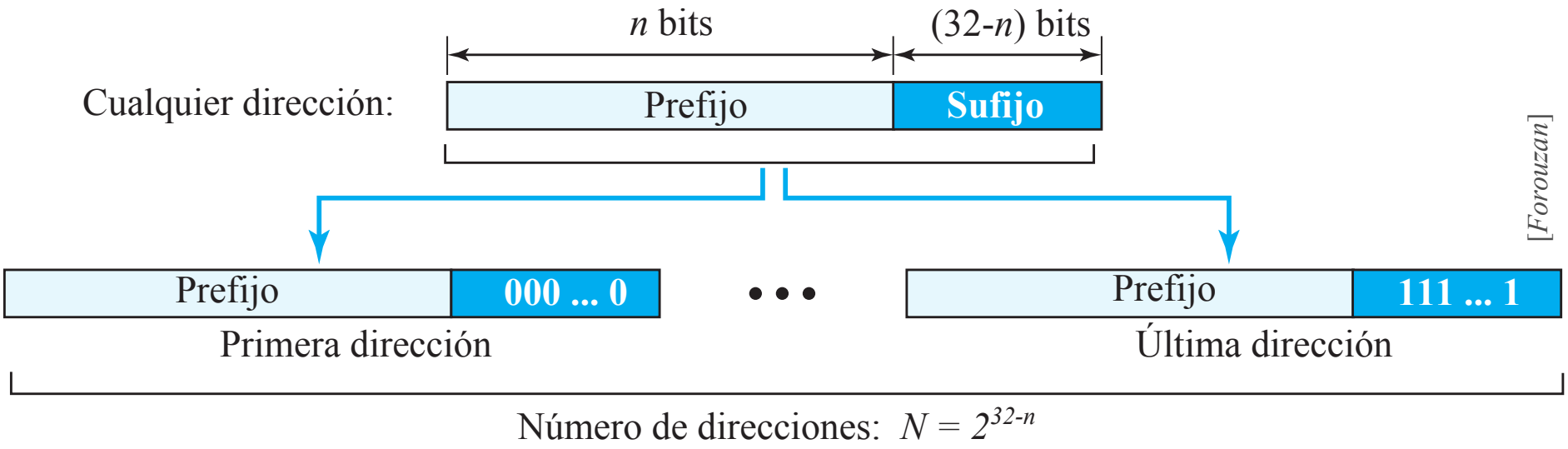
La longitud de prefjo, de longitud n, se añade junto a la dirección mediante una /.

Se denomina notación CIDR (*Classless interdomain routing*).

En direccionamiento *Classless*, una dirección, por sí sola, no define un bloque de direcciones o red; es necesario proporcionar la longitud de prefijo.

La longitud de prefijo viene definida por la **máscara**.

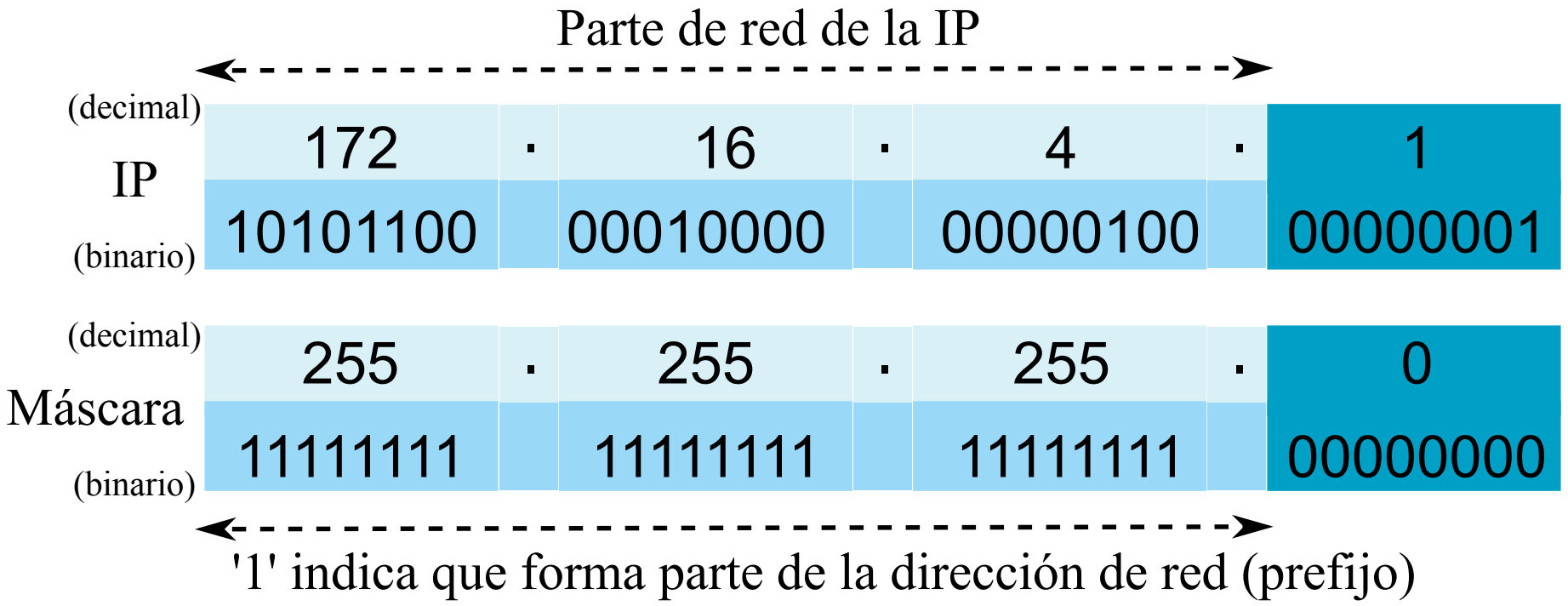
Extracción información en IP *Classless* (I)



Para un determinado bloque de direcciones, definido por la longitud de prefijo n, habitualmente es necesario obtener la siguiente información:

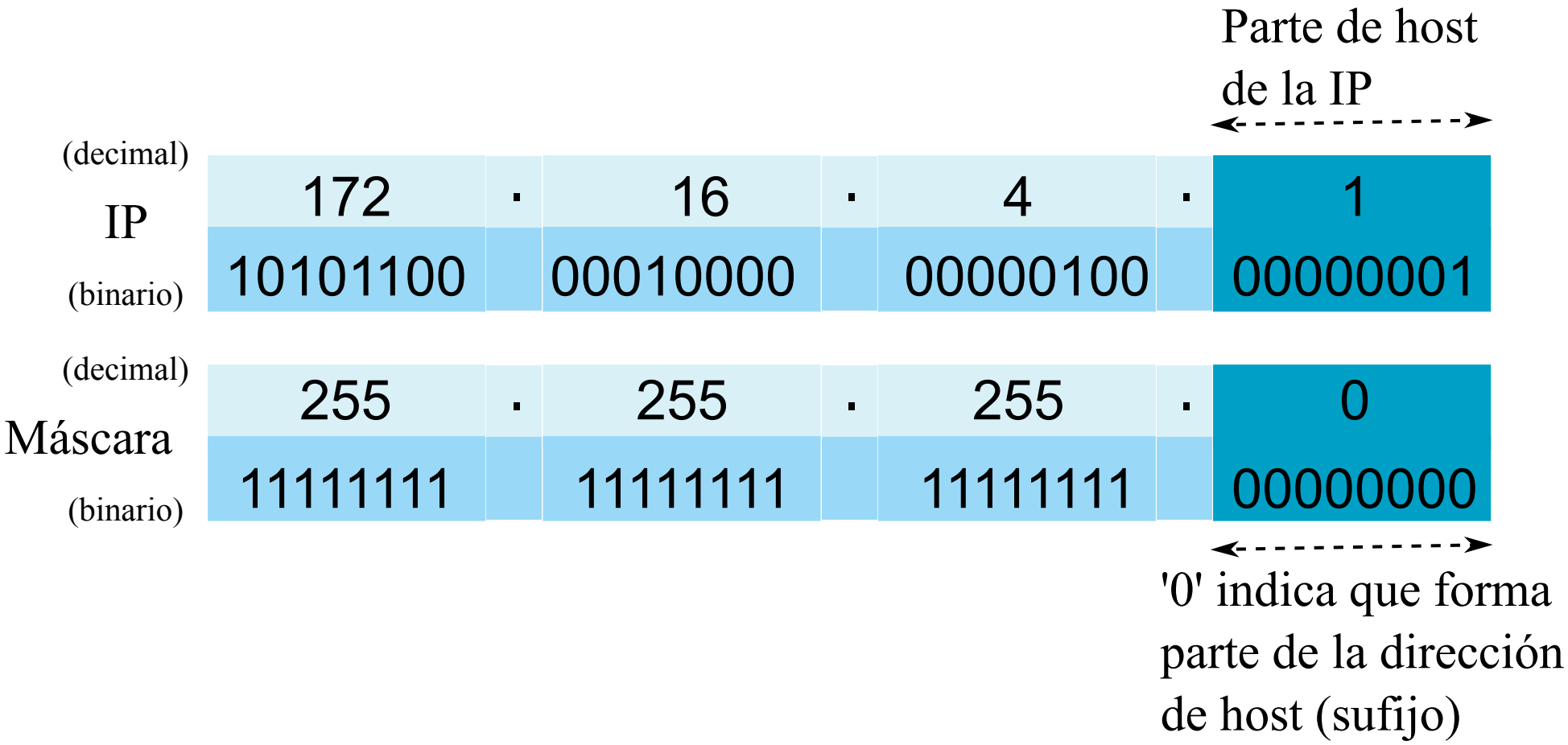
- **Nº de direcciones** del bloque:
$$N = 2^{32-n}$$
- **Primera dirección:** se obtiene manteniendo los n bits de la izquierda y estableciendo los $(32-n)$ bits de la derecha a 0 (**dirección de red**).
- **Última dirección:** se obtiene manteniendo los n bits de la izquierda y estableciendo los $(32-n)$ bits de la derecha a 1 (**dirección de broadcast**).

Dirección IP: bits de red (uso de la máscara de red)



La máscara de red, es una secuencia de 32 bits, de forma que los '1s' identifican aquellos bits que forman parte de la dirección de red.

Dirección IP: bits de host



Los '0s' identifican aquellos bits que forman parte de la dirección de host.

Extracción información en IP *Classless* (II)

Otra forma de encontrar la primera y última dirección en un bloque es mediante operaciones digitales básicas utilizando la máscara.

La máscara de dirección es un número de 32 bits en los que los n bits a la izquierda están a 1, y el resto, $(32-n)$, están a 0.

Un dispositivo IP puede encontrar fácilmente la máscara porque es el complemento de $(2^{32-n}-1)$.

A partir de la definición de la máscara, mediante operaciones digitales (`AND`, `OR` y `NOT`), se extrae la información de bloque:

- Nº de direcciones N= **NOT**(máscara) + 1.
- Primera dirección = (cualquier dirección en el bloque) **AND** (máscara).
- Última dirección = (cualquier dirección en el bloque) **OR** [**NOT** (máscara)].

Dirección de red

¿Cuál es la dirección de red de 192.0.0.1/16?

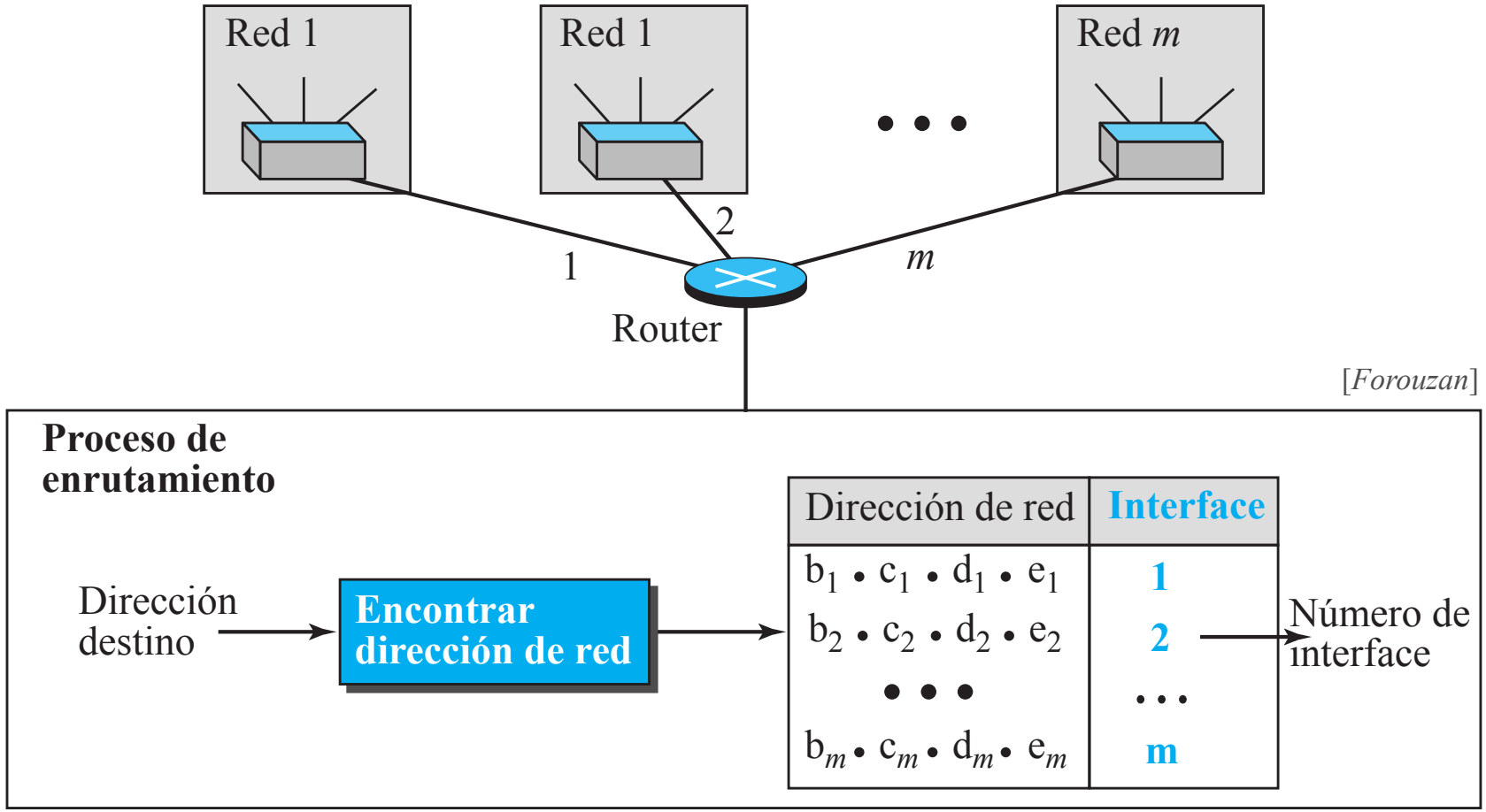
	Bits de orden superior (/16)				Bits de orden inferior			
Dirección de host	192 11000000	.	0 00000000	.	0 00000000	.	1 00000001	
Máscara	255 11111111	.	255 11111111	.	0 00000000	.	0 00000000	
Dirección de red	11000000 192	.	00000000 0	.	00000000 0	.	00000000 0	

Operación AND

	192.16.132.70/20			
Dirección de host	192	16	132	70
Dirección de host (binario)	11000000	00010000	10000100	01000110
AND Máscara de subred (binario)	11111111	11111111	11110000	00000000
Dirección de red (binario)	11000000	00010000	10000000	00000000
Dirección de red	192	16	128	0

La operación AND *bit-a-bit* entre la dirección IP y máscara proporciona la dirección de red de esa dirección IP.

Averiguando la dirección de red



La **dirección de red** es particularmente importante en el proceso de enrutamiento.

Cuando un paquete alcanza un router a través de alguna de sus interfaces, el router necesita averiguar a qué red debe ser enviado.

Por tanto, por cuál de sus interfaces debe ser enviado.

Distribución de bloques

El siguiente aspecto a tener en cuenta en *classless* es la **distibución de bloques**.

En última instancia la ubicación global de direcciones es responsabilidad de ICANN (*Internet Corporation for Assigned Names and Numbers*).

ICANN asigna grandes bloques de direcciones a los diferentes ISPs (*Internet Service Provider*).

Sea N el número de direcciones solicitadas, entonces, para una correcta operación de CIDR, es necesario contemplar dos restricciones en la ubicación de bloques:

1. N, tiene que ser potencia de 2. Como tenemos que:

$$N = 2^{32-n}$$

y, por lo tanto:

$$n = 32 - \log_2 N$$

Si N no es potencia de 2, no podemos obtener un valor entero de n.

2. Los bloques solicitados necesitan ser ubicados de forma que haya la suficiente cantidad de direcciones contiguas disponibles en el espacio de direcciones.

Direccionamiento IP: bits de red y host (I)

172.16.4.0/24		
Dir (binario)	**Dir (decimal)**	**Tipo dir.**
10101100.00010000.00000100. 00000000	172.16.4.0	red
10101100.00010000.00000100. 00000001	172.16.4.1	↑
⋮	⋮	hosts
10101100.00010000.00000100. 11111110	172.16.4.254	↓
10101100.00010000.00000100. 11111111	172.16.4.255	broadcast
$N = 2^{32-24} = 2^8 = 256$	Direcciones útiles = $N - 2 = 254$	

A partir de la dirección 172.16.4.0/24, sabemos que:

- La dirección de red son los 24 primeros bits, y por lo tanto, los 8 bits restantes son la parte hosts.
- Las direcciones útiles (asignables a hosts) son $2^8 - 2 = 254$.

172.16.4.0/25		
Dir (binario)	**Dir (decimal)**	**Tipo dir.**
10101100.00010000.00000100.0 0000000	172.16.4.0	red
10101100.00010000.00000100.0 0000001	172.16.4.1	↑
⋮	⋮	hosts
10101100.00010000.00000100.0 1111110	172.16.4.126	↓
10101100.00010000.00000100.0 1111111	172.16.4.127	broadcast
$N = 2^{32-25} = 2^7 = 128$	Direcciones útiles = $N - 2 = 126$	

A partir de la dirección 172.16.4.0/25, sabemos que:

- La dirección de red son los 25 primeros bits, y por lo tanto, los 7 bits restantes son la parte hosts.
- Las direcciones útiles (asignables a hosts) son $2^7 - 2 = 126$.

Las direcciones *útiles* (asignables a hosts) son las totales, menos dos, correspondientes a la dirección de *red* y *broadcast* (reservadas).

Direccionamiento IP: bits de red y host (II)

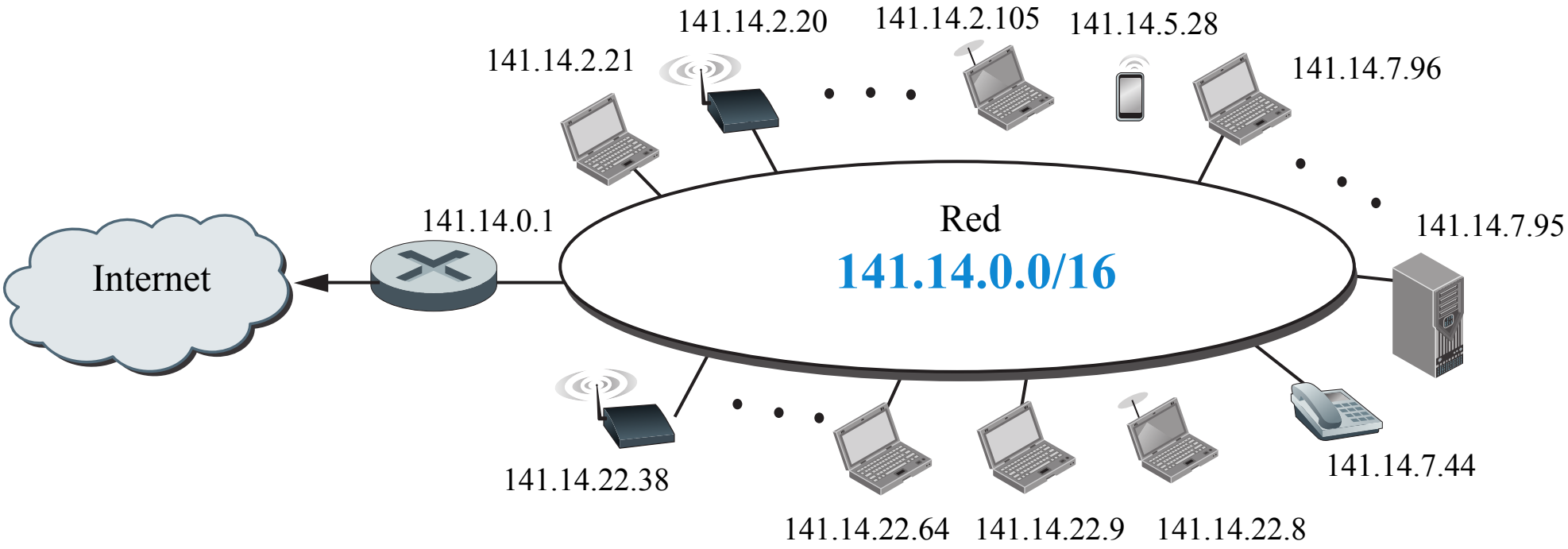
172.16.4.0/26		
Dir (binario)	**Dir (decimal)**	**Tipo dir.**
10101100.00010000.00000100.00 00000	172.16.4.0	red
10101100.00010000.00000100.00 00001	172.16.4.1	↑
⋮	⋮	hosts
10101100.00010000.00000100.00 111110	172.16.4.62	↓
10101100.00010000.00000100.00 111111	172.16.4.63	broadcast
$N = 2^{32-26} = 2^6 = 64$	Direcciones útiles = $N - 2 = 62$	

172.16.4.0/27		
Dir (binario)	**Dir (decimal)**	**Tipo dir.**
10101100.00010000.00000100.000 00000	172.16.4.0	red
10101100.00010000.00000100.000 00001	172.16.4.1	↑
⋮	⋮	hosts
10101100.00010000.00000100.000 11110	172.16.4.30	↓
10101100.00010000.00000100.000 11111	172.16.4.31	broadcast
$N = 2^{32-27} = 2^5 = 32$	Direcciones útiles = $N - 2 = 30$	

Nivel de red

1. Introducción al nivel de red
2. Conmutación de paquetes
3. Direccionamiento IPv4
4. ***Segmentación de redes***
5. Reenvío de paquetes IP
6. Fragmentación
7. Otros protocolos de nivel de red
8. Enrutamiento

Organización de la red: sin subredes



2 niveles de jerarquía (red + host)

Sin el uso de subredes, todos los dispositivos pertenecen a la misma red `141.14.0.0`.

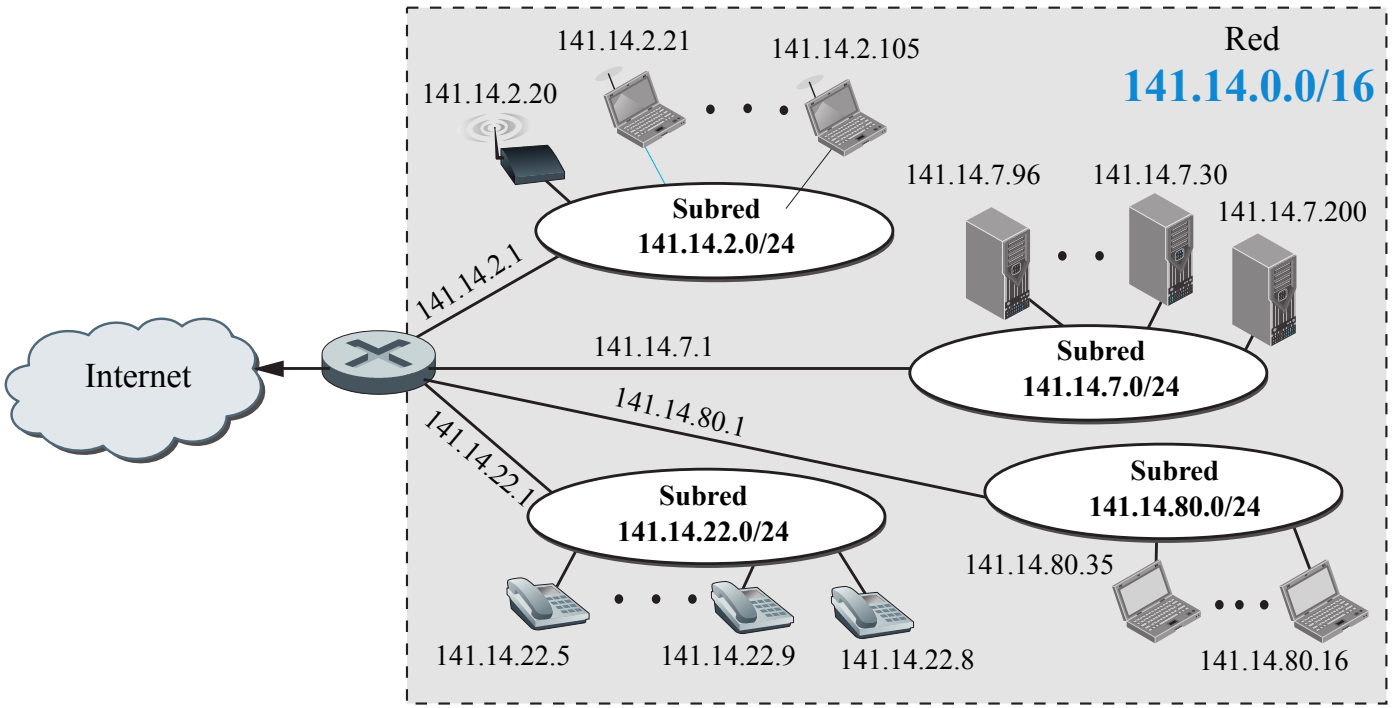
Desde el exterior, los datagramas IP alcanzan la red, y debe de encontrar el dispositivo destino de entre los $2^{16} - 2$ hosts de la red.

Inconvenientes de seguridad, acceso, rendimiento, organización, etc.

Organización de la red: subredes

Tres niveles de jerarquía (red + subred + host)

Los dispositivos se agrupan de forma lógica en subredes aisladas del resto de la red:

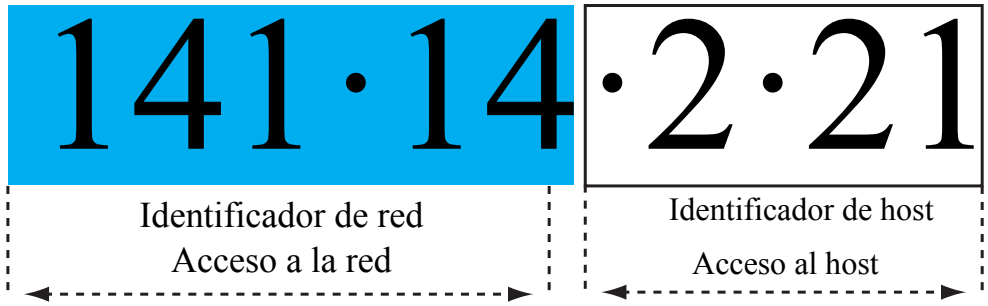


En este caso, cada subred dispone de $2^8 - 2$ hosts.

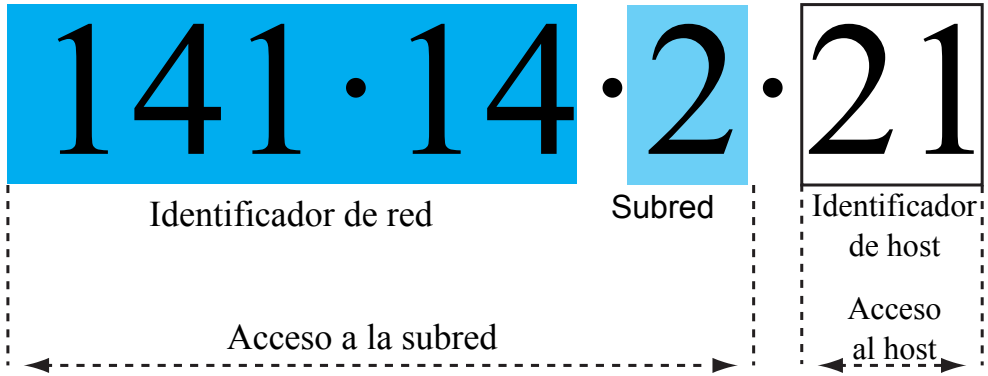
Desde el exterior, la red es conocida como `141.14.0.0`, pero internamente, el router encamina los datagramas a las subredes correspondientes.

Las subredes deben ser, cuidadosamente diseñadas, para habilitar un enrutamiento correcto de paquetes.

Bits prestados para subredes



Direccionamiento sin subredes



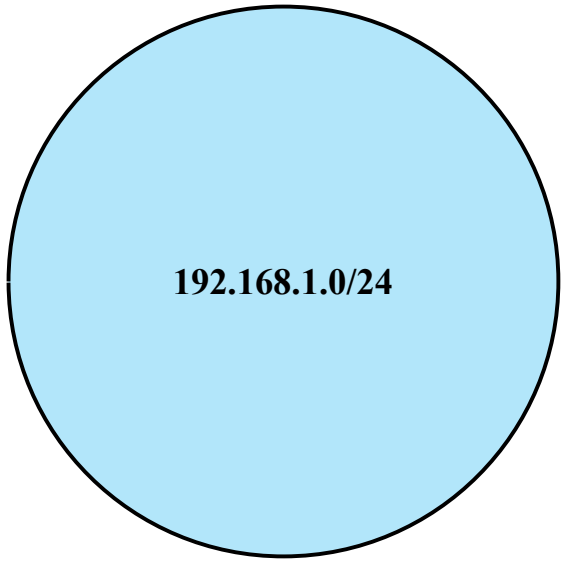
Direccionamiento con subredes

Las subredes se definen a partir de los bits prestados que se toman de la parte de host.

Subnetting

(Longitud fija)

Subnetting: 0 bits⇒ No subnetting



192.168.1.0/24 (0 bits subred → 0 subredes y 2^8 direcciones)				
Subred	**Dir (binario)**	**Dir (decimal)**	**Máscara**	**Tipo**
1	11000000.10101000.00000001.00000000	192.168.1.0	/24	red
$N = 2^{32-24} = 256$	11000000.10101000.00000001.00000001	192.168.1.1	/24	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 256 = 24$	11000000.10101000.00000001.11111110	192.168.1.254	/24	↓
	11000000.10101000.00000001.11111111	192.168.1.255	/24	broadcast

En este caso no se definen subredes.

Subnetting: 1 bit

192.168.1.0/24 (1 bits subred → $2^1 = 2$ subredes y $2^7 = 128$ direcciones/subred)				
Subred	**Dir (binario)**	**Dir (decimal)**	**Máscara**	**Tipo**
1	11000000.10101000.00000001.0 0000000	192.168.1.0	/25	red
$N = 2^{32-25} = 2^7 = 128$	11000000.10101000.00000001.0 0000001	192.168.1.1	/25	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 128 = 25$	11000000.10101000.00000001.0 1111110	192.168.1.126	/25	↓
	11000000.10101000.00000001.0 1111111	192.168.1.127	/25	broadcast
2	11000000.10101000.00000001.1 0000000	192.168.1.128	/25	red
$N = 2^{32-25} = 2^7 = 128$	11000000.10101000.00000001.1 0000001	192.168.1.129	/25	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 128 = 25$	11000000.10101000.00000001.1 1111110	192.168.1.254	/25	↓
	11000000.10101000.00000001.1 1111111	192.168.1.255	/25	broadcast

Si tomamos un bit de host para definir subredes ⇒ podemos definir $2^1 = 2$ subredes ⇒ con los 7 bits restantes de host, se pueden definir $2^7 = 128$ direcciones cada subred.

El espacio de direcciones útiles es $2^7 - 2 = 126$.

Subnetting: 2 bits

192.168.1.0/24 (2 bits subred → $2^2 = 4$ subredes y $2^6 = 64$ direcciones/subred)				
Subred	**Dir (binario)**	**Dir (decimal)**	**Máscara**	**Tipo**
1 $N = 2^{32-26} = 2^6 = 64$ $n = 32 - \log_2 64 = 26$	11000000.10101000.00000001.00 000000	192.168.1.0	/26	red
	11000000.10101000.00000001.00 000001	192.168.1.1	/26	↑ hosts
	⋮	⋮	⋮	
	11000000.10101000.00000001.00 111110	192.168.1.62	/26	↓
	11000000.10101000.00000001.00 111111	192.168.1.63	/26	broadcast
2 $N = 2^{32-26} = 2^6 = 64$ $n = 32 - \log_2 64 = 26$	11000000.10101000.00000001.01 000000	192.168.1.64	/26	red
	11000000.10101000.00000001.01 000001	192.168.1.65	/26	↑ hosts
	⋮	⋮	⋮	
	11000000.10101000.00000001.01 111110	192.168.1.126	/26	↓
	11000000.10101000.00000001.01 111111	192.168.1.127	/26	broadcast
3 $N = 2^{32-26} = 2^6 = 64$ $n = 32 - \log_2 64 = 26$	11000000.10101000.00000001.10 000000	192.168.1.128	/26	red
	11000000.10101000.00000001.10 000001	192.168.1.129	/26	↑ hosts
	⋮	⋮	⋮	
	11000000.10101000.00000001.10 111110	192.168.1.190	/26	↓
	11000000.10101000.00000001.10 111111	192.168.1.191	/26	broadcast
4 $N = 2^{32-26} = 2^6 = 64$ $n = 32 - \log_2 64 = 26$	11000000.10101000.00000001.11 000000	192.168.1.192	/26	red
	11000000.10101000.00000001.11 000001	192.168.1.193	/26	↑ hosts
	⋮	⋮	⋮	
	11000000.10101000.00000001.11 111110	192.168.1.254	/26	↓
	11000000.10101000.00000001.11 111111	192.168.1.255	/26	broadcast

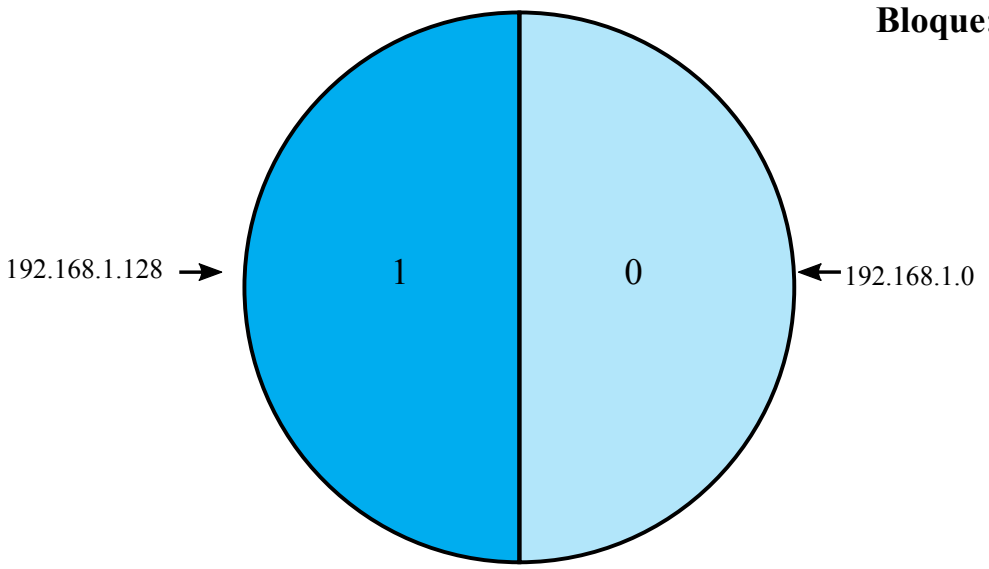
Subnetting: 3 bits

192.168.1.0/24 (3 bits subred → $2^3 = 8$ subredes y $2^5 = 32$ direcciones/subred)				
Subred	**Dir (binario)**	**Dir (decimal)**	**Máscara**	**Tipo**
1 $N = 2^{32-27} = 2^5 = 32$ $n = 32 - \log_2 32 = 27$	11000000.10101000.00000001.000 00000 11000000.10101000.00000001.000 00001 ⋮ 11000000.10101000.00000001.000 11110 11000000.10101000.00000001.000 11111	192.168.1.0 192.168.1.1 ⋮ 192.168.1.30 192.168.1.31	/27 /27 ⋮ /27 /27	red ↑ hosts ↓ broadcast
2 $N = 2^{32-27} = 2^5 = 32$ $n = 32 - \log_2 32 = 27$	11000000.10101000.00000001.001 00000 11000000.10101000.00000001.001 00001 ⋮ 11000000.10101000.00000001.001 11110 11000000.10101000.00000001.001 11111	192.168.1.32 192.168.1.33 ⋮ 192.168.1.62 192.168.1.63	/27 /27 ⋮ /27 /27	red ↑ hosts ↓ broadcast
3	...	...	...	...
4	...	...	...	...
5	...	...	...	...
6	...	...	...	...
7	...	...	...	...
8 $N = 2^{32-27} = 2^5 = 32$ $n = 32 - \log_2 32 = 27$	11000000.10101000.00000001.111 00000 11000000.10101000.00000001.111 00001 ⋮ 11000000.10101000.00000001.111 11110 11000000.10101000.00000001.111 11111	192.168.1.224 192.168.1.225 ⋮ 192.168.1.254 192.168.1.255	/27 /27 ⋮ /27 /27	red ↑ hosts ↓ broadcast

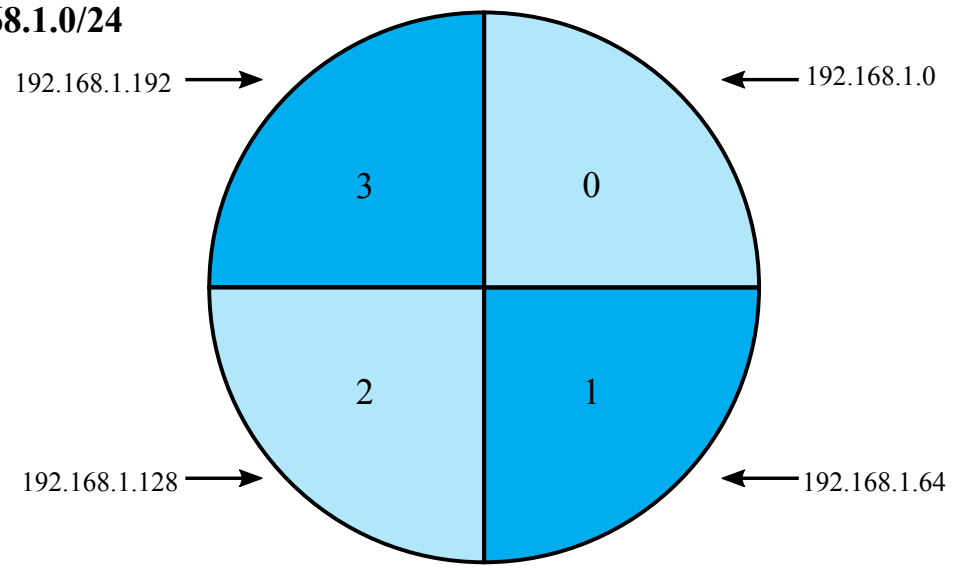
Subnetting: 4 bits

192.168.1.0/24 (4 bits subred → $2^4 = 16$ subredes y $2^4 = 16$ direcciones/subred)				
Subred	**Dir (binario)**	**Dir (decimal)**	**Máscara**	**Tipo**
1 $N = 2^{32-28} = 2^4 = 16$ $n = 32 - \log_2 16 = 28$	11000000.10101000.00000001.0000 0000 11000000.10101000.00000001.0000 0001 ⋮ 11000000.10101000.00000001.0000 1110 11000000.10101000.00000001.0000 1111	192.168.1.0 192.168.1.1 ⋮ 192.168.1.14 192.168.1.15	/28 /28 ⋮ /28 /28	red ↑ hosts ↓ broadcast
2	...	...	...	...
3	...	...	...	...
4	...	...	...	...
5	...	...	...	...
6	...	...	...	...
7	...	...	...	...
8	...	...	...	...
9	...	...	...	...
10	...	...	...	...
11	...	...	...	...
12	...	...	...	...
13	...	...	...	...
14	...	...	...	...
15	...	...	...	...
16 $N = 2^{32-28} = 2^4 = 16$ $n = 32 - \log_2 16 = 28$	11000000.10101000.00000001.1111 0000 11000000.10101000.00000001.1111 0001 ⋮ 11000000.10101000.00000001.1111 1110 11000000.10101000.00000001.1111 1111	192.168.1.240 192.168.1.241 ⋮ 192.168.1.254 192.168.1.255	/28 /28 ⋮ /28 /28	red ↑ hosts ↓ broadcast

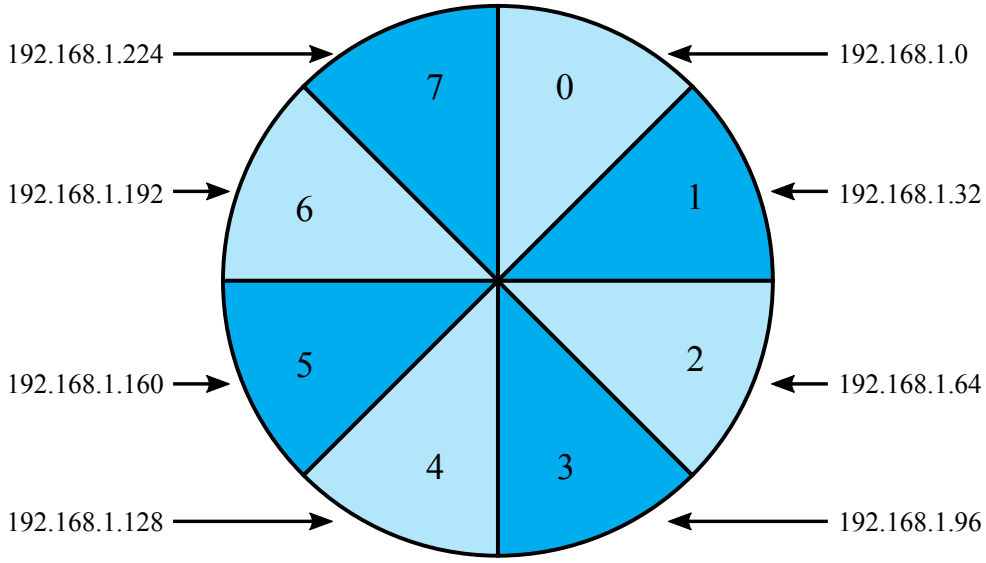
Subnetting tradicional: bloques de mismo tamaño



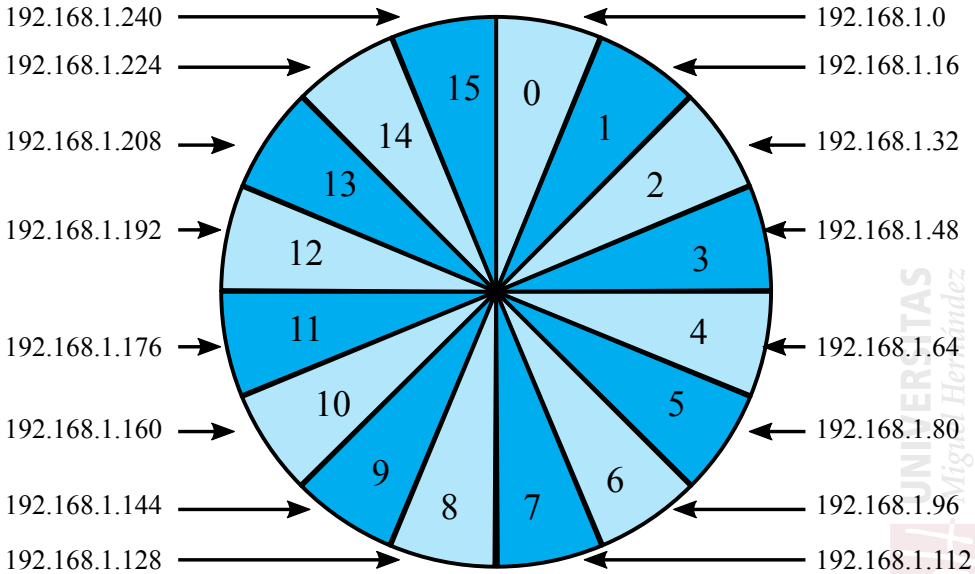
a) Subnetting: 1 bit → /25



b) Subnetting: 2 bits → /26



c) Subnetting: 3 bits → /27



d) Subnetting: 4 bits → /28

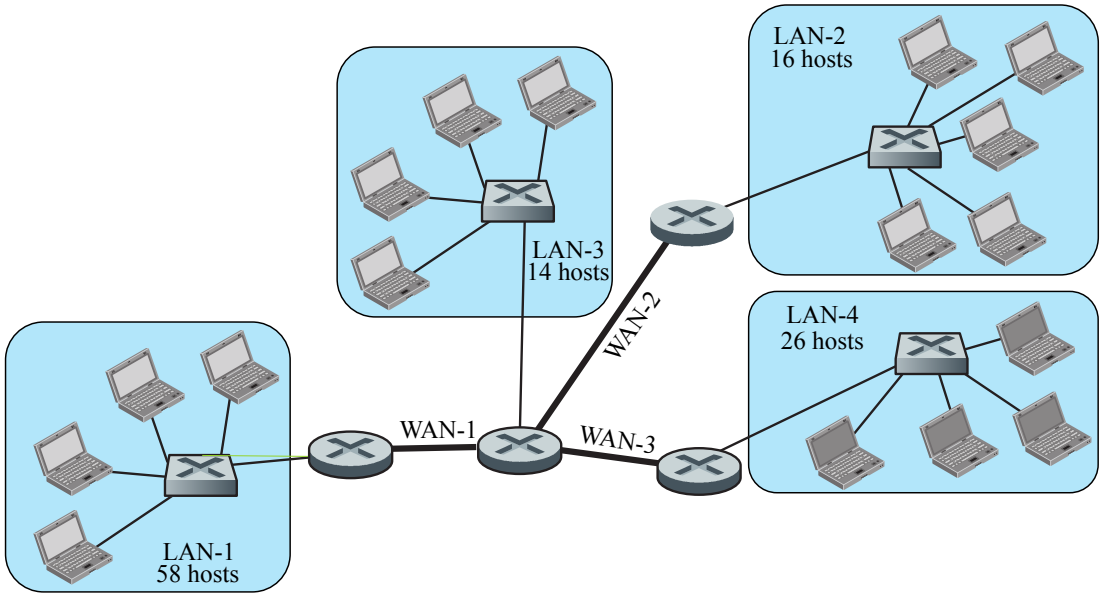
Subnetting: escenario A

Direccionar la siguiente topología:

Bloque: `125.130.172.0/23`

LANs: la puerta de salida (*gateway*) está incluída en los hosts.

WANs: enlaces punto a punto



¿Cuántos bits hay disponibles para subnetting?

¿Cuántas redes hay que direccionar en la topología?

¿Cuántos bits se necesitan '*tomar prestados*' para hacer subnetting?

¿Cuántos bits de hosts quedan disponibles?

¿Cuántos hosts por subred se pueden direccionar?

Por lo tanto, ¿es posible realizar el subnetting?

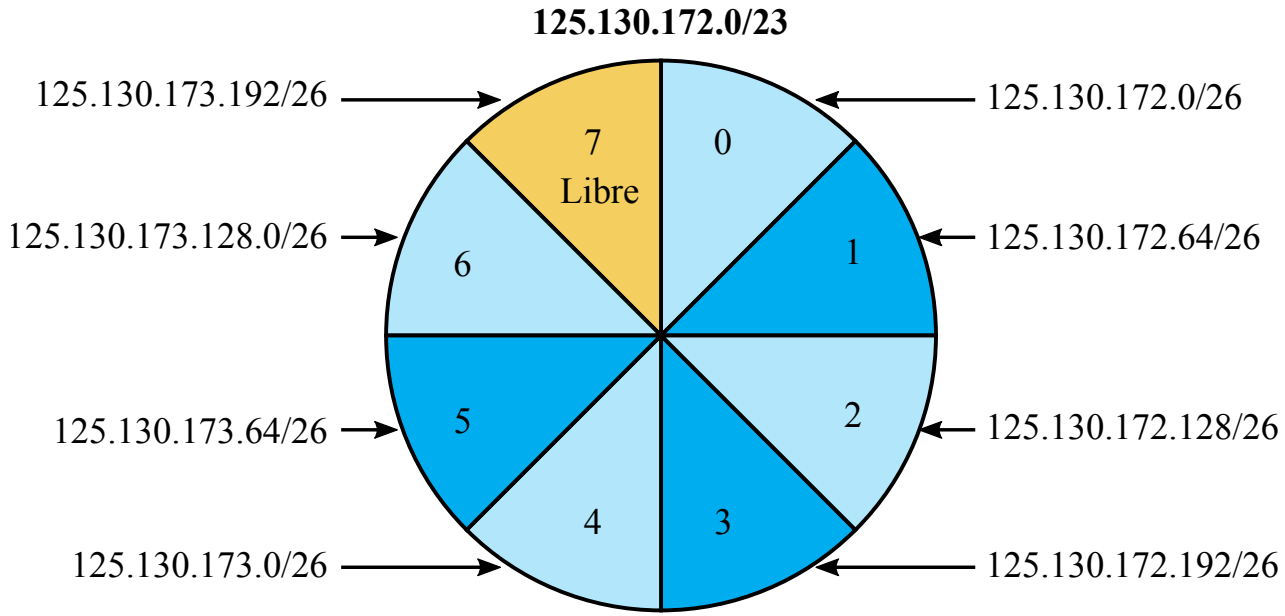
Subnetting tradicional (I)

125.130.172.0/23 (7 subredes → 3 bits de subred)				
Red	**Binario**	**Decimal**	**Máscara**	**Tipo**
[LAN-1: 58 hosts]	125.130.10101100.00 000000	125.130.172.0	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101100.00 000001	125.130.172.1	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101100.00 111110	125.130.172.62	/26	↓
	125.130.10101100.00 111111	125.130.172.63	/26	broadcast
[LAN-2: 16 hosts]	125.130.10101100.01 000000	125.130.172.64	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101100.01 000001	125.130.172.65	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101100.01 111110	125.130.172.126	/26	↓
	125.130.10101100.01 111111	125.130.172.127	/26	broadcast
[LAN-3: 14 hosts]	125.130.10101100.10 000000	125.130.172.128	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101100.10 000001	125.130.172.129	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101100.10 111110	125.130.172.190	/26	↓
	125.130.10101100.10 111111	125.130.172.191	/26	broadcast
[LAN-4: 26 hosts]	125.130.10101100.11 000000	125.130.172.192	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101100.11 000001	125.130.172.193	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101100.11 111110	125.130.172.254	/26	↓
	125.130.10101100.11 111111	125.130.172.255	/26	broadcast

Subnetting tradicional (II)

125.130.172.0/23 (7 subredes → 3 bits de subred)				
Red	**Binario**	**Decimal**	**Máscara**	**Tipo**
[WAN-1: 2 hosts]	125.130.10101101.00 000000	125.130.173.0	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101101.00 000001	125.130.173.1	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101101.00 111110	125.130.173.62	/26	↓
	125.130.10101101.00 111111	125.130.173.63	/26	host
[WAN-2: 2 hosts]	125.130.10101101.01 000000	125.130.173.64	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101101.01 000001	125.130.173.65	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101101.01 111110	125.130.173.126	/26	↓
	125.130.10101101.01 111111	125.130.173.127	/26	broadcast
[WAN-3: 2 hosts]	125.130.10101101.10 000000	125.130.173.128	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101101.10 000001	125.130.173.129	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101101.10 111110	125.130.173.190	/26	↓
	125.130.10101101.10 111111	125.130.173.191	/26	host
Libre	125.130.10101101.11 000000	125.130.173.192	/26	red
$N = 2^{32-26} = 2^6 = 64$	125.130.10101101.11 000001	125.130.173.193	/26	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 64 = 26$	125.130.10101101.11 111110	125.130.173.254	/26	↓
	125.130.10101101.11 111111	125.130.173.255	/26	broadcast

Subnetting (longitud fija)



Red	Requisitos	Útiles	No utilizadas
LAN-1	58	62	4
LAN-2	16	62	46
LAN-3	14	62	48
LAN-4	26	62	36
WAN-1	2	62	60
WAN-2	2	62	60
WAN-3	2	62	60

Con subnetting tradicional y al tratarse de subredes de tamaño fijo, provoca una elevada pérdida de direcciones IP, que no pueden ser utilizadas en otras redes.

Subnetting
(Longitud variable)

Subnetting: escenario B

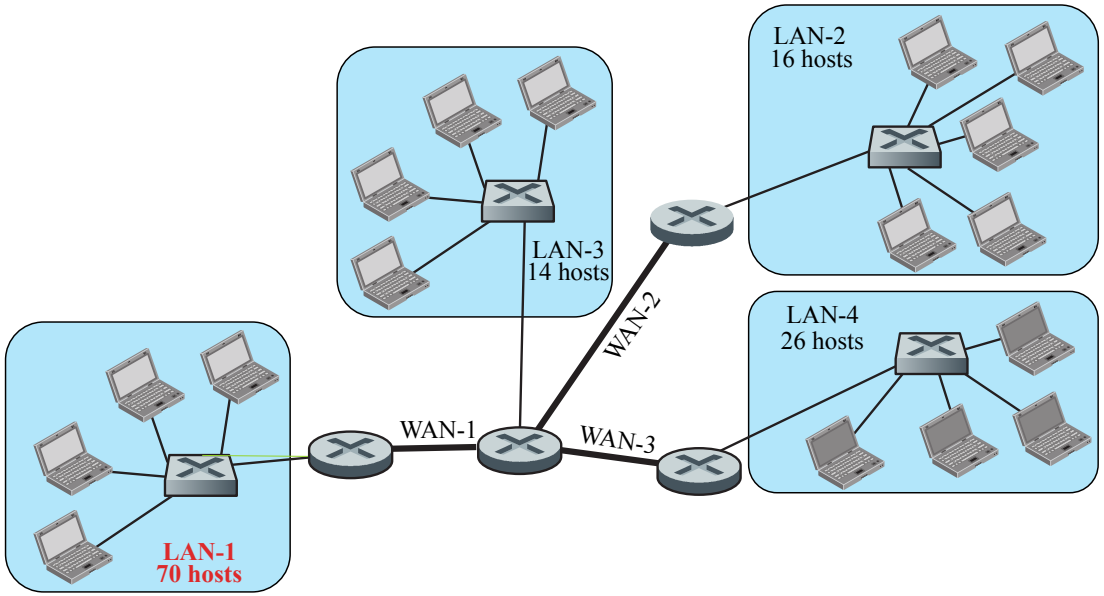
Direccionar la siguiente topología:

Bloque: `125.130.172.0/23`

LANs: la puerta de salida (*gateway*) está incluída en los hosts.

WANs: enlaces punto a punto

Han aumentado los requisitos de hosts en LAN-1



¿Cuántos bits hay disponibles para subnetting?

¿Cuántas redes hay que direccionar en la topología?

¿Cuántos bits se necesitan '*tomar prestados*' para hacer subnetting?

¿Cuántos bits de hosts quedan disponibles?

¿Cuántos hosts por subred se pueden direccionar?

Por lo tanto, ¿es posible realizar el subnetting?

Subredes de longitud variable

El subnetting de longitud variable se realiza mediante el denominado **VLSM** (*Variable Length Subnet Mask*):

Si el total de direcciones que dispone una organización es N y su longitud de prefijo es n, el número de direcciones utilizadas para subredes es N_{sub} y el prefijo es n_{sub}, atendiendo a las siguientes restricciones:

1. El número de direcciones en cada subred debe ser potencia de 2.
2. La longitud de prefijo en cada subred se calcula según la siguiente expresión:

$$n_{sub} = 32 - \log_2 N_{sub}$$

Es imprescindible realizar el proceso de subnetting siempre de mayor a menor número de direcciones necesarias en cada subred.

Con el uso de VLSM se obtiene un direccionamiento más optimizado, compacto, con redes más ajustadas y conjunto de direcciones libres al final.

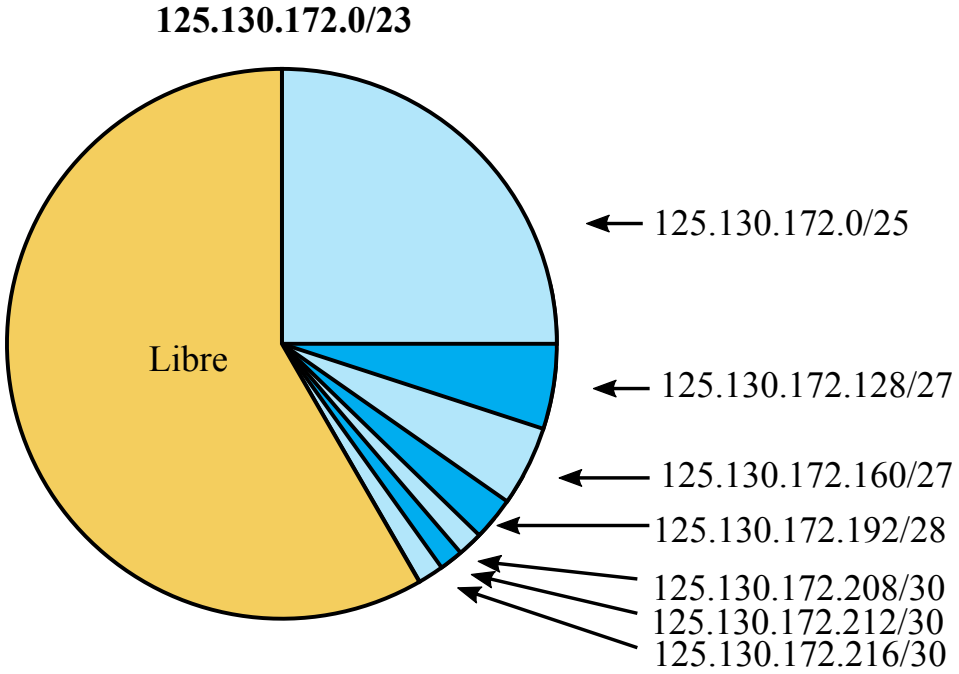
VLSM: direccionamiento (I)

125.130.172.0/23				
Red	**Binario**	**Decimal**	**Máscara**	**Tipo**
[LAN-1: 70 hosts]	125.130.10101100.0 0000000	125.130.172.0	/25	red
$N = 2^{32-25} = 2^7 = 128$	125.130.10101100.0 0000001	125.130.172.1	/25	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 128 = 25$	125.130.10101100.0 1111110	125.130.172.126	/25	↓
	125.130.10101100.0 1111111	125.130.172.127	/25	broadcast
[LAN-4: 26 hosts]	125.130.10101100.100 00000	125.130.172.128	/27	red
$N = 2^{32-27} = 2^5 = 32$	125.130.10101100.100 00001	125.130.172.129	/27	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 32 = 27$	125.130.10101100.100 11110	125.130.172.158	/27	↓
	125.130.10101100.100 11111	125.130.172.159	/27	broadcast
[LAN-2: 16 hosts]	125.130.10101100.101 00000	125.130.172.160	/27	red
$N = 2^{32-27} = 2^5 = 32$	125.130.10101100.101 00001	125.130.172.161	/27	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 32 = 27$	125.130.10101100.101 11110	125.130.172.190	/27	↓
	125.130.10101100.101 11111	125.130.172.191	/27	broadcast
[LAN-3: 14 hosts]	125.130.10101100.1100 0000	125.130.172.192	/28	red
$N = 2^{32-28} = 2^4 = 16$	125.130.10101100.1100 0001	125.130.172.193	/28	↑
	⋮	⋮	⋮	hosts
$n = 32 - \log_2 16 = 28$	125.130.10101100.1100 1110	125.130.172.206	/28	↓
	125.130.10101100.1100 1111	125.130.172.207	/28	broadcast

VLSM: direccionamiento (II)

125.130.172.0/23				
Red	**Binario**	**Decimal**	**Máscara**	**Tipo**
[WAN-1: 2 hosts]	125.130.10101100.110100 00	125.130.172.208	/30	red
$N = 2^{32-30} = 2^2 = 4$	125.130.10101100.110100 01	125.130.172.209	/30	host
	125.130.10101100.110100 10	125.130.172.210	/30	host
$n = 32 - \log_2 4 = 30$	125.130.10101100.110100 11	125.130.172.211	/30	broadcast
[WAN-2: 2 hosts]	125.130.10101100.110101 00	125.130.172.212	/30	red
$N = 2^{32-30} = 2^2 = 4$	125.130.10101100.110101 01	125.130.172.213	/30	host
	125.130.10101100.110101 10	125.130.172.214	/30	host
$n = 32 - \log_2 4 = 30$	125.130.10101100.110101 11	125.130.172.215	/30	broadcast
[WAN-3: 2 hosts]	125.130.10101100.110110 00	125.130.172.216	/30	red
$N = 2^{32-30} = 2^2 = 4$	125.130.10101100.110110 01	125.130.172.217	/30	host
	125.130.10101100.110110 10	125.130.172.218	/30	host
$n = 32 - \log_2 4 = 30$	125.130.10101100.110110 11	125.130.172.219	/30	broadcast
Libre	125.130.10101100.11011100 ⋮ 125.130.10101101.11111111	125.130.172.220 ⋮ 125.130.173.255		

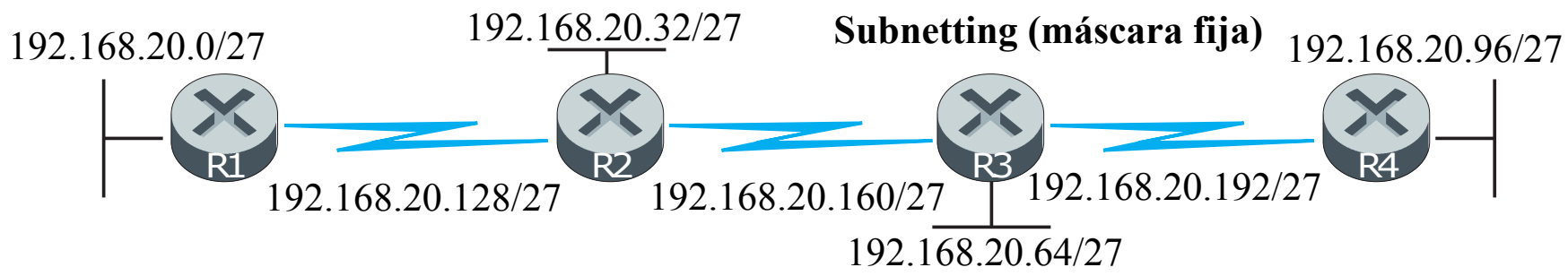
VLSM



Red	Requisitos	Útiles	No utilizadas
LAN-1	70	126	56
LAN-2	16	30	14
LAN-3	14	16	2
LAN-4	26	30	4
WAN-1	2	2	0
WAN-2	2	2	0
WAN-3	2	2	0

Mediante VLSM se obtiene una optimización en el uso de direccionamiento IP al conseguir menores pérdidas.

Aplicación de VLSM

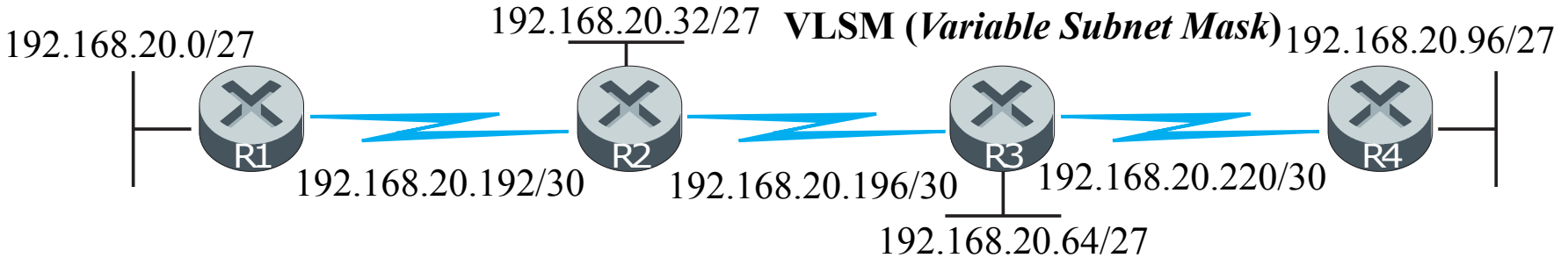


192.168.20.0/24

Se vuelve hacer subnetting sobre 192.168.20.192/27 obteniendo subredes de longitud fija /30

Subred	Dirección
1	192.168.20.0/27
2	192.168.20.32/27
3	192.168.20.64/27
4	192.168.20.96/27
5	192.168.20.128/27
6	192.168.20.160/27
7	192.168.20.192/27
8	192.168.20.224/27

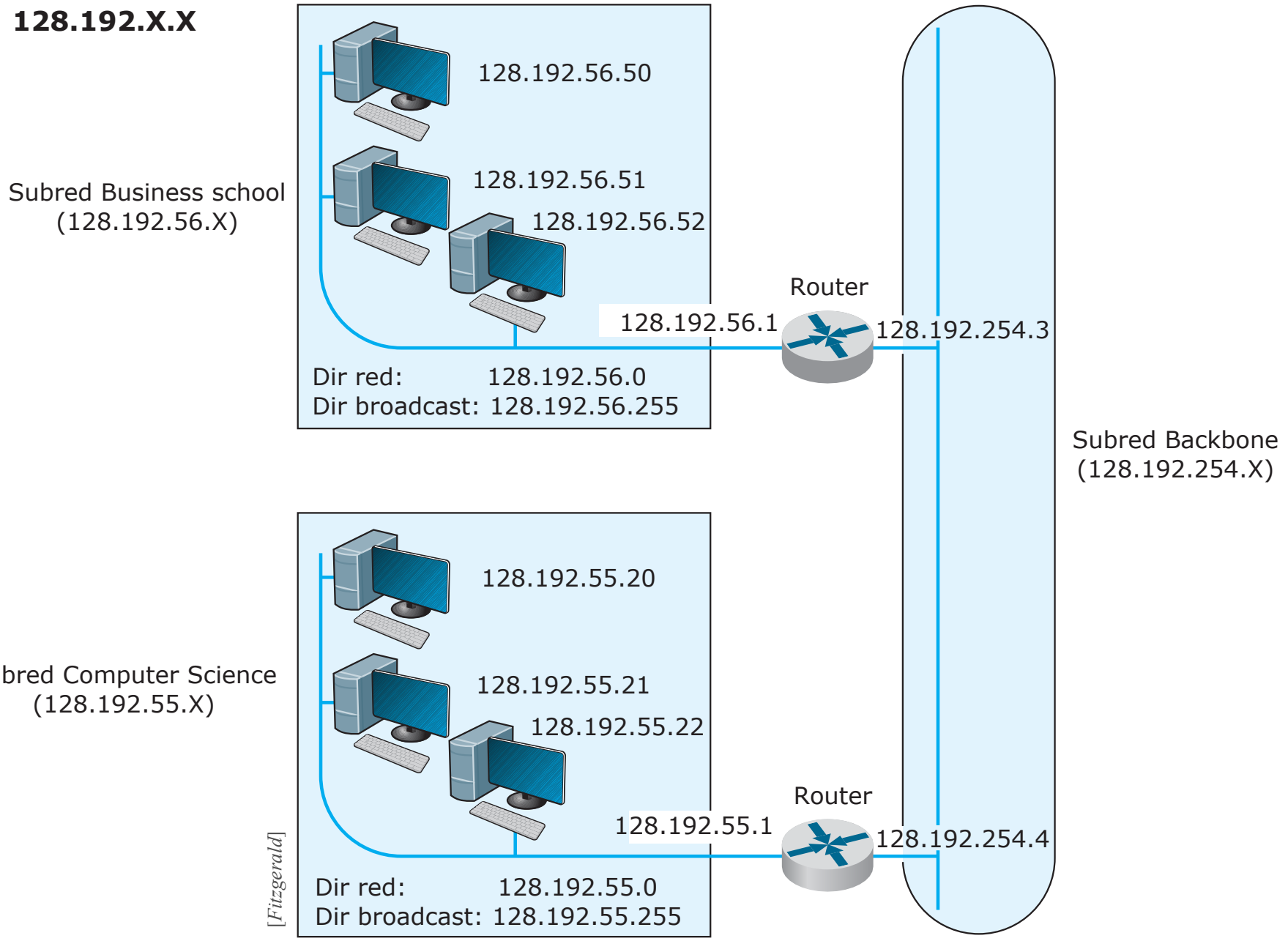
Subred	Dirección
1	192.168.20.192/30
2	192.168.20.196/30
3	192.168.20.200/30
4	192.168.20.204/30
5	192.168.20.208/30
6	192.168.20.212/30
7	192.168.20.216/30
8	192.168.20.220/30



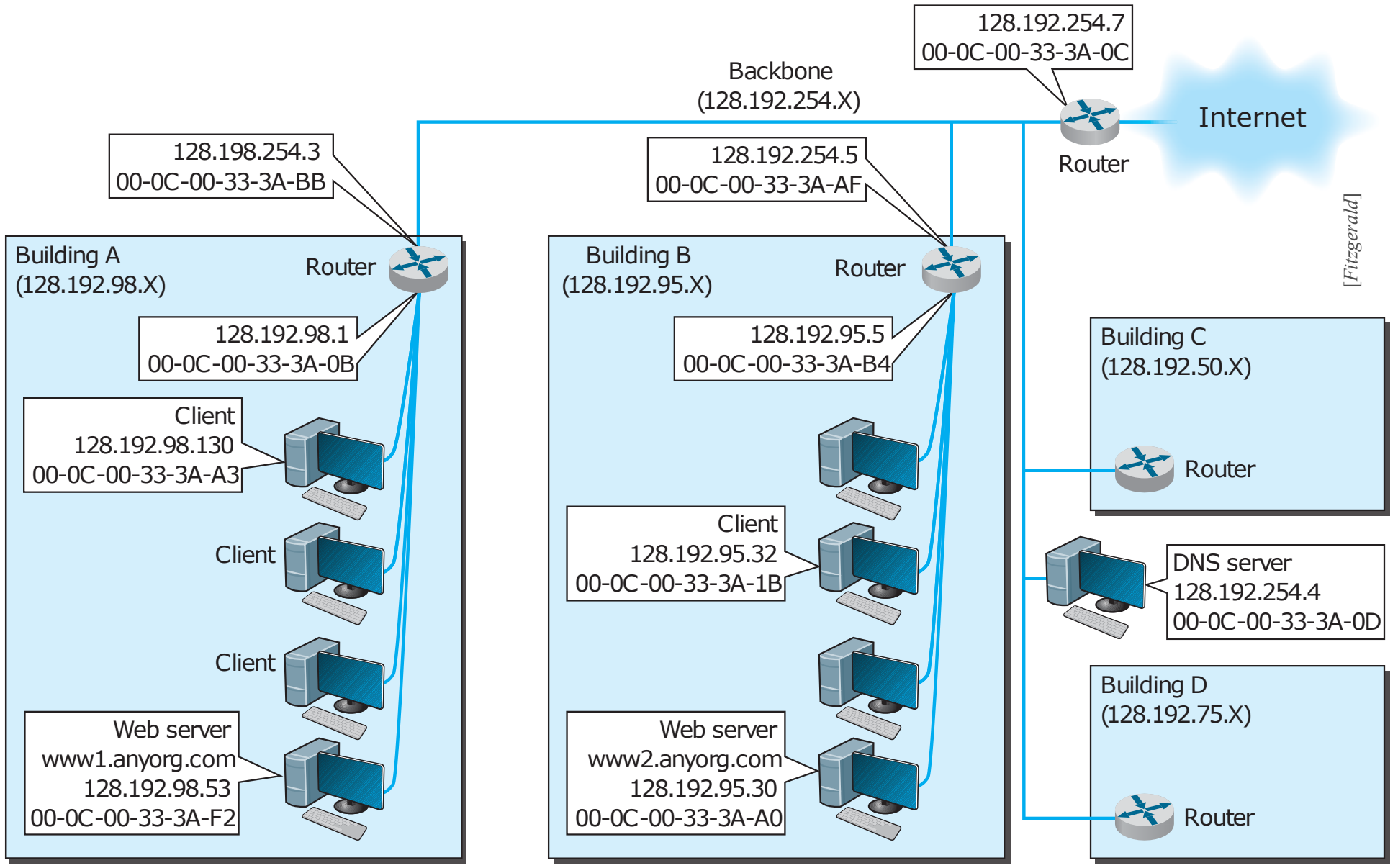
Subnetting chart

Máscara (decimal)	Binario	CIDR	Redes	Hosts
255.128.0.0	11111111.10000000.00000000.00000000	/9	2	8388606
255.192.0.0	11111111.11000000.00000000.00000000	/10	4	4194302
255.224.0.0	11111111.11100000.00000000.00000000	/11	8	2097150
255.240.0.0	11111111.11110000.00000000.00000000	/12	16	1048574
255.248.0.0	11111111.11111000.00000000.00000000	/13	32	524286
255.252.0.0	11111111.11111100.00000000.00000000	/14	64	262142
255.254.0.0	11111111.11111110.00000000.00000000	/15	128	131070
255.255.0.0	11111111.11111111.00000000.00000000	/16	256	65534
255.255.128.0	11111111.11111111.10000000.00000000	/17	512	32766
255.255.192.0	11111111.11111111.11000000.00000000	/18	1024	16382
255.255.224.0	11111111.11111111.11100000.00000000	/19	2048	8190
255.255.240.0	11111111.11111111.11110000.00000000	/20	4096	4094
255.255.248.0	11111111.11111111.11111000.00000000	/21	8192	2046
255.255.252.0	11111111.11111111.11111100.00000000	/22	16384	1022
255.255.254.0	11111111.11111111.11111110.00000000	/23	32768	510
255.255.255.0	11111111.11111111.11111111.00000000	/24	65536	254
255.255.255.128	11111111.11111111.11111111.10000000	/25	131072	126
255.255.255.192	11111111.11111111.11111111.11000000	/26	262144	62
255.255.255.224	11111111.11111111.11111111.11100000	/27	524288	30
255.255.255.240	11111111.11111111.11111111.11110000	/28	1048576	14
255.255.255.248	11111111.11111111.11111111.11111000	/29	2097152	6
255.255.255.252	11111111.11111111.11111111.11111100	/30	4194304	2
255.255.255.254	11111111.11111111.11111111.11111110	/31	8388608	0
255.255.255.255	11111111.11111111.11111111.11111111	/32	*Broadcast*	

Ejemplo (I)


128.192.X.X
Subred Business school
(128.192.56.X)
128.192.56.50
128.192.56.51
128.192.56.52
Router
128.192.56.1
128.192.254.3
Dir red: 128.192.56.0
Dir broadcast: 128.192.56.255
Subred Backbone
(128.192.254.X)
Subred Computer Science
(128.192.55.X)
128.192.55.20
128.192.55.21
128.192.55.22
Router
128.192.55.1
128.192.254.4
Dir red: 128.192.55.0
Dir broadcast: 128.192.55.255
[Fitzgerald]

Ejemplo (II)


128.192.254.7
00-0C-00-33-3A-0C
Backbone
(128.192.254.X)
Internet
Router
128.198.254.3
00-0C-00-33-3A-BB
128.192.254.5
00-0C-00-33-3A-AF
[Fitzgerald]
Building A
(128.192.98.X)
Router
128.192.98.1
00-0C-00-33-3A-0B
Client
128.192.98.130
00-0C-00-33-3A-A3
Client
Client
Web server
www1.anyorg.com
128.192.98.53
00-0C-00-33-3A-F2
Building B
(128.192.95.X)
Router
128.192.95.5
00-0C-00-33-3A-B4
Client
128.192.95.32
00-0C-00-33-3A-1B
Web server
www2.anyorg.com
128.192.95.30
00-0C-00-33-3A-A0
Building C
(128.192.50.X)
Router
DNS server
128.192.254.4
00-0C-00-33-3A-0D
Building D
(128.192.75.X)
Router

Agregación de direcciones

Agregación de direcciones

Una de las ventajas de la estrategia CIDR es la **agregación de direcciones**.

También denominada **sumarización de direcciones/rutas** o **supernetting**.

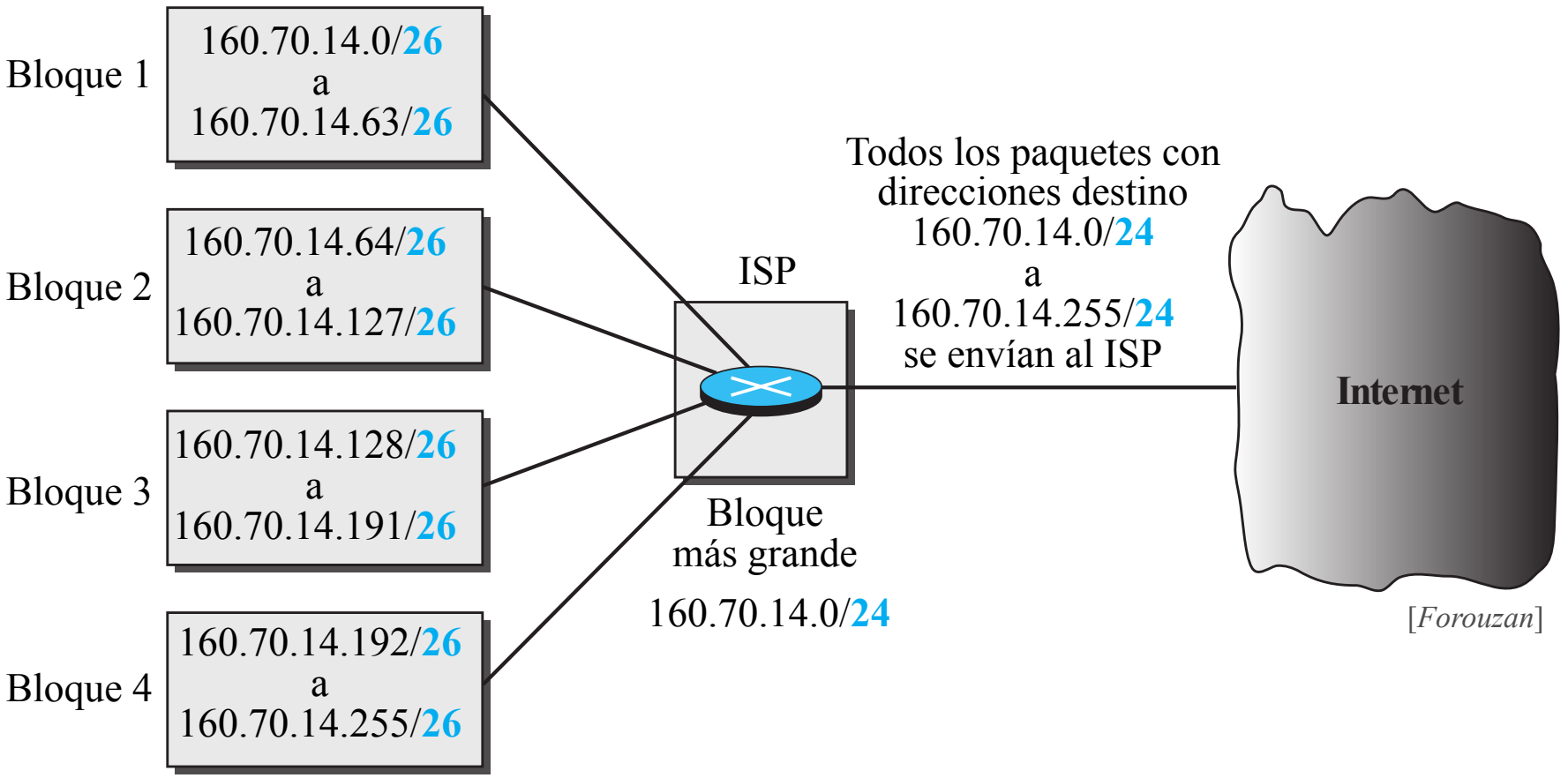
Consiste en la combinación de múltiples bloques de direcciones en otro de mayor tamaño.

Permite que el enrutamiento se realice según el prefijo de mayor tamaño.

Combina varias entradas en la tabla de enrutamiento, reduciendo el tamaño de las tablas y eliminando procesamiento.

ICANN asigna grandes bloques de direcciones a los ISPs. Cada ISP divide sus bloques asignados en subbloques más pequeños y los asigna a sus clientes.

Agregación de direcciones en ISP



[*Forouzan*]

Direcciones especiales

Direcciones especiales: *reservadas*

Dir. de red: parte de la dirección de red compartidos por todos los hosts de la red. Parte de host a '0'.

Dir. de broadcast: dirección que identifica todos los nodos de la red. Parte de host a '1'.

`0.0.0.0/8`**:** utilizada por BOOTP y DHCP.

`127.0.0.0/8`**:** dirección de loopback.

Link-local (`169.254.0.0/16`**):** utilizadas para pruebas.

TEST-NET (`192.0.2.0/24`**):** utilizadas para pruebas.

Las **direcciones reservadas** son aquellas IPs que no pueden ser utilizadas como direccionamiento de host.

Direcciones especiales: *privadas*

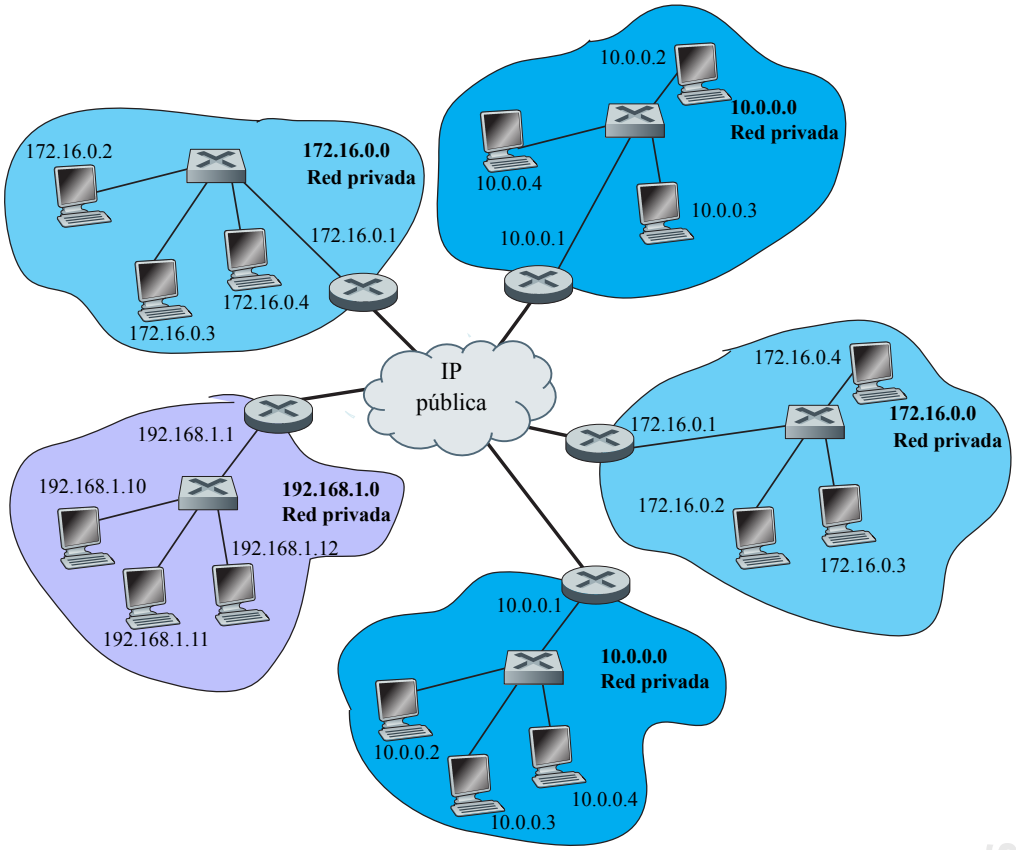
Uso privado en organizaciones.

Se establecieron ante la falta de direcciones IP.

Necesario protocolo NAT (*Network Address Translation*).

Rangos:

```
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
169.254.0.0/16
```



Los routers NO deben encaminarlas.

Las **direcciones privadas** son aquellas que pueden ser utilizadas internamente en las organizaciones.

Es necesario que los routers gateway soporten NAT para la traducción de IP privadas en públicas.

Nivel de red

1. Introducción al nivel de red
2. Conmutación de paquetes
3. Direccionamiento IPv4
4. Segmentación de redes
5. ***Reenvío de paquetes IP***
6. Fragmentación
7. Otros protocolos de nivel de red
8. Enrutamiento

Reenvío de paquetes IP (I)

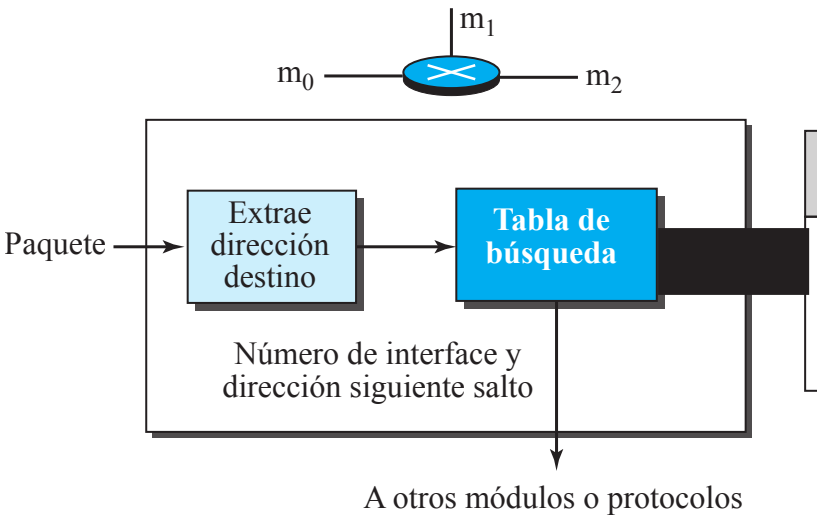


Tabla de reenvío

Dirección de red incluído máscara	Dir IP sig salto	Interface
$x_0.y_0.z_0.t_0/n_0$	.....	m_0
$x_1.y_1.z_1.t_1/n_1$	.....	m_1
$x_2.y_2.z_2.t_2/n_1$	.....	m_2

[*Forouzan*]

Los hosts y los routers disponen de la denominada **tabla de reenvío**.

Cuando un hosts tiene un paquete para enviar, o un paquete alcanza un router, buscan en la tabla de reenvío para encontrar el siguiente salto por el que enviar el paquete.

En direccionamiento classless, todo el espacio de direcciones es una entidad, no hay clases, por lo que, la tabla de reenvío necesita una entrada para cada bloque.

La búsqueda se realiza en base a la **dirección de red** (primera dirección del bloque).

La dirección destino en el paquete no incorpora información sobre la dirección de red, por lo tanto, es necesario incluir la máscara ($/n$) en la tabla.

La tabla de reenvío, en direccionamiento classless, incluye los siguientes campos:

[*Máscara*, *Dirección de red*, *Interface*, *Dirección IP del siguiente salto*]

Nota: la máscara y dirección de red, suelen aparecer juntas.

Reenvío de paquetes IP (II)

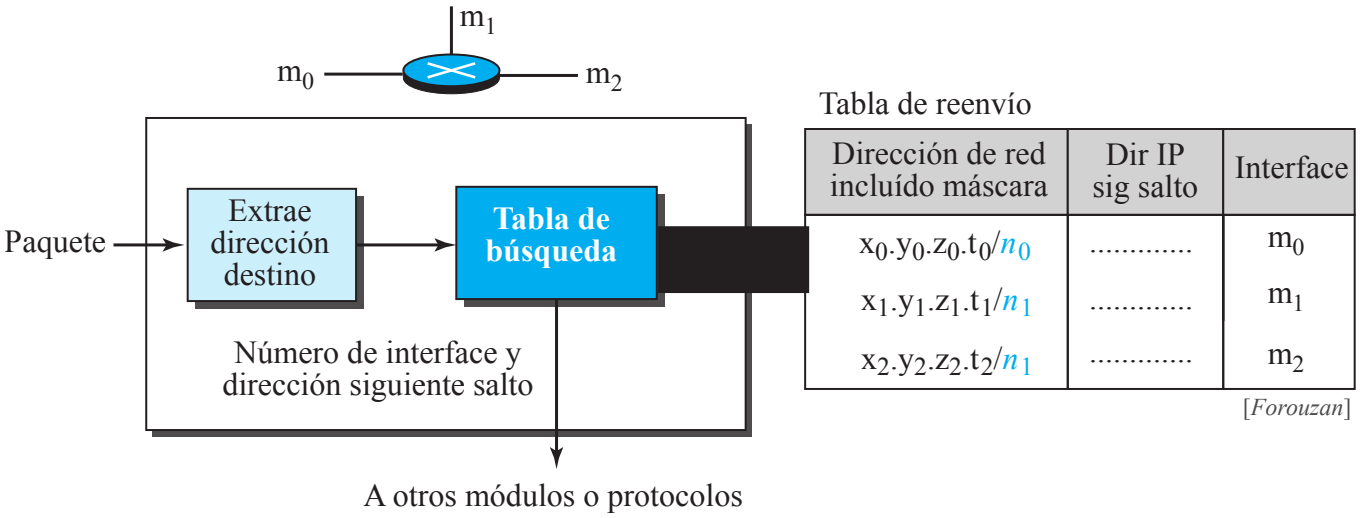


Tabla de reenvío

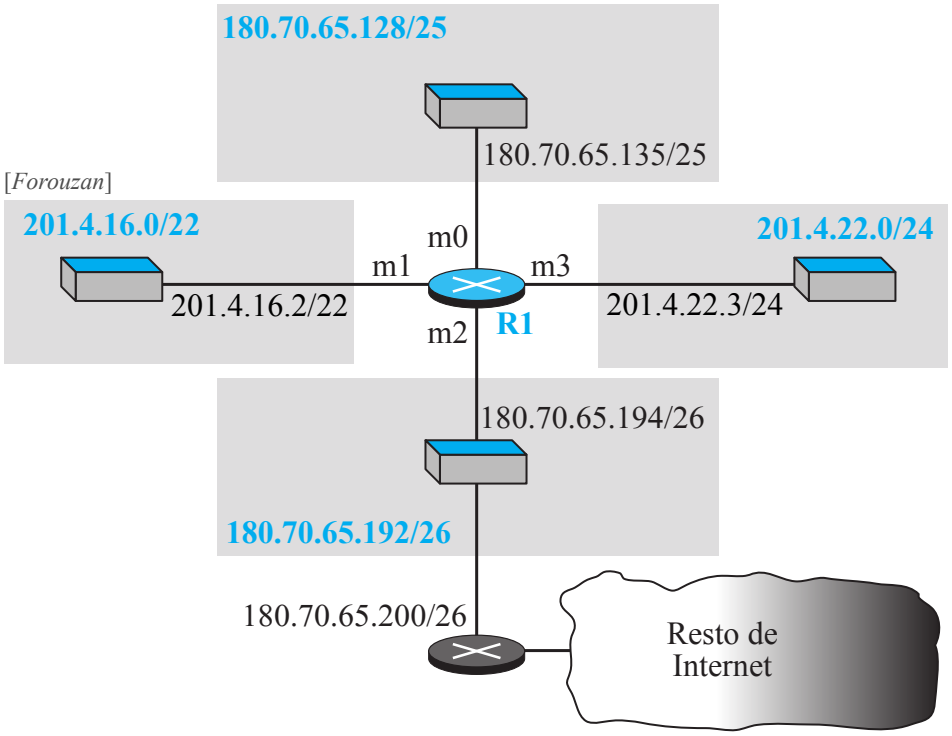
Dirección de red incluído máscara	Dir IP sig salto	Interface
$x_0.y_0.z_0.t_0/n_0$	.....	m_0
$x_1.y_1.z_1.t_1/n_1$	.....	m_1
$x_2.y_2.z_2.t_2/n_1$	.....	m_2

[*Forouzan*]

A partir de la **dirección destino** del paquete, el reenvío realiza las siguientes tareas:

- Busca en la tabla, fila a fila, de forma secuencial.
- En cada fila, los n bits más a la izquierda (prefijo) se mantienen, y el resto (sufijo) se establecen a 0.
- Si la dirección destino (dirección de red) coincide con la primera columna, se extrae la información de las otras columnas. En otro caso, continúa la búsqueda.
- Si no hay ninguna coincidencia, el paquete es **descartado**.
- Salvo que la última fila contenga la **ruta por defecto (default)**, que encaja con cualquier dirección.
- Habitualmente, la ruta por defecto, se indica por `0.0.0.0`.

Reenvío de paquetes IP: ejemplo (I)



La tabla de reenvío para el router **R1**:

Dirección/máscara de red	*Siguiente salto (hop)*	*Interface*
180.70.65.192/**26**	–	m2
180.70.65.128/**25**	–	m0
201.4.22.0/**24**	–	m3
201.4.16.0/**22**	–	m1
Default	180.70.65.200	m2

La tabla de reenvío para el router **R1**, utilizando, sólo, los bits de **prefijo**:

Bits más a la izquierda en la dirección destino	*Next hop*	*Interface*
`10110100 01000110 01000001 11`	–	m2
`10110100 01000110 01000001 1`	–	m0
`11001001 00000100 00011100`	–	m3
`11001001 00000100 000100`	–	m1
Default	180.70.65.200	m2

A partir de la dirección destino del paquete, los bits más la izquierda se comparan con la tabla, si coinciden:

... 26 bits, el paquete se reevía por $m2$

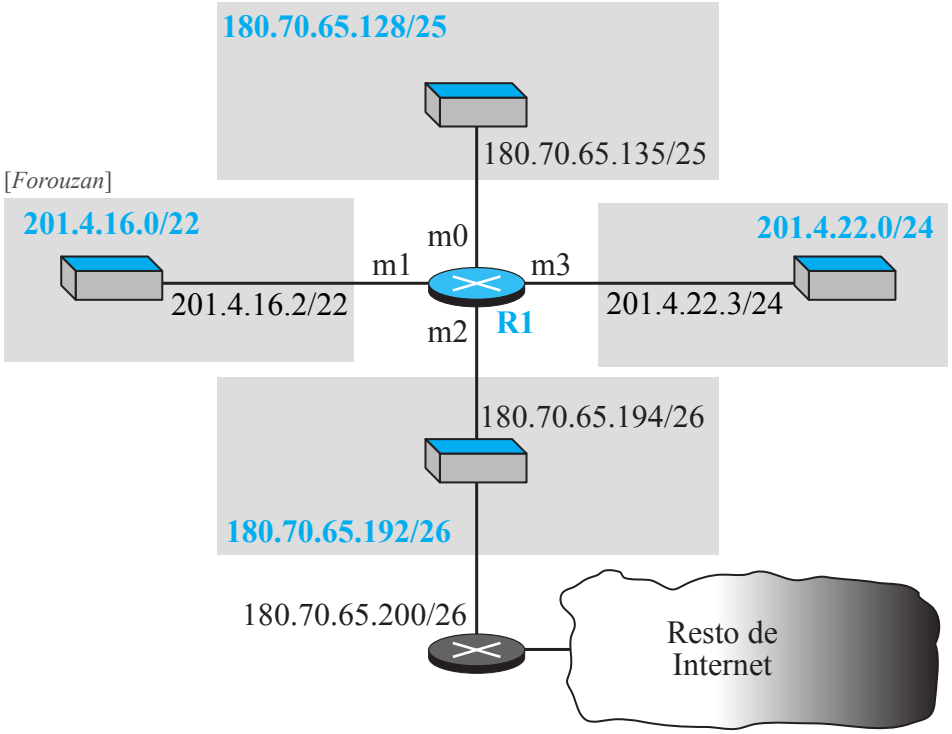
... 25 bits, el paquete se reevía por $m0$

... 24 bits, el paquete se reevía por $m3$

... 22 bits, el paquete se reevía por $m1$

En caso contrario, se reenvía a la ruta por defecto, es decir, `180.70.65.200` a través de *m2*.

Reenvío de paquetes IP: ejemplo (II)



Dirección/máscara de red	*Siguiente salto (hop)*	*Interface*
180.70.65.192/**26**	–	m2
180.70.65.128/**25**	–	m0
201.4.22.0/**24**	–	m3
201.4.16.0/**22**	–	m1
Default	180.70.65.200	m2

Bits más a la izquierda en la dirección destino	*Next hop*	*Interface*
`10110100 01000110 01000001 11`	–	m2
`10110100 01000110 01000001 1`	–	m0
`11001001 00000100 00011100`	–	m3
`11001001 00000100 000100`	–	m1
Default	180.70.65.200	m2

Supongamos un paquete, con dirección destino 180.70.65.140, llega a R1:

- Se aplica la primera máscara, (/26), a la dirección destino. El resultado es 180.70.65.128, que no coincide con la dirección de red.
- Se aplica la segunda máscra, (/25), el resultado es 180.70.65.128, que si coincide con la dirección de red, se extrae la información de siguiente salto e interface (*m0*) para el reenvío del paquete.

Supongamos un paquete, con dirección destino 140.60.20.10, llega a R1:

- Aplicando máscaras de forma sucesiva, comprueba que ninguna coincide con la dirección de red de la tabla.
- Si que encaja con la ruta por defecto, por lo que, se extrae la información de siguiente salto (180.70.65.200) e interface (*m2*) para el reenvío del paquete.

Agregación de rutas

Agregación de rutas (I)

En direccionamiento *classfull*:

- Sólo hay una entrada en la tabla de reenvío para cada sitio fuera de la organización.
- Esa entrada define el sitio, incluso si internamente ha sido segmentada (*subnetting*).
- Cuando un paquete llega a un router, chequea la correspondiente entrada y reenvía el paquete.

En direccionamiento *classless*:

- Es probable que el número de entradas en la tabla de reenvío se incremente de forma considerable, ya que el espacio de direcciones se divide en bloques.
- El aumento de tamaño puede resultar en un incremento en la cantidad de tiempo necesario para buscar en dicha tabla.
- Para aliviar este problema, se propone la **agregación de rutas**.

Agregación de rutas (II)

R1 está conectado a cuatro organizaciones, con 64 direcciones, cada una.

R2 está en algún sitio lejos de R1.

R1 tiene una tabla de reenvío más grande porque cada paquete debe ser correctamente reenvíado a la organización apropiada.

Pero R2 tiene una tabla mucho más pequeña.

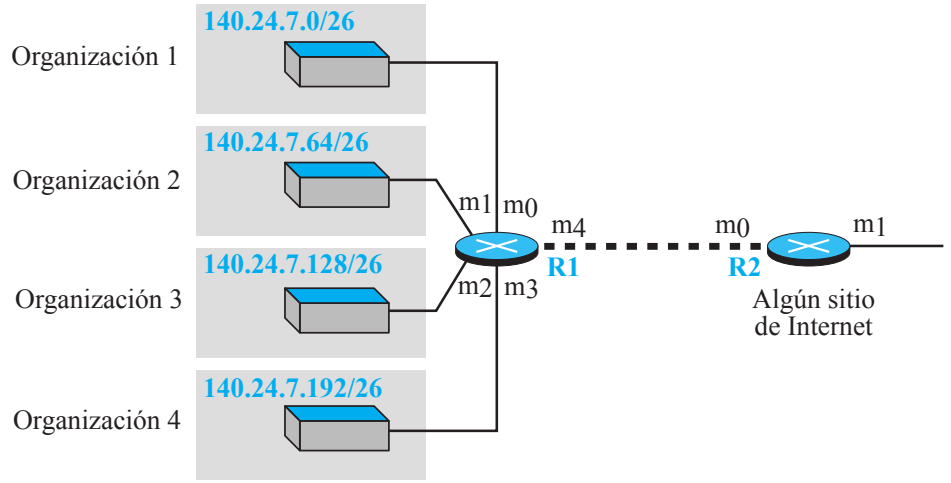


Tabla de reenvío para R1

Dirección red/mácara	Dirección siguiente salto	Interface
140.24.7.0/26	--------	m0
140.24.7.64/26	--------	m1
140.24.7.128/26	--------	m2
140.24.7.192/26	--------	m3
0.0.0.0/0	dirección de R2	m4

[Forouzan]

Tabla de reenvío para R2

Dirección red/mácara	Dirección siguiente salto	Interface
140.24.7.0/24	--------	m0
0.0.0.0/0	default router	m1

Para R2, cualquier paquete con destino `140.24.7.0` a la `140.24.7.255` debe ser reenviado por la interface *m0*, independientemente de la red destino final.

Se denomina **agregación de rutas** porque cuatro bloques se agregan en un sólo más grande.

R2 tendría una tabla de reenvío más grande si las cuatro organizaciones tuvieran direcciones que no pudieran ser agregadas en un bloque.

Máscara más larga (I)

¿Qué ocurre si una organización no está, geográficamente, cercana a las otras?

Supongamos que la organización 4 no puede ser conectada a R1.

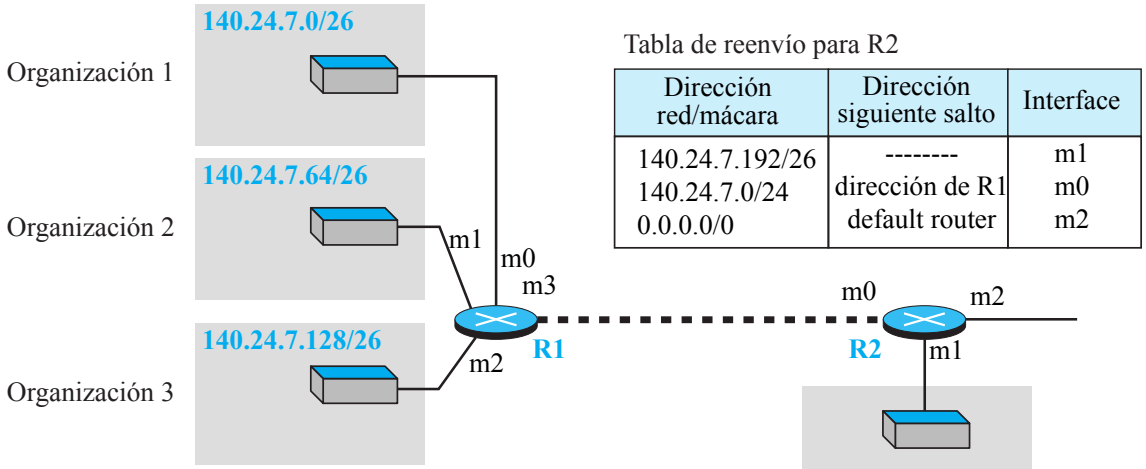


Tabla de reenvío para R2

Dirección red/mácara	Dirección siguiente salto	Interface
140.24.7.192/26	--------	m1
140.24.7.0/24	dirección de R1	m0
0.0.0.0/0	default router	m2

Tabla de reenvío para R1

Dirección red/mácara	Dirección siguiente salto	Interface
140.24.7.0/26	--------	m0
140.24.7.64/26	--------	m1
140.24.7.128/26	--------	m2
0.0.0.0/0	default router	m3

[Forouzan]

¿Podemos seguir manteniendo la agregación y mantener el bloque 140.24.7.192/26 a la organización 4?

Respuesta: **Si**

Debido a que enrutamiento classless utiliza el principio de encaje por la **máscara más larga**.

La tabla de reenvío se ordena por longitud de máscara, de mayor a menor.

Máscara más larga (II)

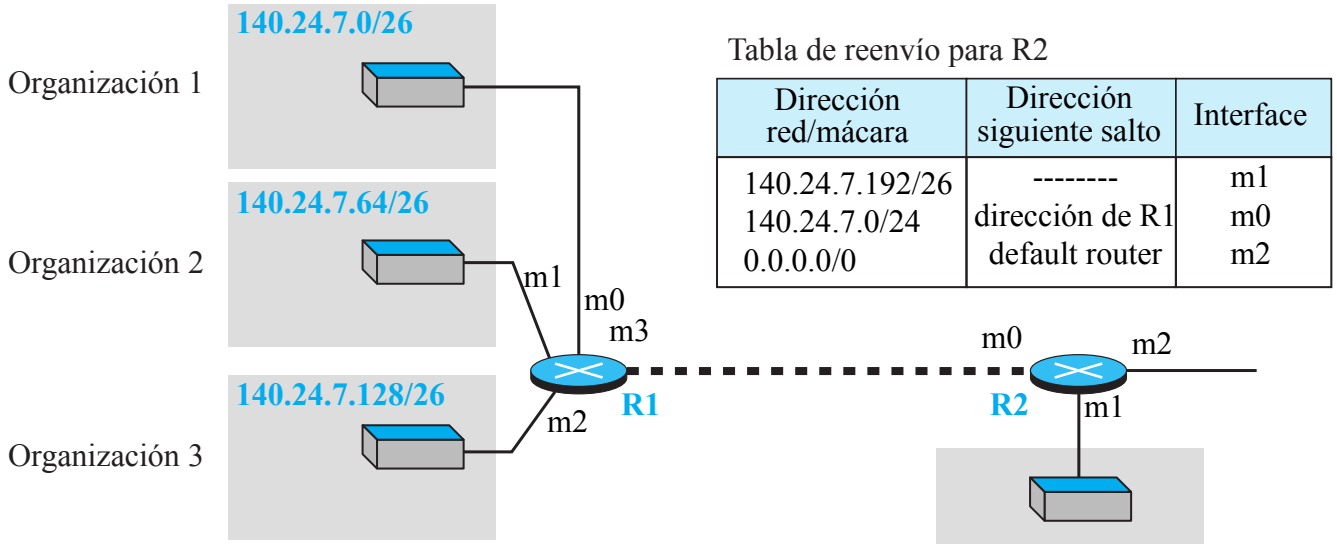


Tabla de reenvío para R2

Dirección red/mácara	Dirección siguiente salto	Interface
140.24.7.192/26	--------	m1
140.24.7.0/24	dirección de R1	m0
0.0.0.0/0	default router	m2

Tabla de reenvío para R1

Dirección red/mácara	Dirección siguiente salto	Interface
140.24.7.0/26	--------	m0
140.24.7.64/26	--------	m1
140.24.7.128/26	--------	m2
0.0.0.0/0	default router	m3

[*Forouzan*]

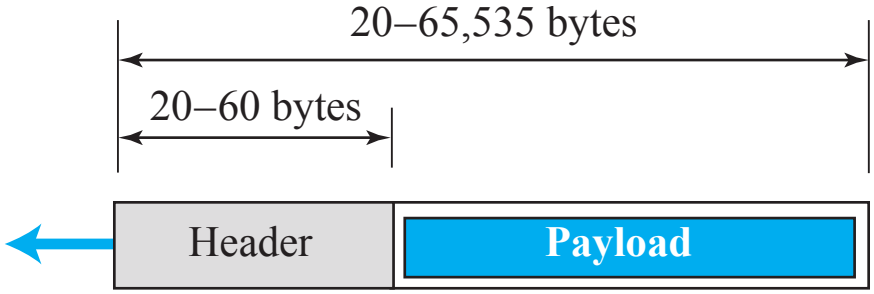
Supongamos que un paquete alcanza el router R2 para la organización 4 con dirección destino `140.24.7.200`.

- Aplica la primera máscara en R2, obteniendo una dirección de red `140.24.7.192`.
- El paquete se enruta correctamente a través de *m1* hacia la organización 4.

Si la tabla no estuviera ordenada por el prefijo más largo, habría aplicado máscara /24, y por lo tanto, habría resultado un enrutamiento incorrecto del paquete hacia R1.

1. Introducción al nivel de red
2. Conmutación de paquetes
3. Direccionamiento IPv4
4. Segmentación de redes
5. Reenvío de paquetes IP
6. ***Fragmentación***
7. Otros protocolos de nivel de red
8. Enrutamiento

Datagrama IP: cabecera



(a) Datagrama IP

Leyenda

VER: nº versión
IHL: Header length
byte: 8 bits

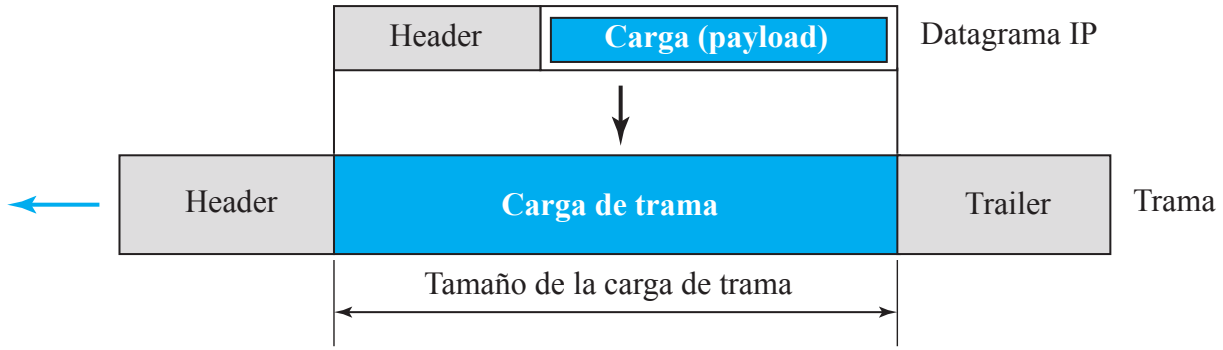
Flags: D M

[*Forouzan*]

0	4	8	16	31
VER 4 bits	HLEN 4 bits	Service type 8 bits	Total length 16 bits	
Identification 16 bits			Flags 3 bits	Fragmentation offset 13 bits
Time-to-live 8 bits		Protocol 8 bits	Header checksum 16 bits	
Source IP address (32 bits)				
Destination IP address (32 bits)				
Options + padding (0 to 40 bytes)				

(b) Cabecera IP

Fragmentación IP



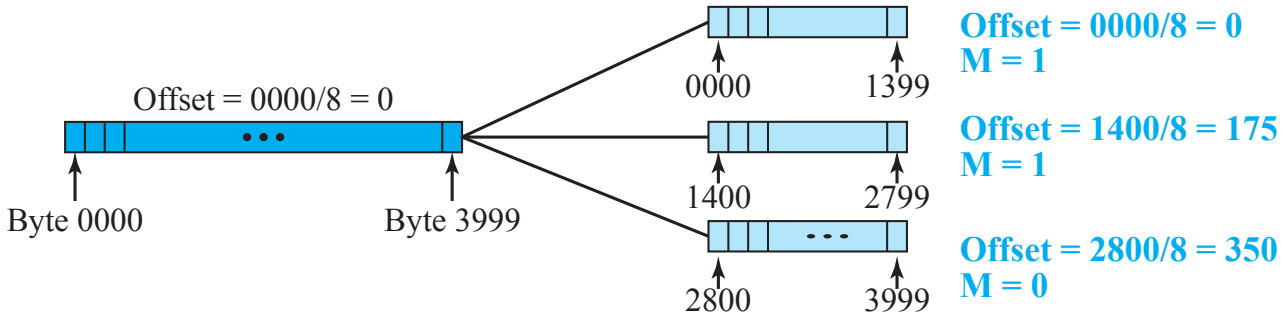
MTU: Tamaño máximo de carga de la trama

Cada protocolo de enlace tiene su propio formato.

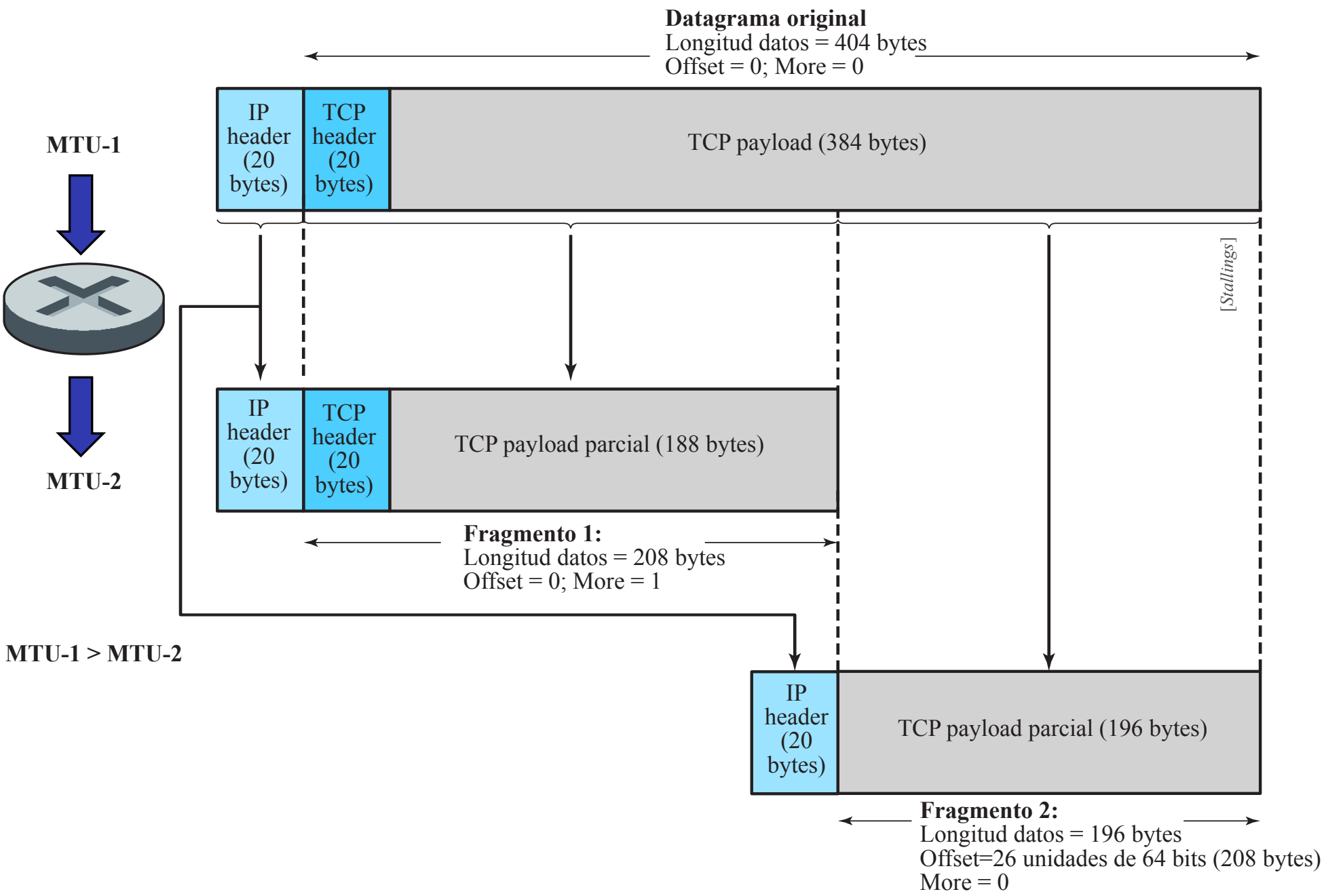
Una de las características de cada formato es el tamaño máximo de carga (*payload*) que puede encapsular, es decir, un datagrama encapsulado en una trama, el tamaño total debe ser menor que un tamaño máximo, definido por la MTU (*Maximum Transfer Unit*).

En caso contrario, se necesita **fragmentación**.

El encargado de la fragmentación, y posterior reensamblado, es el router, mediante el campo *offset* (grupos de 8 bytes) y el flag `M`.



Fragmentación IP: ejemplo

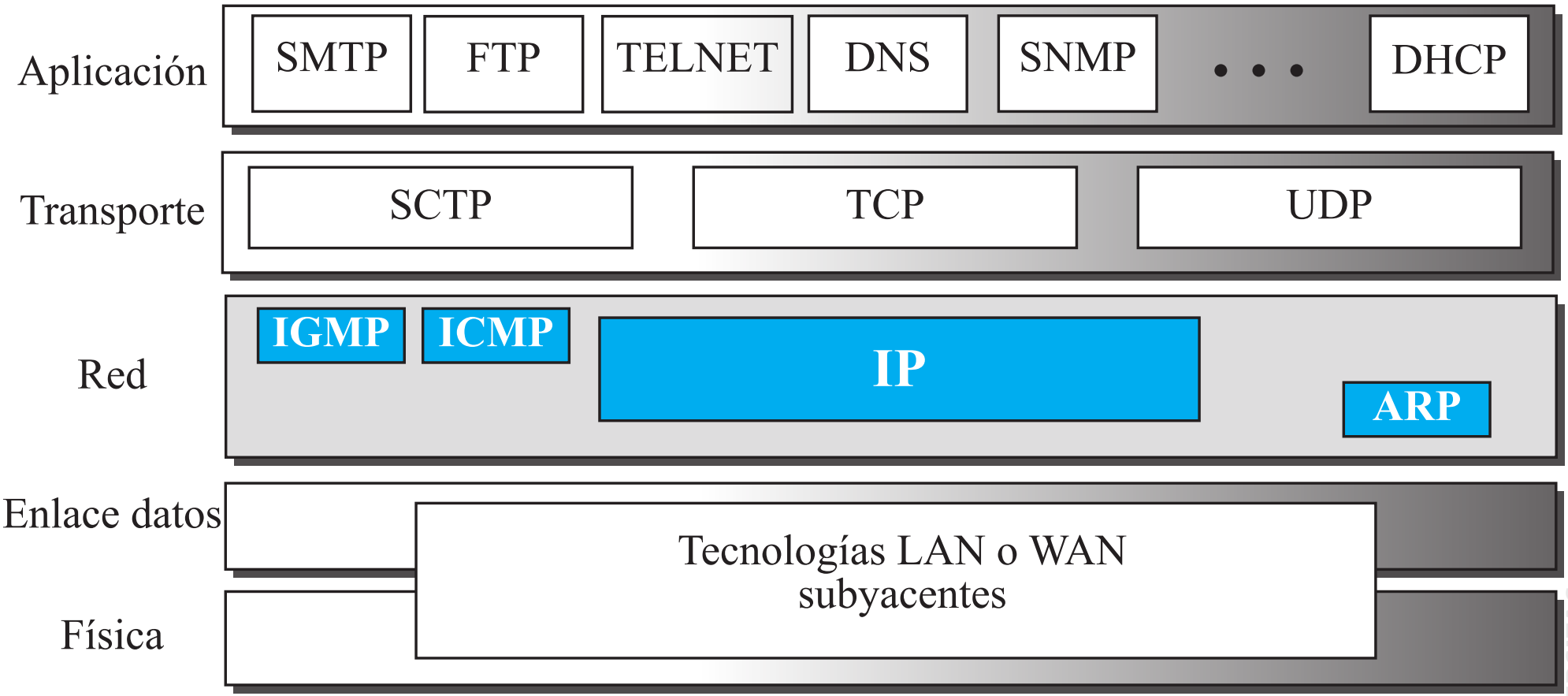


Nivel de red

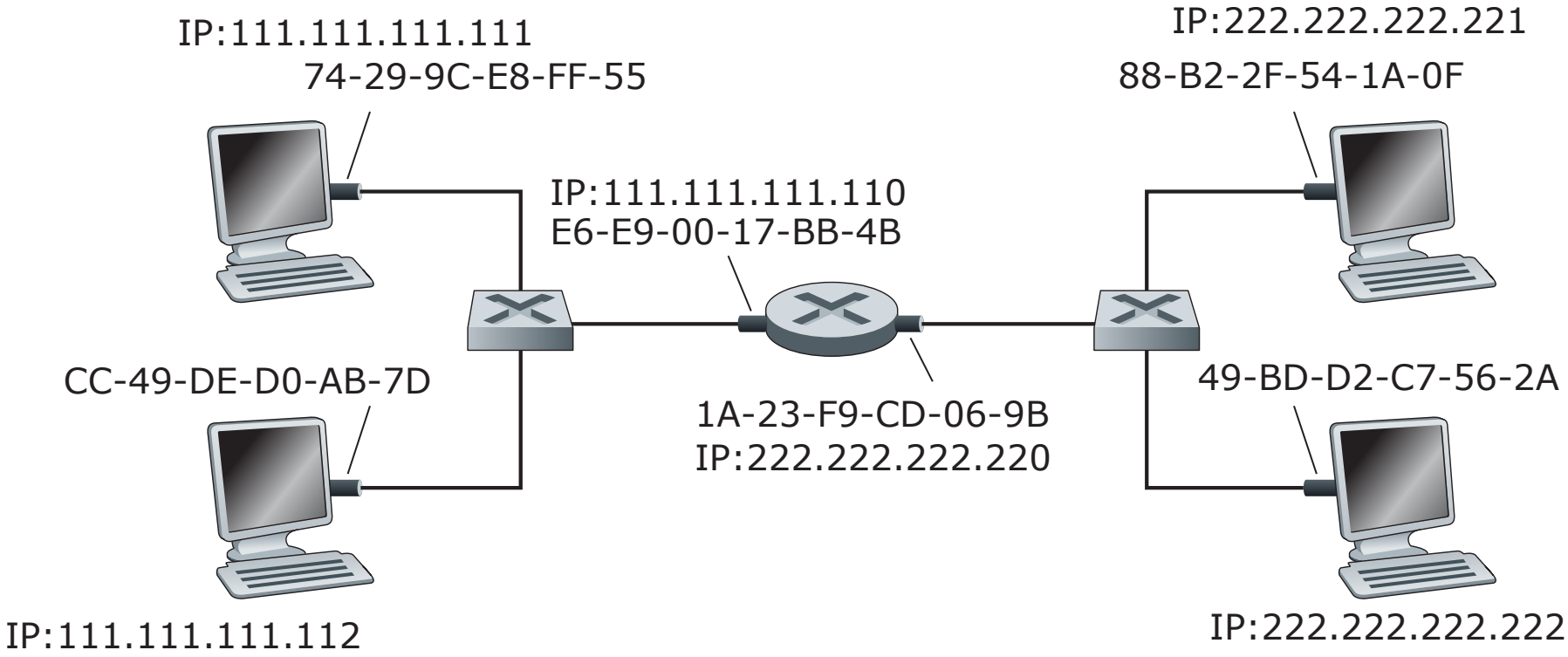
1. Introducción al nivel de red
2. Conmutación de paquetes
3. Direccionamiento IPv4
4. Segmentación de redes
5. Reenvío de paquetes IP
6. Fragmentación
7. ***Otros protocolos de nivel de red***
8. Enrutamiento

Pila protocolos TCP/IP

[*Forouzan*]

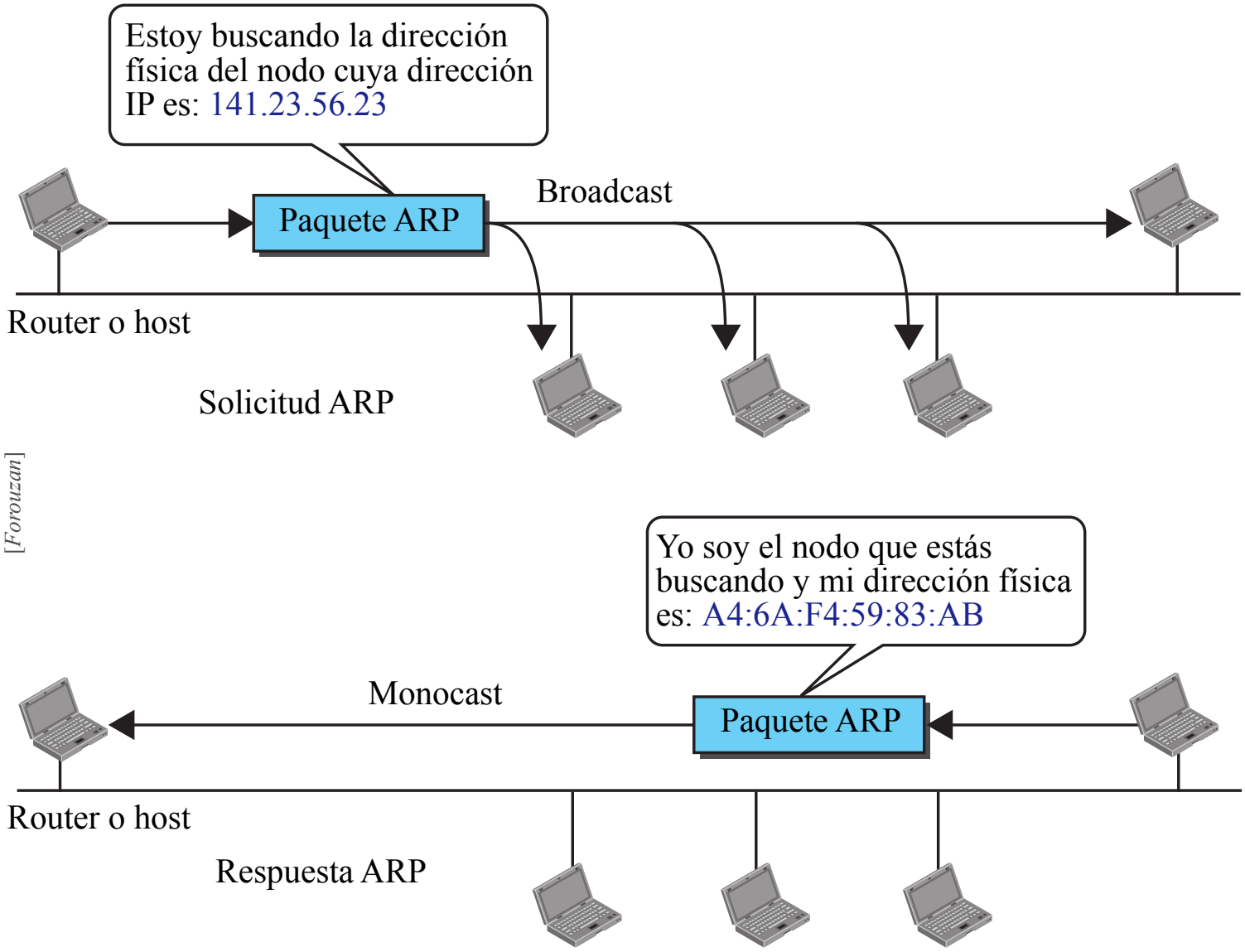


IP y MAC



Cada dispositivo conectado a Internet debe de disponer de una dirección MAC e IP.

ARP (*Address Resolution Protocol*)



ARP se utiliza para averiguar la MAC de un dispositivo a partir de su IP.

ICMP: *ping* y *traceroute*

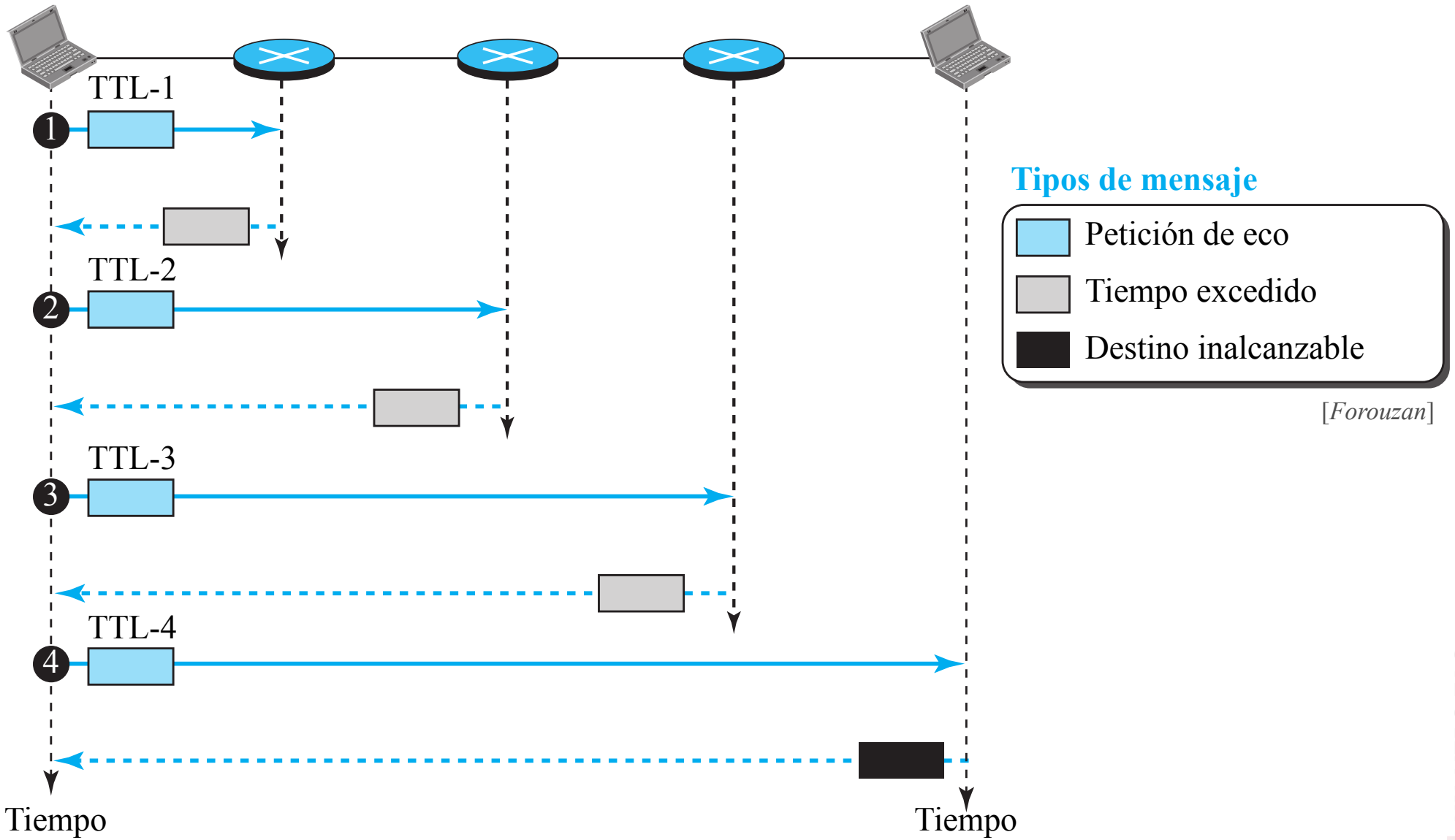
RARP (*Reverse Address Resolution Protocol*):

- RARP permite a un host descubrir su dirección de red cuando sólo conoce su dirección física.
- Utilizado en host que no tienen posibilidad de guardar dirección IP (terminales ligeros), proceso de arranque.

ICMP (*Internet Control Message Protocol*)

- Protocolo utilizado por hosts y gateways para enviar notificaciones de error.
- Utiliza los mecanismos de test y eco de direcciones IP para comprobar su accesibilidad.
- *ping*, *traceroute*

Protocolo ICMP: *traceroute*

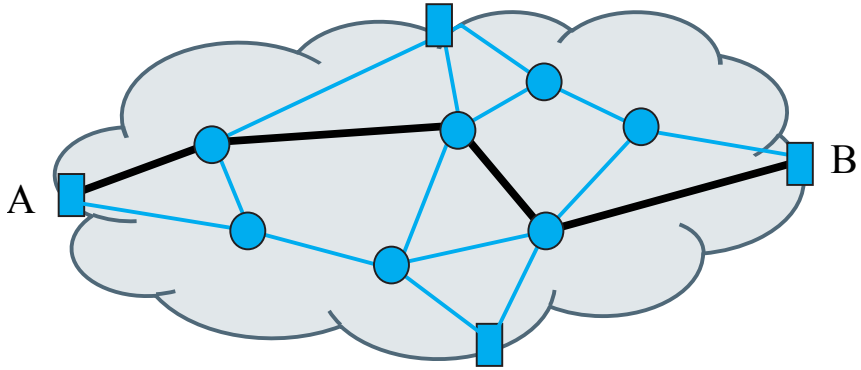


[*Forouzan*]

Nivel de red

1. Introducción al nivel de red
2. Conmutación de paquetes
3. Direccionamiento IPv4
4. Segmentación de redes
5. Reenvío de paquetes IP
6. Fragmentación
7. Otros protocolos de nivel de red
8. ***Enrutamiento***

Fundamentos de enrutamiento



Las funciones de enrutamiento son fundamentales para la interconexión de redes.

Permite la comunicación de dos nodos, A y B, que no están directamente conectados.

Los paquetes se encaminan salto a salto, desde el origen hasta el destino, con la ayuda de las tablas de reenvío (**forwarding table**).

- Los hosts origen y destino no necesitan tabla de reenvío porque envían/reciben el paquete a/de su router por defecto (*gateway*) en su LAN.
- Sólo los routers se encargan de unir las redes (**vecinos**), por lo tanto, son los únicos que necesitan las tablas de reenvío.

El proceso de enrutamiento consiste en encaminar un paquete desde el router origen (gateway del host origen) hasta el router destino (gateway de la red destino).

Al existir diferentes rutas y caminos alternativos entre router origen y destino:

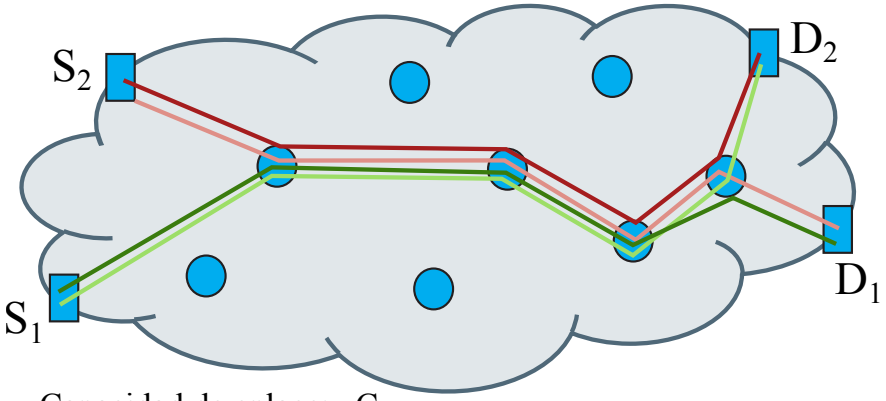
¿Cómo elegir la mejor ruta?

Enrutamiento y capacidad

- En redes broadcast no se necesita enrutamiento.
- El tráfico soportado máximo depende de la capacidad del canal.
- En redes en malla, múltiples flujos pueden utilizar el mismo enlace de forma simultánea.

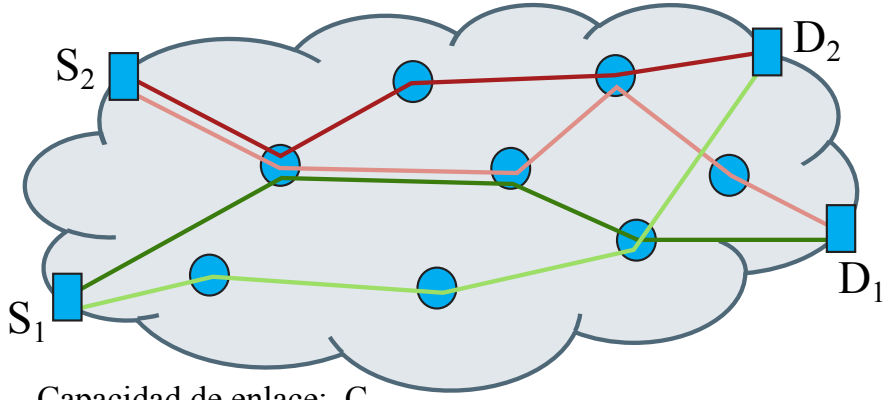
¿Qué enlaces utilizados influyen sobre el rendimiento de la red?

Sin Planificación de enrutamiento



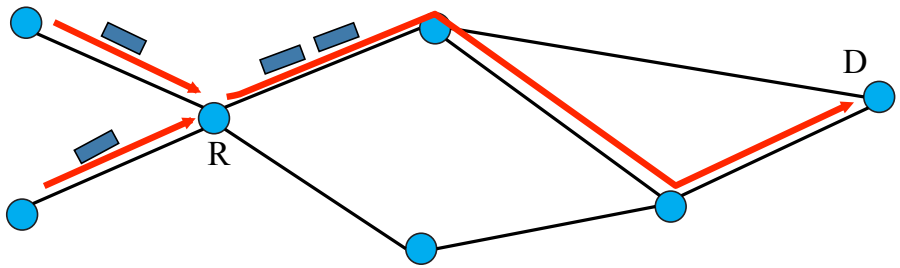
Capacidad de enlace: C
Tráfico máximo : 4C

Planificación de enrutamiento



Capacidad de enlace: C
Tráfico máximo : 3C

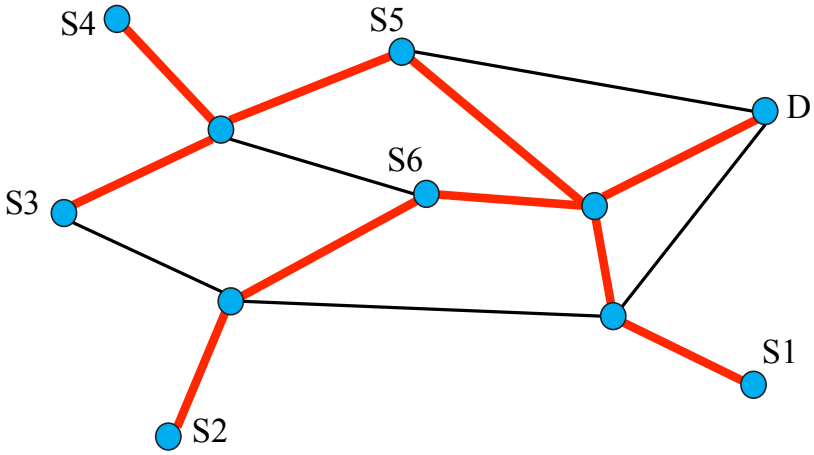
Enrutamiento en Internet



El reenvío se basa en la siguiente información:

- Nodo destino
- Siguiente salto (*next hop*)

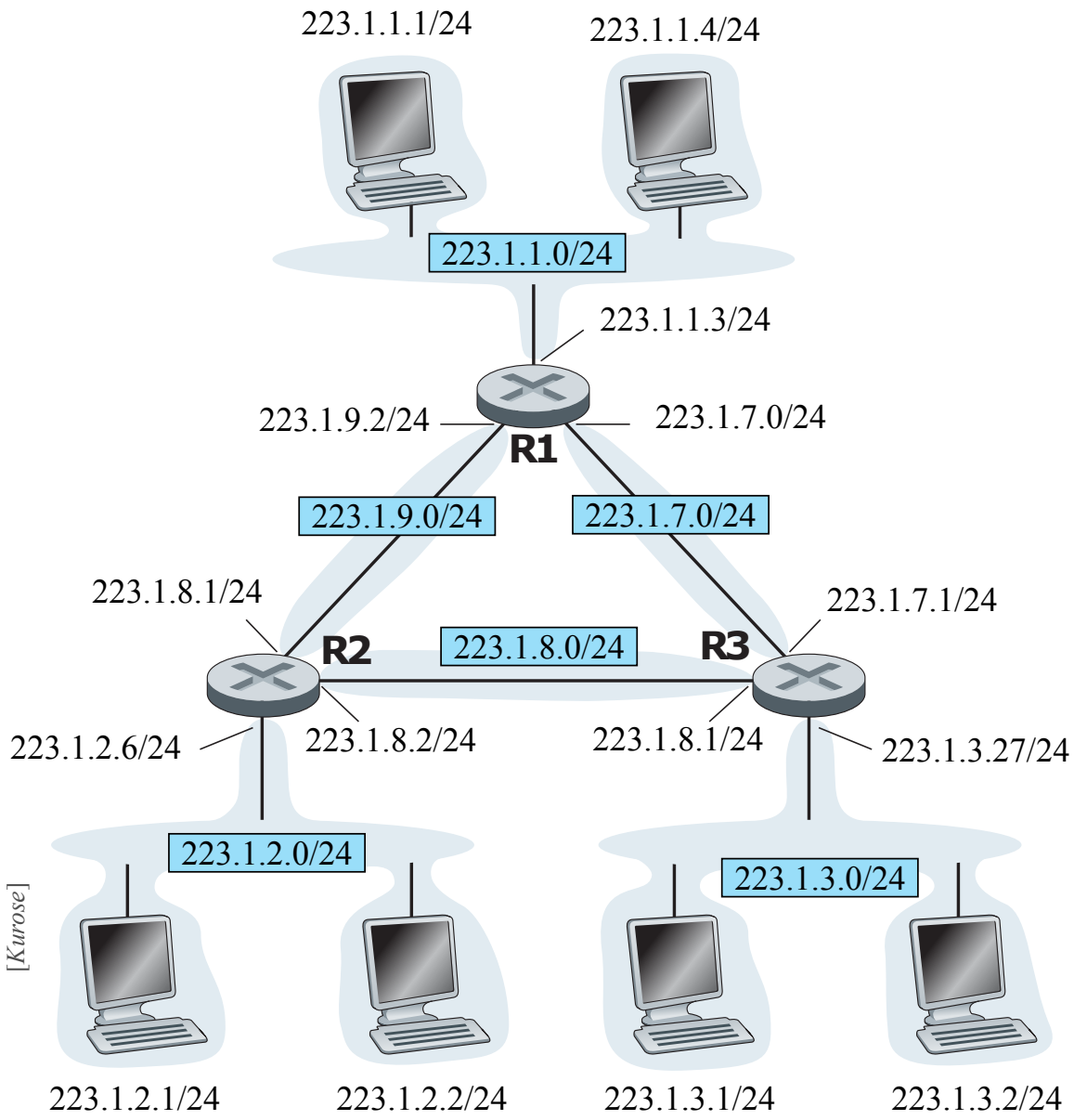
Consecuencia: todos los paquetes con destino D que lleguen a R seguirán el mismo camino después de $R \longrightarrow$ congestión.



Además, tenemos la siguiente restricción:

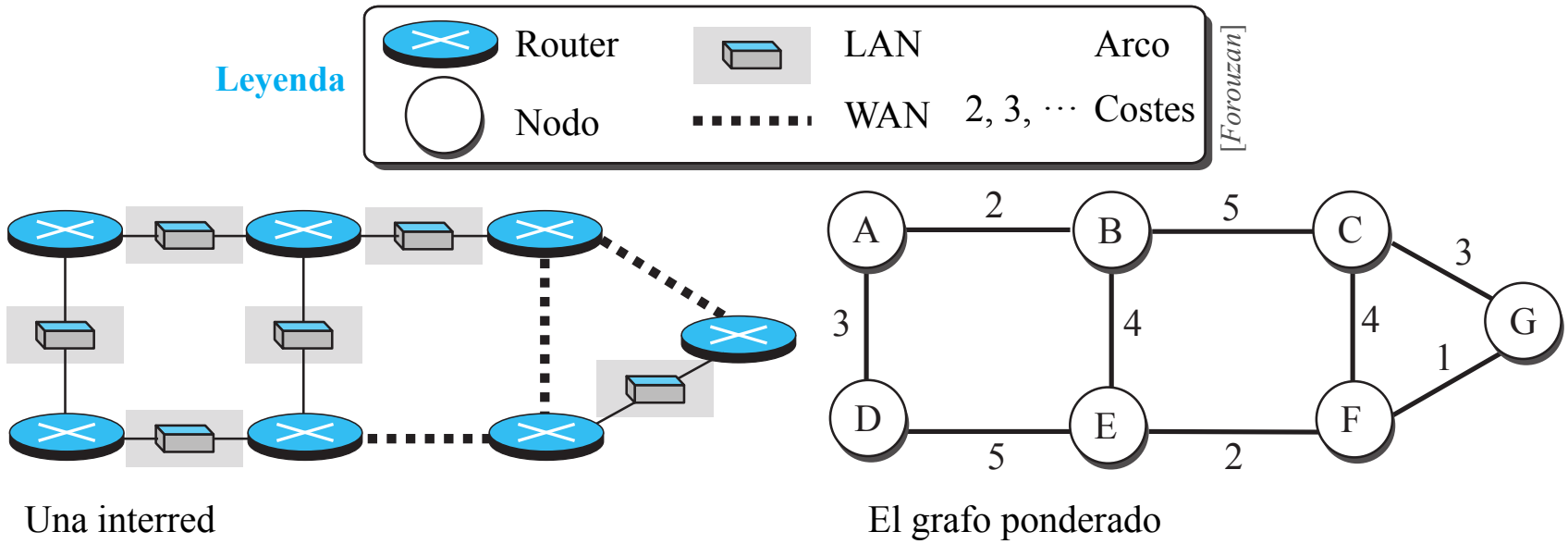
- Todas las rutas desde todas las fuentes a un destino D deben formar un **árbol**, para cada D.
- Los pares origen-destino no pueden ser enrutados de forma independiente del resto de pares.

Los routers forman parte de las redes



[Kurose]

Interred y grafos



Una interred

El grafo ponderado

Para encontrar la mejor ruta, la interred se modela como una **grafo** ponderado:

- Los routers son **nodos**.
- La red entre cada par de routers es un **arco**.

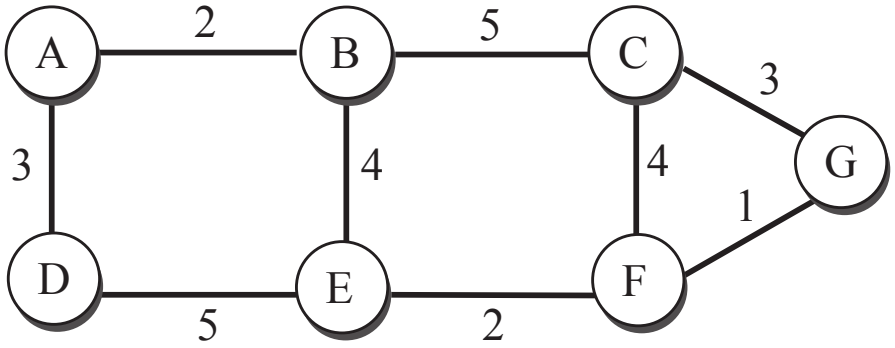
Grafo ponderado → los arcos tienen un **coste** asociado.

Si no hay arco entre dos nodos entonces el coste es infinito (∞).

Objetivo: encontrar la mejor ruta entre origen y destino, es decir, **con el menor coste**.

Cada router necesita encontrar la ruta de menor coste hacia el resto de routers de la red.

Árbol de mínimo coste (I)



Si hay N routers en una interred, entonces hay $(N-1)$ caminos mínimos desde cualquier router hasta cualquier otro router.

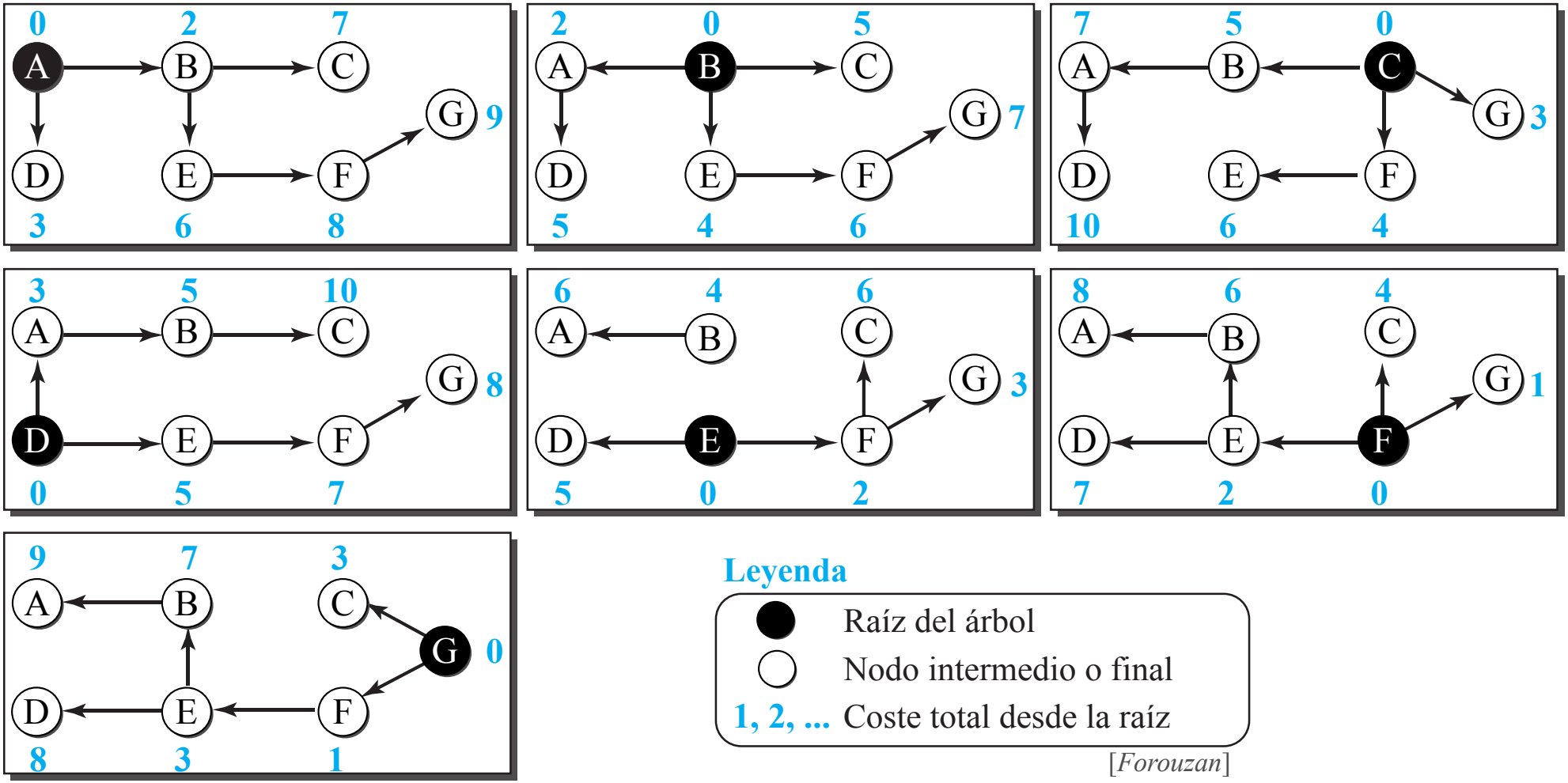
Por lo tanto, se necesitan $N \times (N-1)$ caminos de mínimo coste para modelar una interred de tamaño N nodos.

Por ejempo, para $N = 10$, se necesitan 90 caminos de mínimo coste.

Para optimizar, todos los caminos de mínimo coste se combinan en un árbol de mínimo coste.

Un **árbol de coste mínimo** es un árbol con el router origen como raíz, y se realiza la expansión del grafo (visitando el resto de nodos), utilizando siempre el camino más corto entre todo par de nodos.

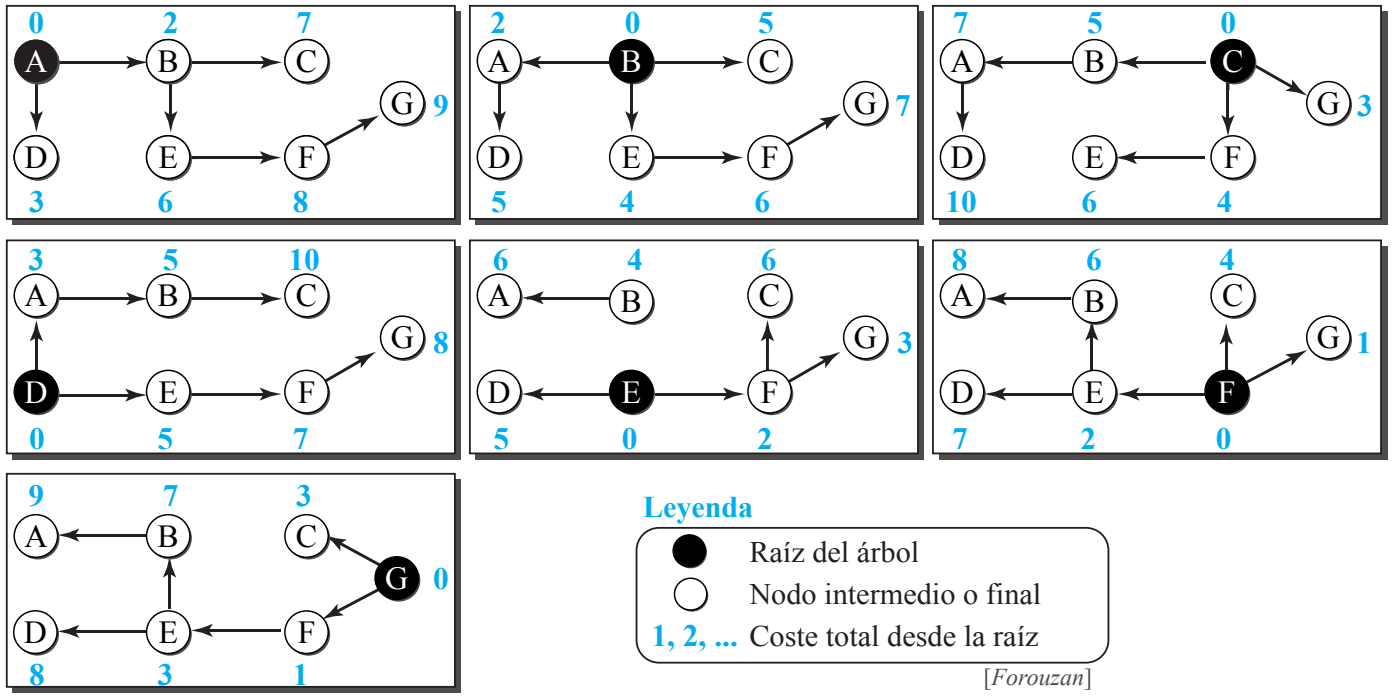
Árbol de mínimo coste (II)



[*Forouzan*]

$N = 7$ nodos $\rightarrow N \times (N-1) = 7 \times 6 = 42$ caminos de mínimo coste

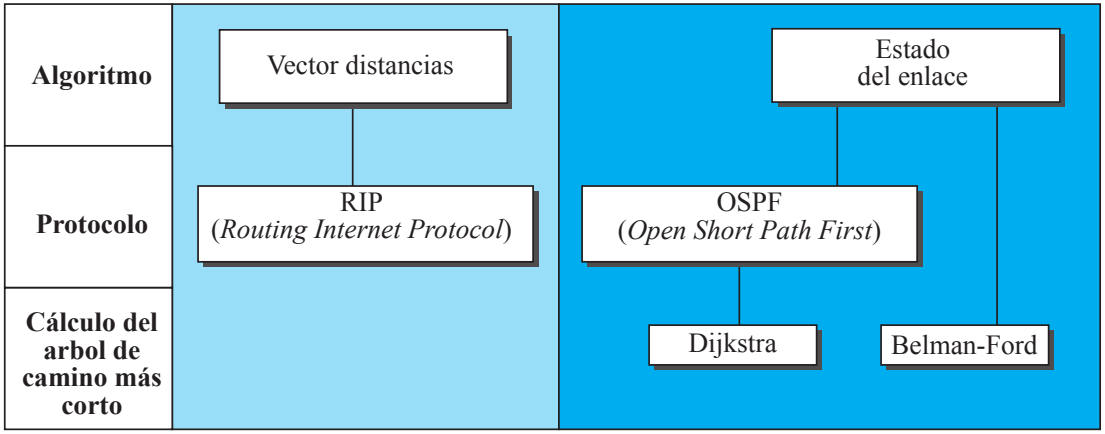
Árbol de mínimo coste: propiedades



[*Forouzan*]

1. La ruta de mínimo coste entre X e Y en el árbol de X es la inversa a la ruta de mínimo coste entre Y y X en el árbol de Y; el coste en ambas direcciones es el mismo:
   - La ruta $A \to F$, desde el árbol de A, es $(A \to B \to E \to F)$.
   - La ruta $F \to A$, desde el árbol de F, es $(F \to E \to B \to A)$, es decir, la inversa.
   - El coste es el mismo: 8.
2. En lugar de viajar de X a Z mediante el árbol de X, se puede viajar de X a Y (con el árbol de X), y continuar de Y hasta Z (con el árbol de Y):
   - Podemos ir de $A \to G$, con el árbol de A, mediante $(A \to B \to E \to F \to G)$.
   - También podemos ir de $A \to E$, con el árbol de A, y hacer $(A \to B \to E)$, y continuar con el árbol de E, mediante $(E \to F \to G)$.
   - La combinación de ambas rutas (segundo caso) es equivalente a una sóla ruta (primer caso).
   - El coste en el primer caso es 9; el coste en el segundo caso es $(6 + 3 = 9)$.

Protocolos y algoritmos de enrutamiento



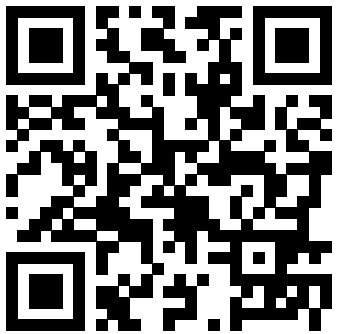
Protocolo de enrutamiento:

- Comprende dos funcionalidades diferentes:
  1. Intercambio de información en relación a la topología de la red, tráfico, etc.
  2. Creación y mantenimiento de la tabla de enrutamiento.
- Formalmente, (1) es el algoritmo de enrutamiento.
- En la práctica, (1) y (2) son fases que se realizan conjuntamente.
- La forma en la que se crean las tablas de enrutamiento depende del intercambio de mensajes de enrutamiento, y viceversa.

Algoritmo de enrutamiento:

- Define el criterio de cómo elegir el camino entre la fuente y el destino.
- ... y construir la tablas de enrutamiento.
- El criterio de elección depende del tipo de red (datagrama o circuito virtual).

Enrutamiento vector distancia



Enrutamiento Vector Distancia

Los routers intercambia información específica de conectividad mediante el **Vector Distancia** (DV, *Distance Vector*):

[*dirección destino*, *distancia*]

El DV se envía con los siguientes criterios:

- Sólo a los routers directamente conectados (**vecinos**).
- Periódicamente y/o cuando se produce cambio en la topología.

Cada nodo crea su propio árbol de camino más corto con información mínima acerca de sus vecinos inmediatos.

Los DV se intercambian entre los vecinos de forma que los árboles van aumentando de complejidad hasta representar la interred en su totalidad.

Cada router informa a sus vecinos, de forma continua, acerca de lo que sabe sobre la totalidad de la red (aunque la información sea incompleta).

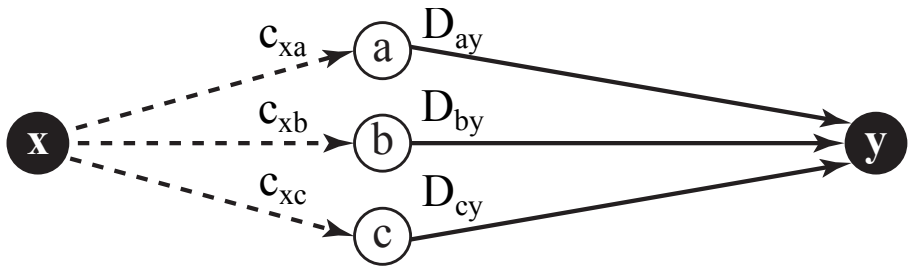
La estimación de la distancia se realiza mediante la ecuación de Bellman-Ford.

Ecuación de Bellman-Ford

La ecuación de Bellman-Ford se utiliza, a partir de un grafo ponderado, para calcular el coste mínimo (distancia más corta) entre un nodo origen x y un nodo destino y, a través de una serie de nodos intermedios $(a, b, c, \cdots)$.

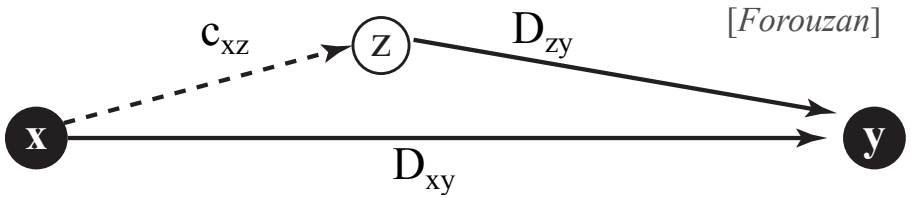
Caso general: D_{ij} representa la distancia más corta y c_{ij} el coste entre los nodos i y j:

$$D_{xy} = min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \cdots\}$$



Actualización de ruta: normalmente, lo que se pretende es actualizar un coste mínimo existente con un coste a través de un nodo intermedio, por ejemplo, z, si el segundo es más pequeño.

$$D_{xy} = min\{D_{xy}, (c_{xz} + D_{zy})\}$$



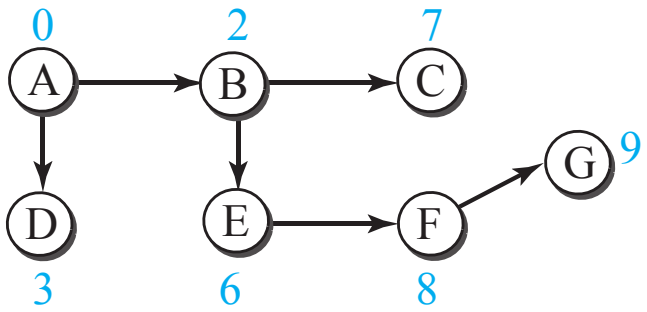
[*Forouzan*]

El algoritmo de Bellman-Ford permite construir una nueva ruta de mínimo coste a partir de otra ruta de mínimo coste, previamente establecida. Por ejemplo:

- Sean las rutas de mínimo coste previamente establecidas $(a \rightarrow y)$, $(b \rightarrow y)$ y $(c \rightarrow y)$.
- Posteriormente, aparece la ruta de mínimo coste $(x \rightarrow y)$, que sustituye a las anteriores.

La ecuación se convierte en un constructor de rutas si se utiliza repetidamente (en modo background).

Vector distancias



Árbol del nodo A

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

Vector distancia del nodo A

A partir del concepto **vector distancia** se obtiene el nombre **enrutamiento vector distancias**.

El **árbol de coste mínimo** es una combinación de rutas de coste mínimo desde la raíz hasta todos los destinos.

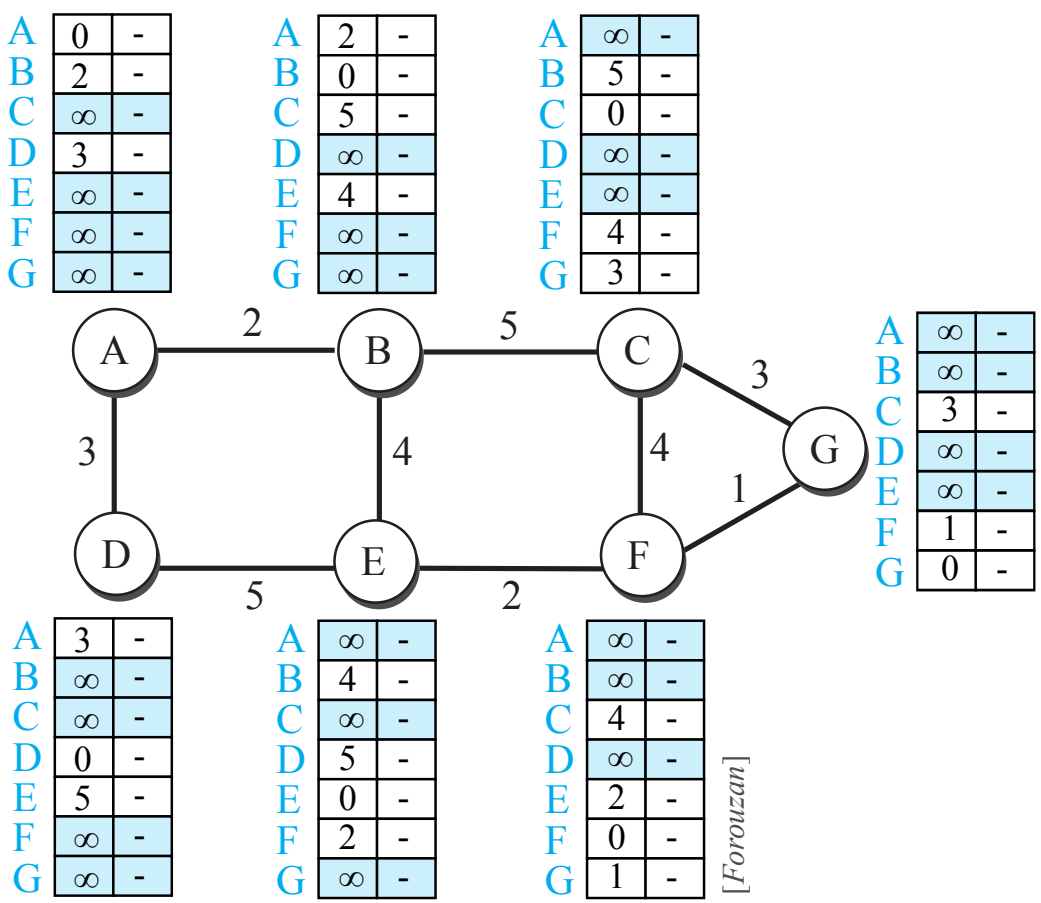
Todas las rutas están, gráficamente unidas, conformando el **árbol**.

El enrutamiento por vector distancia divide las rutas y crea un vector distancia, que consiste en un array unidimensional que representa el árbol.

El nombre del vector distancia define la **raíz**, los índices definen los destinos y los valores de cada celda define el coste mínimo desde la raíz hasta el destino.

Un vector distancia no proporciona el camino hasta los destinos (como proporciona el árbol de mínimo coste); proporciona sólo el coste mínimo hasta los destinos.

Cálculo del vector distancias: situacion inicial



En una fase inicial, los nodos crean las vectores a partir de la información que disponen (que es bastante limitada).

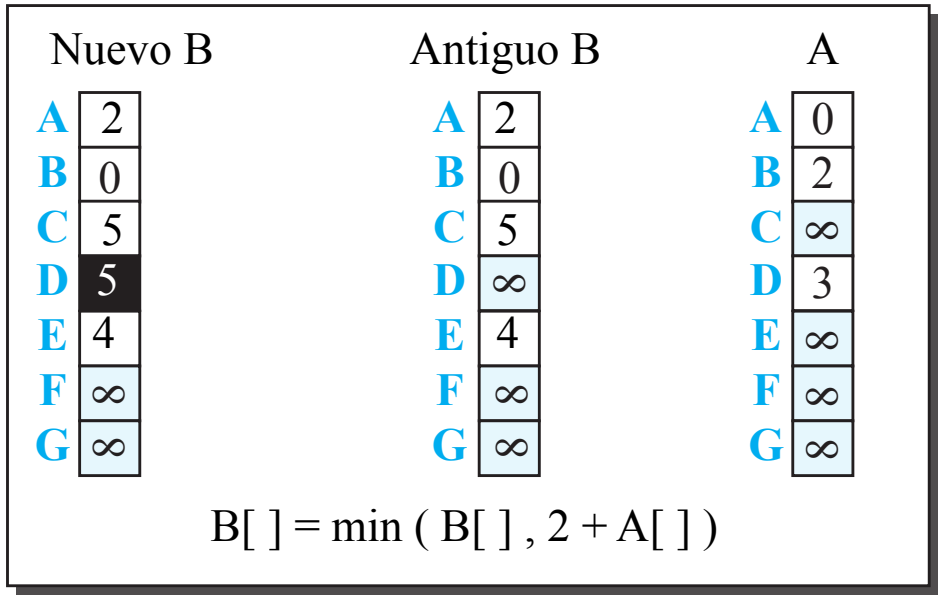
Por ejemplo, **A** cree que no está conectado a **G**, ya que la celda correspondiente indica una información de mínimo coste ∞.

De forma **asíncrona**, envían una copia del vector distancias a sus vecinos.

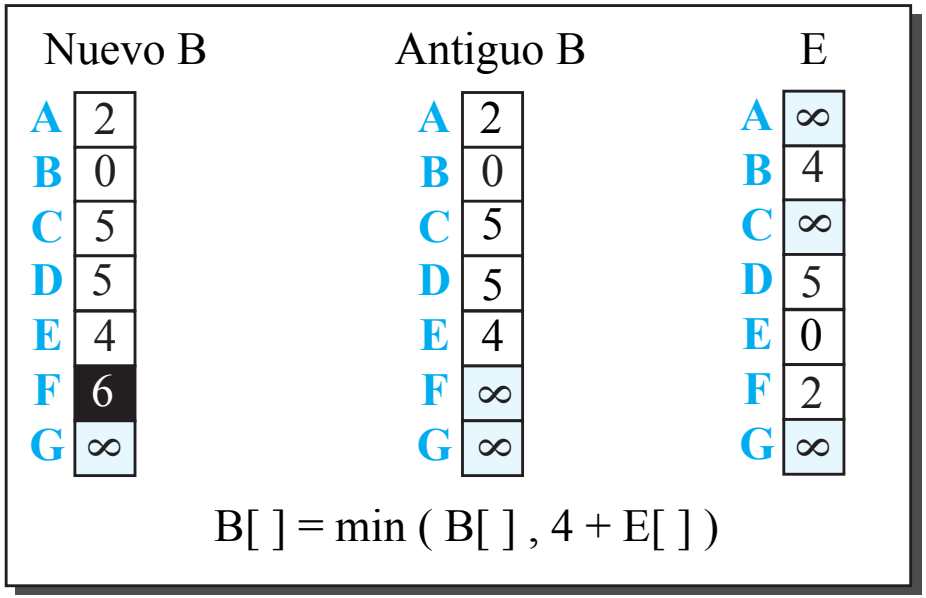
Cálculo del vector distancias: actualización de B

Nota:
X[ ]: el vector total

[*Forouzan*]

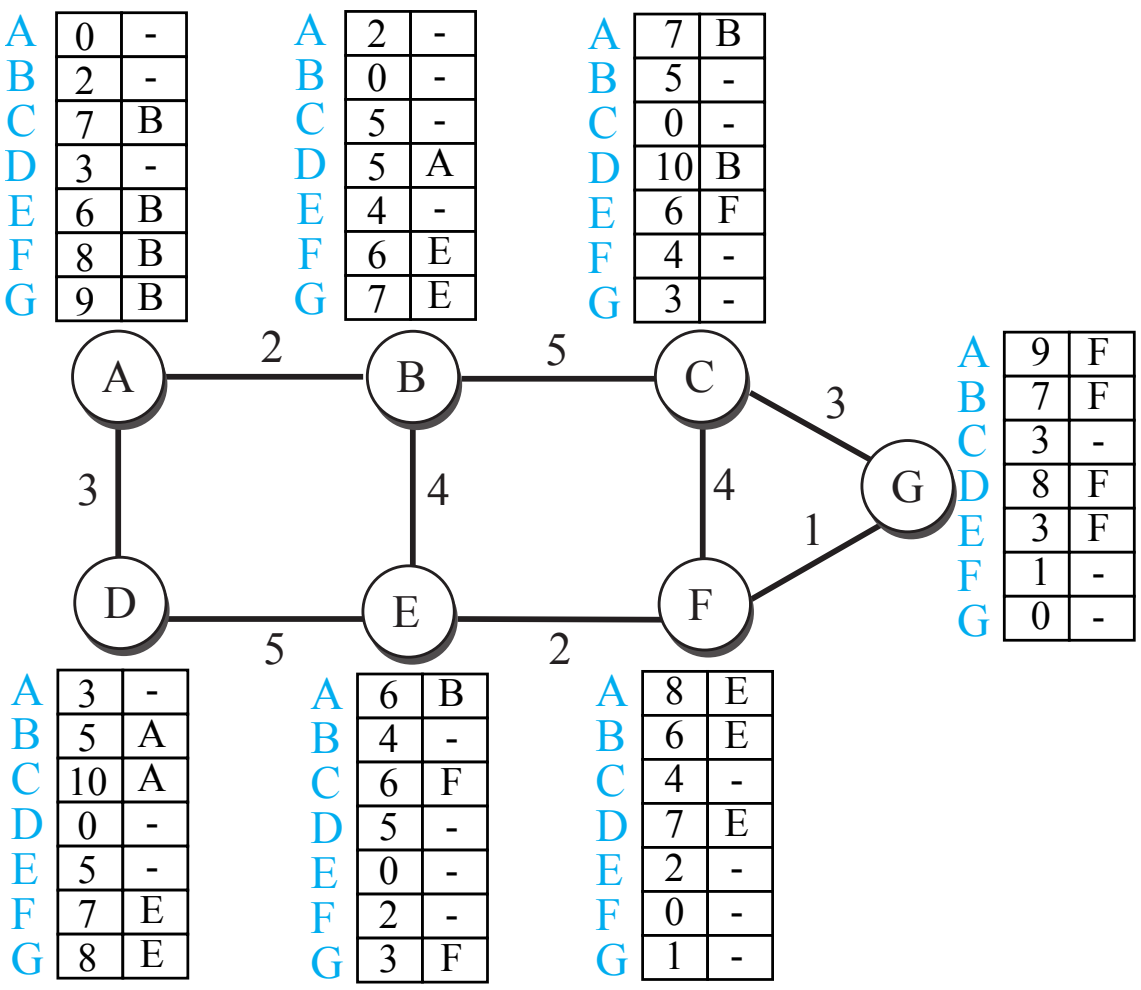


Primer evento: B recibe una copia del vector de A



Segundo evento: B recibe una copia del vector de E

Cálculo del vector distancias: situación final



Conteo al infinito

Un problema del routing vector distancias es que, mientras la disminución del coste (buenas noticias) se propagan rápidamente, el aumento del coste (malas noticias), se propagan muy lentamente.

Si un link se rompe (el coste se hace infinito), el resto de routers deberían ser conscientes de forma inmediata, pero en routing vector distancias, puede llevar bastante tiempo.

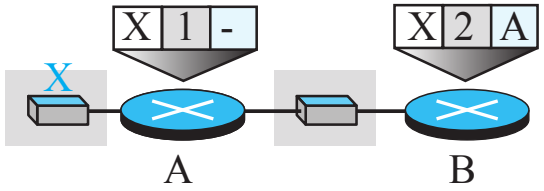
El problema se conoce como **conteo al infinito**.

Algunas veces, puede necesitar varias actualizaciones antes de que el coste del enlace roto sea registrado como infinito por el resto de routers.

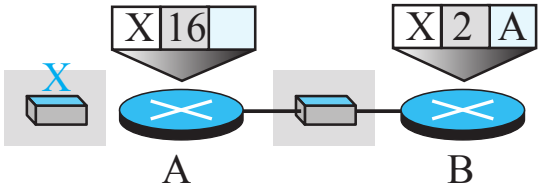
Bucle entre dos nodos (conteo al infinito)

Supongamos:

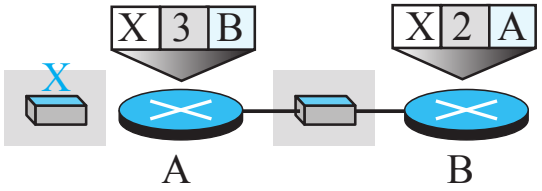
- Sistema formado por tres nodos (*sólo se muestran los nodos involucrados*).
- Todos los costes establecidos a 1.



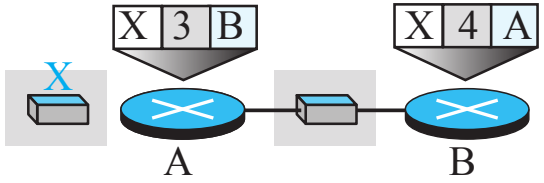
(a) Antes de fallo



(b) Después de fallo de enlace



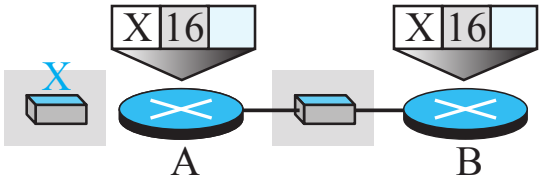
(c) Después A es actualizado por B



(d) Después de B es actualizado por A

• • •

[*Forouzan*]



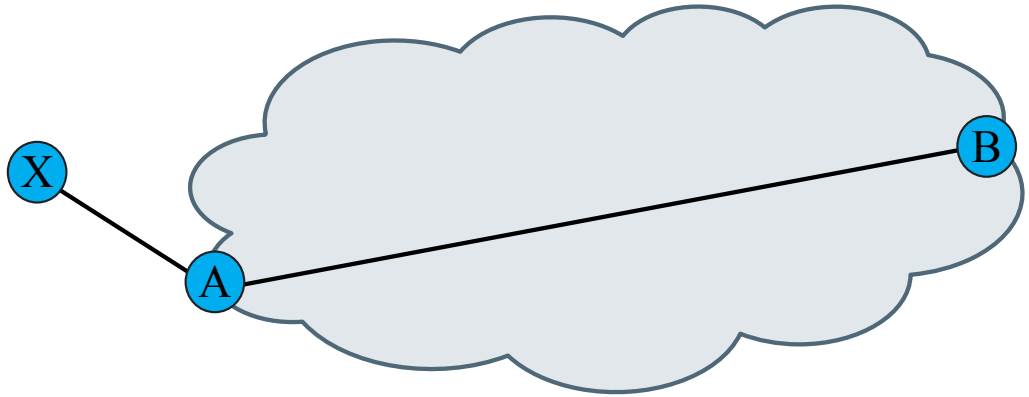
(e) Finalmente

(a) Al inicio, ambos nodos, **A** y **B**, saben cómo alcanzar **X**.
El enlace entre **A** y **X**, falla.
(b) **A** cambia su tabla, estableciendo el coste infinito hasta **X** (valor 16).
Si **A** enviara su tabla inmediatamente a **B**, no habría problema.
Sin embargo, el sistema se vuelve inestable si **B** envía su tabla de reenvío a **A**, antes de recibir la de **A**.
(c) **A** recibe la actualización y asume que **B** sabe cómo alcanzar **X**, por lo que, actualiza su tabla.
(d) **A** envía su actualización a **B**. **B** cree que algo ha cambiado, por lo que actualiza su tabla.
El coste de alcanzar **X** se incrementa hasta que alcanza infinito.
(e) En este punto, **A** y **B**, saben que **X** es inalcanzable, pero durante este tiempo, el sistema ha estado inestable.

Horizonte dividido

Una solución a la inestabilidad: **horizonte dividido (*split horizon*)**.

- En lugar de inundar la red a través de todas las interfaces, cada nodo sólo envía una parte de la tabla a través de cada interface.
- Si, a partir de su tabla, **B** cree que la ruta óptima para alcanzar **X** es vía **A**, no necesita anunciar este trozo de información a **A** (ya la debe conocer).



- Por lo tanto, **B** elimina la última línea de su tabla de reenvío antes de enviarla a **A**.
- En este caso, **A** mantiene su valor de infinito como distancia a **X**.
- Posteriormente, cuando **A** envía su tabla de información a **B**, éste también corrige su tabla de reenvío, estableciendo infinito.
- El sistema se estabiliza después de la primera actualización: ambos nodos saben que **X** es inalcanzable.

Reverso envenenado

El mecanismo de horizonte dividido tiene un inconveniente: *normalmente, los protocolos utilizan timers. Si un nodo no recibe noticias de un enrutador, en un tiempo determinado, el nodo elimina el enrutador de su tabla..*

Cuando el nodo **B** elimina la ruta a **X** de su anuncio a **A**, éste no puede determinar si es debido a la estrategia de horizonte dividido (la fuente de la información era **A**) o porque **B** no ha recibido ninguna noticia sobre **X**, recientemente.

Mediante la estrategia **reverso envenenado**, **B** todavía puede anunciar el valor de **X**, pero si la fuente de la información es **A**, entonces reemplaza la información de distancia con infinito con un aviso (marcado):

«No utilizar este valor; que esta ruta vino de ti».

La inestabilidad producida por el bucle entre dos nodos se puede evitar utilizando horizonte dividido junto con reverso envenenado.

Sin embargo, si se produce inestabilidad entre tres nodos, no se puede garantizar la convergencia.

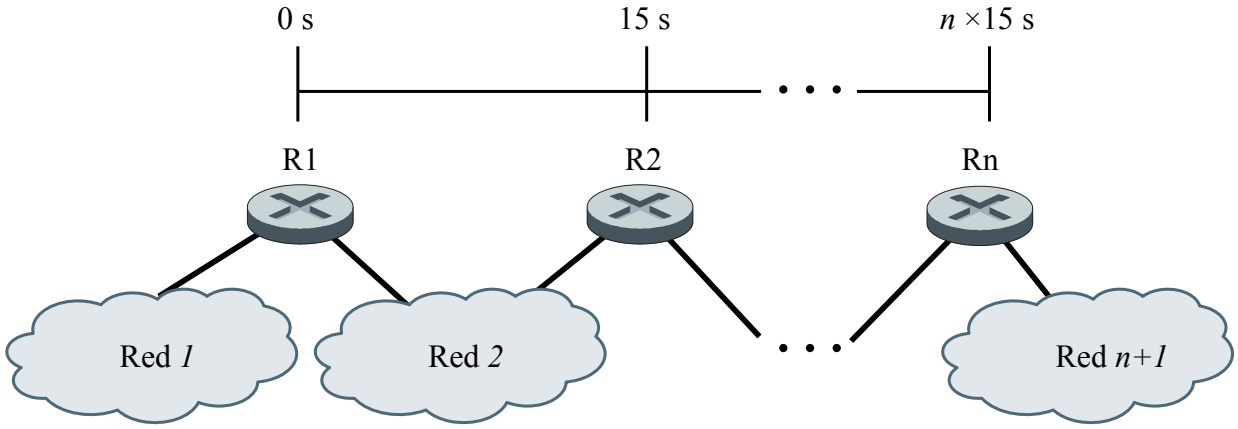
Protocolos Vector Distancias: resumen

Ventajas:

- Fácil implementación.

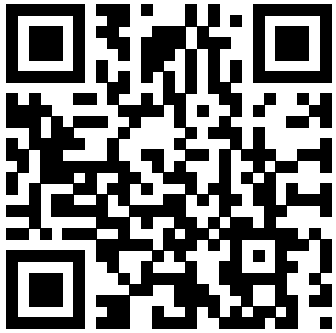
Inconvenientes:

- El tiempo de convergencia crece proporcionalmente al número de nodos ⇒ Baja escalabilidad.



- Limitado por el nodo de menor capacidad.
- Posibilidad de bucles.
- Inestabilidad en redes grandes (conteo al infinito).

RIP (*Routing Information Protocol*)



Routing Information Protocol (RIP)

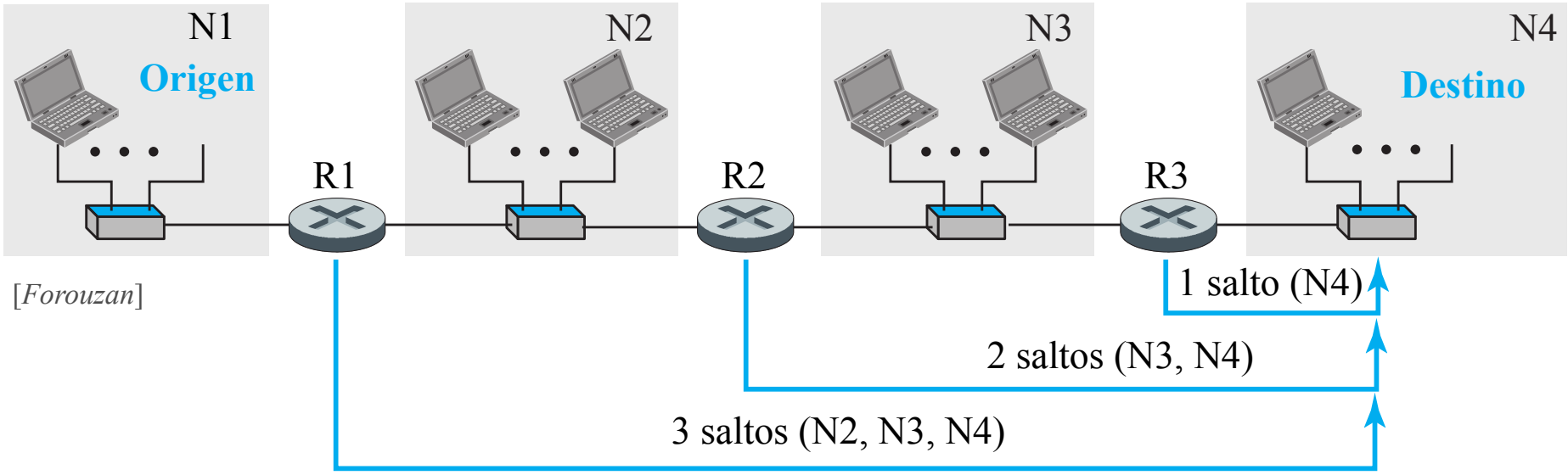
Routing Information Protocol (RIP) es uno de los protocolos más extendidos.

Basado en el algoritmo de **enrutamiento por vector distancia**.

RIP es un protocolo de enrutamiento intradominio.

Iniciado por Xerox Network System (XNS), posteriormente añadido a la distribución de UNIX Berkeley Software Distribution (BSD), que contribuyó a su difusión.

RIP: contador de saltos



Los routers RIP anuncian el coste de alcanzar las diferentes redes.

El coste se define entre un router y la red en la que está localizado el host.

Para simplificar la implementación, el coste se define como el **número de saltos** (número de redes que el paquete debe cruzar desde el router origen hasta el router destino).

La red en la que el host origen está conectado no cuenta ya que el host no utiliza tabla de reenvío, los paquetes son enviados al router por defecto (gateway).

El máximo coste permitido en RIP son 15 saltos, de forma que 16 es considerado ∞ (los paquetes son eliminados).

RIP: tablas de reenvío

Tabla re-envío para R1

Red Destino	Siguiente router	Coste en saltos
N1	——	1
N2	——	1
N3	R2	2
N4	R2	3

Tabla re-envío para R2

Red Destino	Siguiente router	Coste en saltos
N1	R1	2
N2	——	1
N3	——	1
N4	R3	2

Tabla re-envío para R3

Red Destino	Siguiente router	Coste en saltos
N1	R2	3
N2	R2	2
N3	——	1
N4	——	1

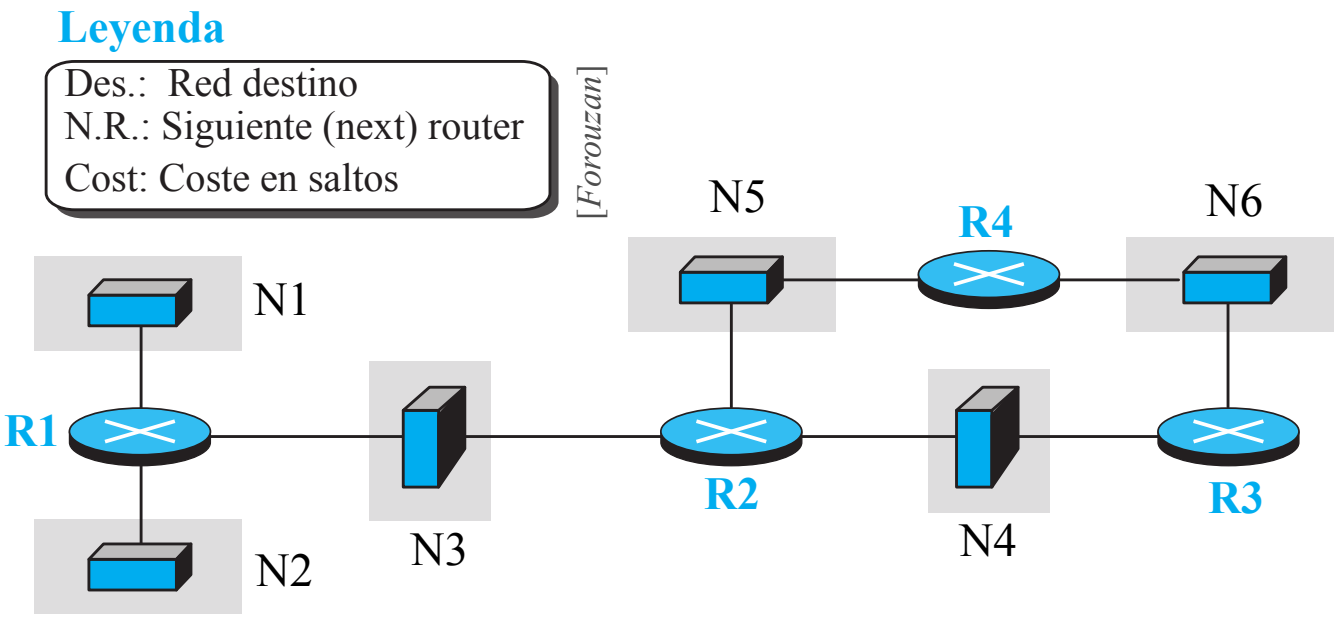
[*Forouzan*]

Incluye la misma información que routing por vector distancias, pero con alguna diferencia:

- El coste se establece en saltos (coste = 1).
- RIP define el siguiente router de forma que proporciona información sobre la totalidad del árbol de mínimo coste.
- Por ejemplo, para alcanzar N4 desde R1:
  - R1 define que el siguiente router para el camino a N4 es R2,
  - R2 define que el siguiente router a N4 es R3, y
  - R3 define que no hay siguiente router para esta ruta:

$$R1 \rightarrow R2 \rightarrow R3 \rightarrow N4$$

RIP: ejemplo (situación inicial)



R1

Des.	N. R.	Cost
N1	———	1
N2	———	1
N3	———	1

R2

Des.	N. R.	Cost
N3	———	1
N4	———	1
N5	———	1

R3

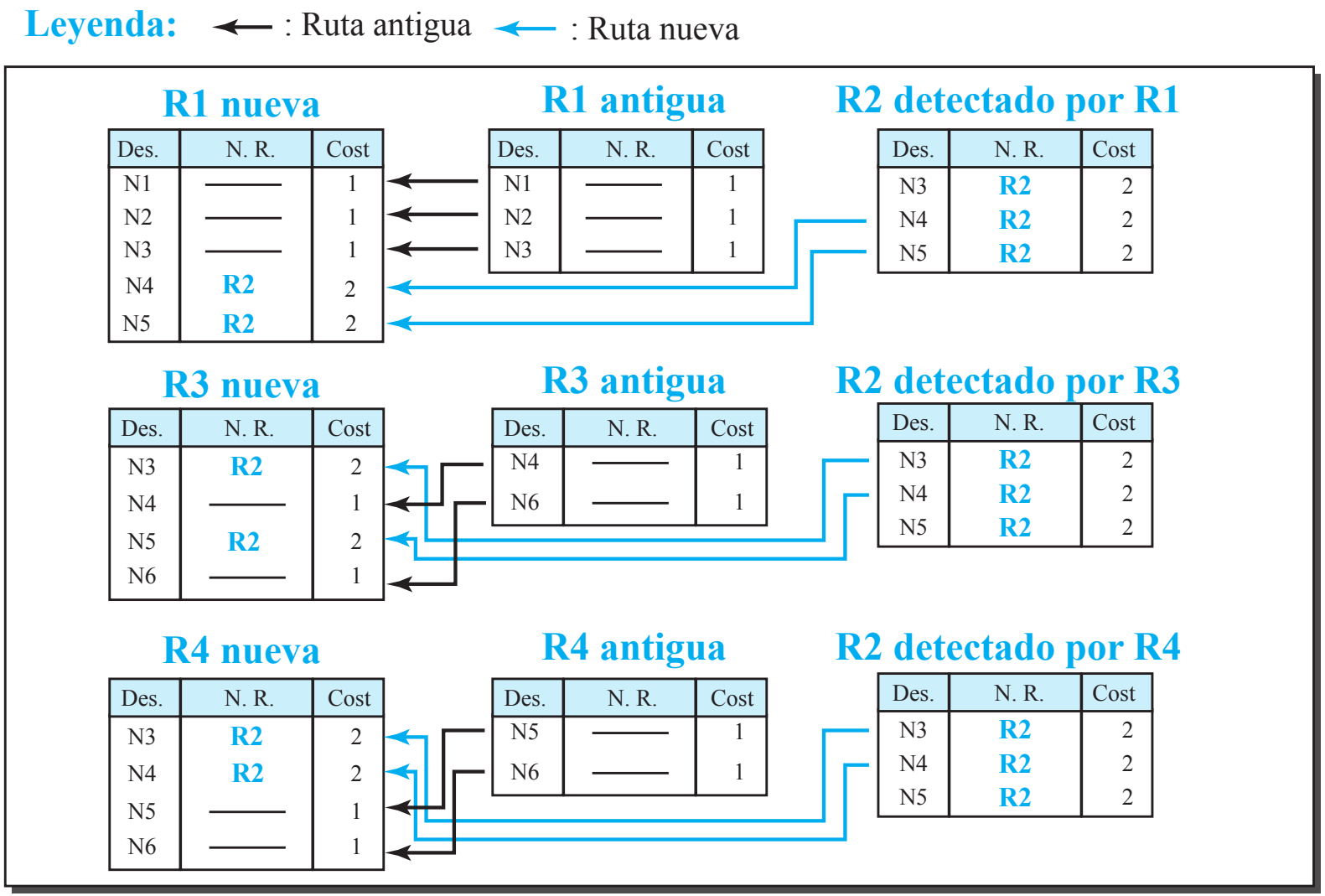
Des.	N. R.	Cost
N4	———	1
N6	———	1

R4

Des.	N. R.	Cost
N5	———	1
N6	———	1

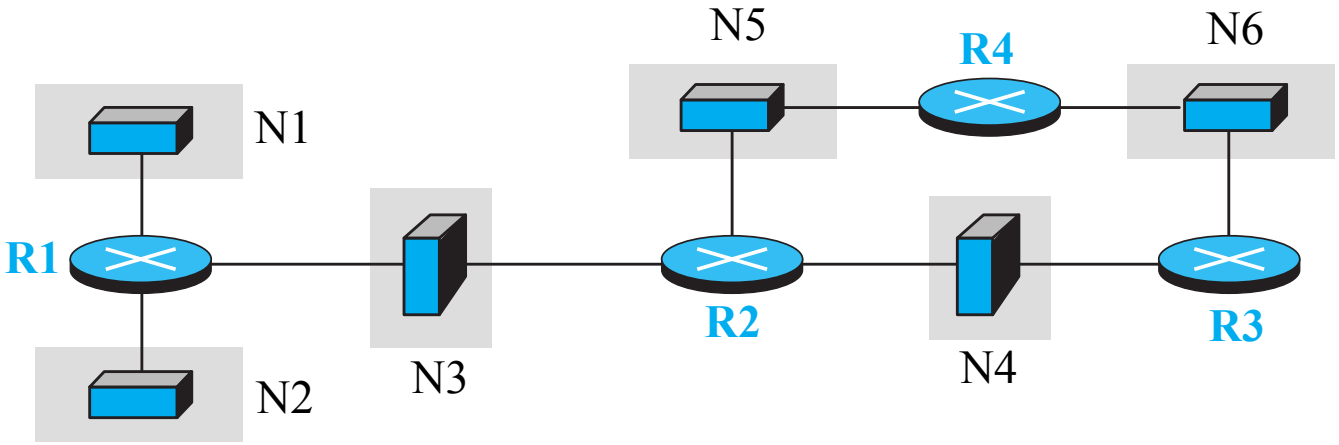
Tablas de re-envío (forwarding) después del inicio de los routers

RIP: ejemplo (actualización de tablas)



Cambios en las tablas de re-envío de los routers R1, R3 y R4 después de recibir una copia de la tabla de R2.

RIP: ejemplo (convergencia)



R1 final

Des.	N. R.	Cost
N1	———	1
N2	———	1
N3	———	1
N4	R2	2
N5	R2	2
N6	R2	3

R2 final

Des.	N. R.	Cost
N1	R1	2
N2	R1	2
N3	———	1
N4	———	1
N5	———	1
N6	R3	2

R3 final

Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	———	1
N5	R2	2
N6	———	1

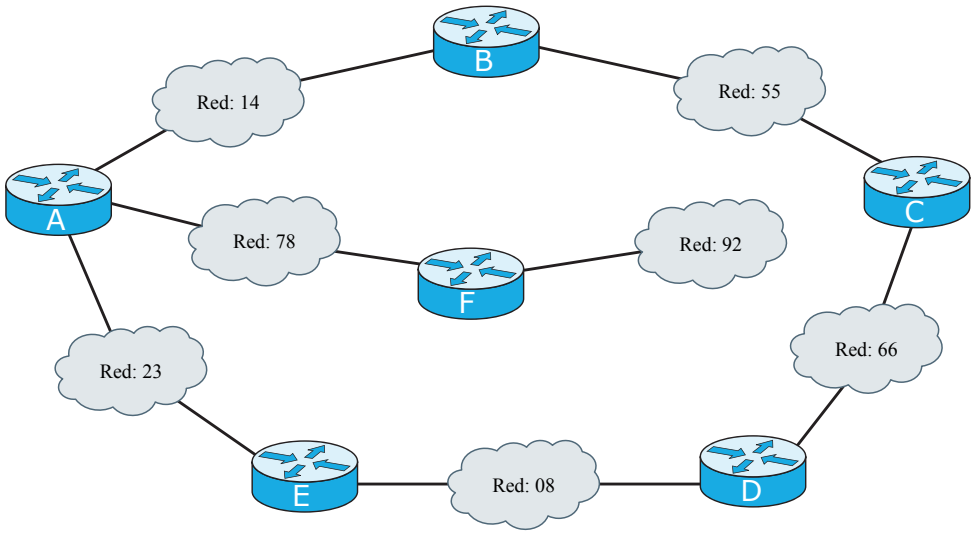
R4 final

Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	R2	2
N5	———	1
N6	———	1

Tablas de re-envío de todos los routers una vez han sido estabilizados

Algoritmo vector distancias: convergencia rápida

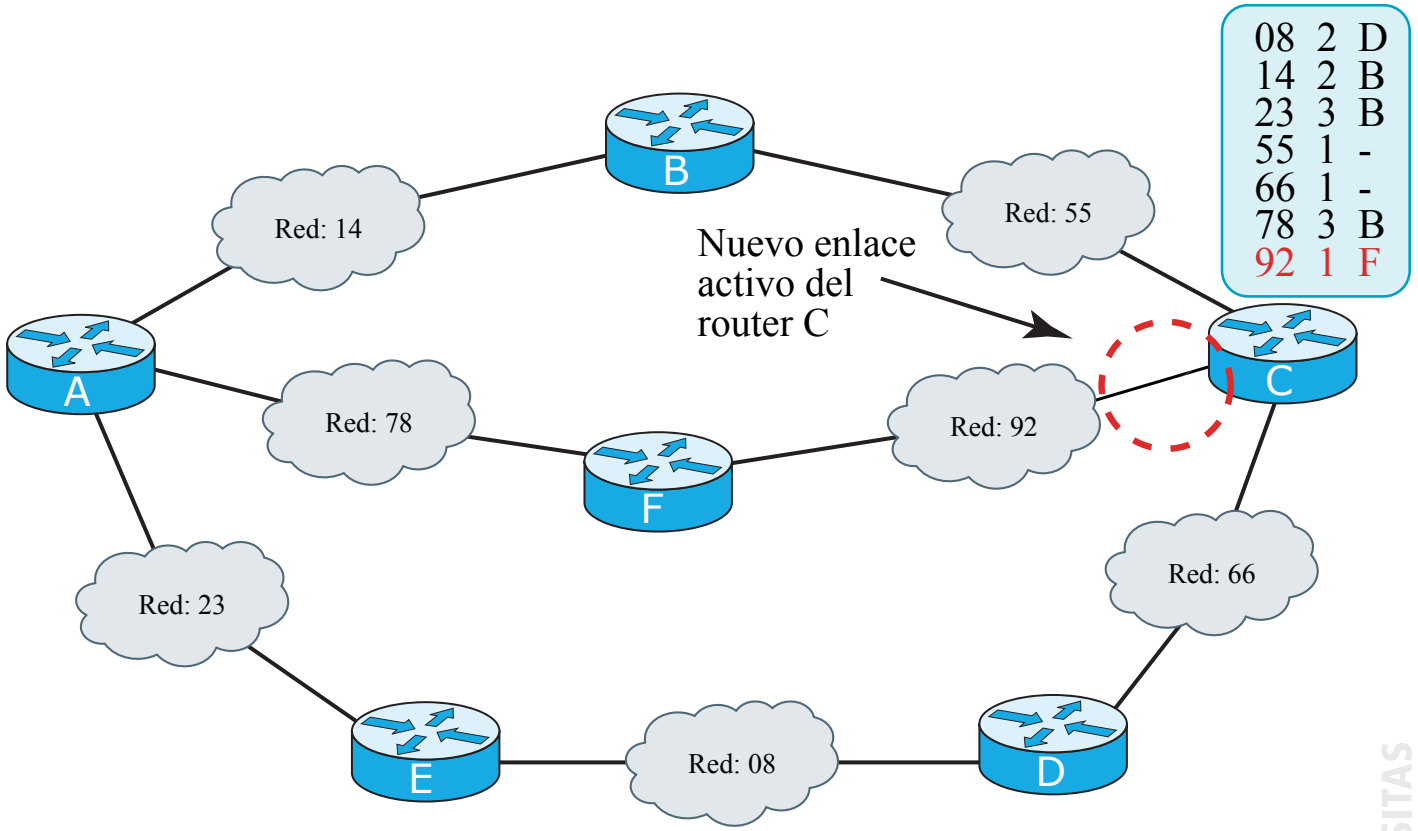
A	14 1 - 23 1 - 78 1 -	08 2 E 14 1 - 23 1 - 55 2 B 78 1 - 92 2 F	08 2 E 14 1 - 23 1 - 55 2 B 66 3 E 78 1 - 92 2 F	08 2 E 14 1 - 23 1 - 55 2 B 66 3 E 78 1 - 92 2 F	08 2 E 14 1 - 23 1 - 55 2 B 66 3 E 78 1 - 92 2 F	08 2 E 14 1 - 23 1 - 55 2 B 66 3 E 78 1 - 92 2 F
B	14 1 - 55 1 -	14 1 - 23 2 A 55 1 - 66 2 C 78 2 A	08 3 A 14 1 - 23 2 A 55 1 - 66 2 C 78 2 A 92 3 A	08 3 A 14 1 - 23 2 A 55 1 - 66 2 C 78 2 A 92 3 A	08 3 A 14 1 - 23 2 A 55 1 - 66 2 C 78 2 A 92 3 A	08 3 A 14 1 - 23 2 A 55 1 - 66 2 C 78 2 A 92 3 A
C	55 1 - 66 1 -	08 2 D 14 2 B 55 1 - 66 1 -	08 2 D 14 2 B 23 3 B 55 1 - 66 1 - 78 3 B	08 2 D 14 2 B 23 3 B 55 1 - 66 1 - 78 3 B 92 4 B	08 2 D 14 2 B 23 3 B 55 1 - 66 1 - 78 3 B 92 4 B	08 2 D 14 2 B 23 3 B 55 1 - 66 1 - 78 3 B 92 4 B
D	08 1 - 66 1 -	08 1 - 23 2 E 55 2 C 66 1 -	08 1 - 14 3 C 23 2 E 55 2 C 66 1 - 78 3 E	08 1 - 14 3 C 23 2 E 55 2 C 66 1 - 78 3 E	08 1 - 14 3 C 23 2 E 55 2 C 66 1 - 78 3 E 92 4 E	08 1 - 14 3 C 23 2 E 55 2 C 66 1 - 78 3 E 92 4 E
E	08 1 - 23 1 -	08 1 - 14 2 A 23 1 - 66 2 D 78 2 A	08 1 - 14 2 A 23 1 - 55 3 A 66 2 D 78 2 A	08 1 - 14 2 A 23 1 - 55 3 A 66 2 D 78 2 A 92 3 A	08 1 - 14 2 A 23 1 - 55 3 A 66 2 D 78 2 A 92 3 A	08 1 - 14 2 A 23 1 - 55 3 A 66 2 D 78 2 A 92 3 A
F	78 1 - 92 1 -	14 2 A 23 2 A 78 1 - 92 1 -	08 3 A 14 2 A 23 2 A 55 3 A 78 1 - 92 1 -	08 3 A 14 2 A 23 2 A 55 3 A 66 4 A 78 1 - 92 1 -	08 3 A 14 2 A 23 2 A 55 3 A 66 4 A 78 1 - 92 1 -	08 3 A 14 2 A 23 2 A 55 3 A 66 4 A 78 1 - 92 1 -



El algoritmo de enrutamiento vector distancias converge rápidamente.

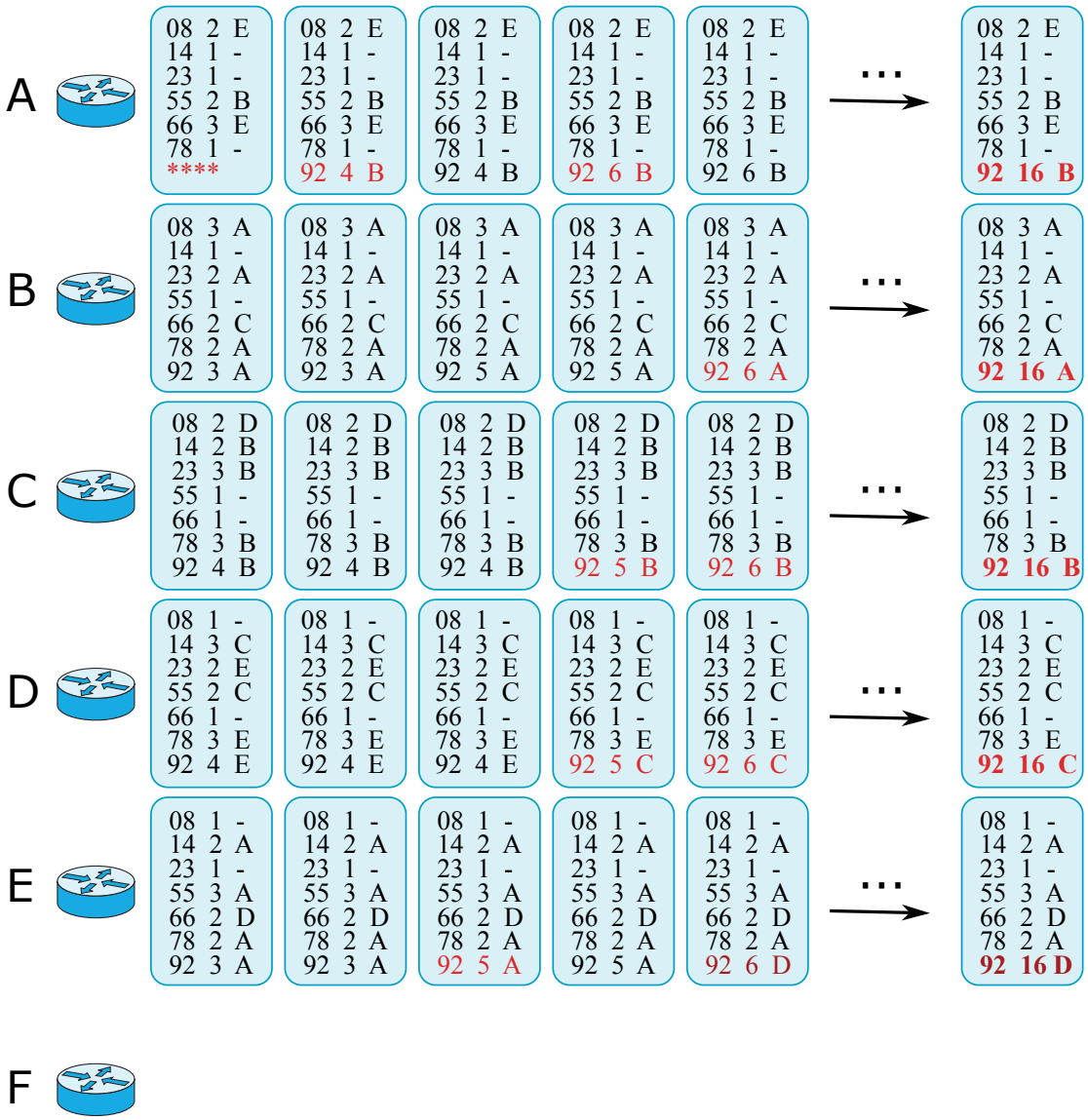
Respuesta rápida ante buenas noticias

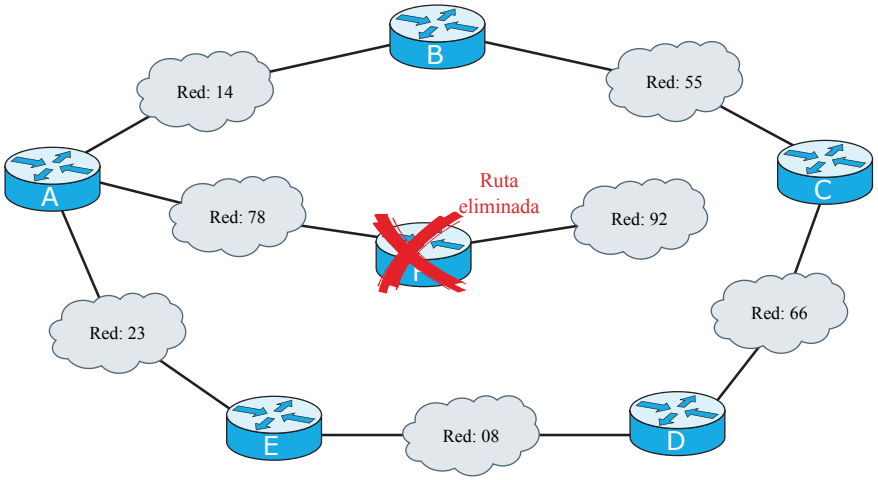
A	08 2 E 14 1 - 23 1 - 55 2 B 66 3 E 78 1 - 92 2 F	08 2 E 14 1 - 23 1 - 55 2 B 66 3 E 78 1 - 92 2 F	08 2 E 14 1 - 23 1 - 55 2 B 66 3 E 78 1 - 92 2 F
B	08 3 A 14 1 - 23 2 A 55 1 - 66 2 C 78 2 A 92 3 A	08 3 A 14 1 - 23 2 A 55 1 - 66 2 C 78 2 A 92 2 C	08 3 A 14 1 - 23 2 A 55 1 - 66 2 C 78 2 A **92 2 C**
C	08 2 D 14 2 B 23 3 B 55 1 - 66 1 - 78 3 B 92 1 F	08 2 D 14 2 B 23 3 B 55 1 - 66 1 - 78 2 F 92 1 F	08 2 D 14 2 B 23 3 B 55 1 - 66 1 - **78 2 F** 92 1 F
D	08 1 - 14 3 C 23 2 E 55 2 C 66 1 - 78 3 E 92 4 E	08 1 - 14 3 C 23 2 E 55 2 C 66 1 - 78 3 E 92 2 C	08 1 - 14 3 C 23 2 E 55 2 C 66 1 - 78 3 E **92 2 C**
E	08 1 - 14 2 A 23 1 - 55 3 A 66 2 D 78 2 A 92 3 A	08 1 - 14 2 A 23 1 - 55 3 A 66 2 D 78 2 A 92 3 A	08 1 - 14 2 A 23 1 - 55 3 A 66 2 D 78 2 A 92 3 A
F	08 3 A 14 2 A 23 2 A 55 3 A 66 4 A 78 1 - 92 1 C	08 3 A 14 2 A 23 2 A 55 2 C 66 2 C 78 1 - 92 1 C	08 3 A 14 2 A 23 2 A **55 2 C** **66 2 C** 78 1 - 92 1 C



Respuesta rápida ante una buena noticia (enlace nuevo).

Conteo al infinito





Conteo al infinito ante mala noticia (ruta inaccesible).

RIP: resumen

Utiliza temporizadores para el envío periódico de tablas de reenvío (25-35 s).

Tiempos de expiración para invalidar rutas (180 s).

Coste máximo 15 saltos → 16 se considera infinito.

Mensajes de actualización de rutas que generan muy poco tráfico.

RIP utiliza algoritmo vector distancias, por lo que es lento para dominios grandes.

Debido a la convergencia lenta, se producen bucles y conteo al infinito. Para solucionarlo, se utiliza horizonte dividido y reverso envenenado.

En relación a la robustez, por el propio mecanismo de propagación de rutas entre vecinos, es posible que problemas de datos corruptos se propague entre vecinos.

Enrutamiento estado del enlace

Enrutamiento por estado del enlace

Enrutamiento por estado del enlace (*Link state routing*) continúa con los principios discutidos anteriormente para la creación de:

- Arbol de coste mínimo
- Tablas de reenvío

Utiliza el término **estado-del-enlace** para determinar la característica de un enlace (arco) que representa una red en la interred.

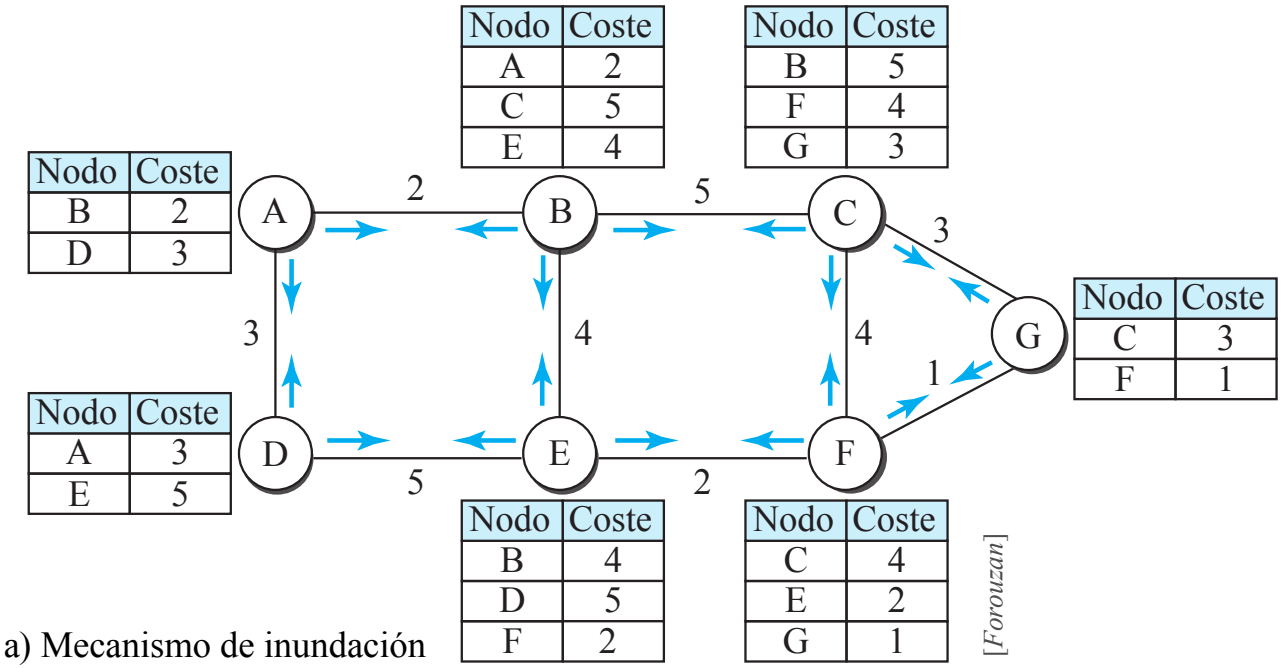
El coste asociado con un arco define el estado del enlace.

Enlaces con coste menor son preferidos a enlaces con costes mayores.

Si el coste es infinito, indica que el enlace no existe o ha sido roto.

Una vez completado el grafo, se obtiene el árbol de camino más corto mediante Dijkstra o Bellman-Ford.

LSDB



a) Mecanismo de inundación

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b) Base de datos de estado del enlace (LSDB)

Inundación: cada nodo envía mensajes LSP, *Link state packet*, a sus vecinos con la siguiente información:

- La identidad del nodo.
- El coste del enlace.

Cada nodo compara el LSP recibido con la información que dispone y actualiza su tabla.

Posteriormente reenvía el LSP por todos interfaces.

Cuando la red se estabiliza, se obtiene el LSDB, *link-state database*, que es el mismo para todos los nodos y muestra el mapa completo de la interred (**topología**).

En definitiva, cada nodo puede construir el mapa de la red a partir del LSDB.

Protocolos Estado del enlace: características principales

Ventajas:

- Flexibilidad y optimización en la definición de la ruta, ya que se dispone de un mapa completo de la topología de red.
- El LSP no se envía periódicamente, sólo cuando se producen cambios en la topología.
- Todos los nodos adquieren, de forma rápida, los cambios en la topología de red.

Inconvenientes:

- Es necesario un protocolo de señalización para mantener la información de topología (*Hello*).
- Necesita inundación (consumo de recursos).
- Los LSP tienen que ser confirmados.
- Difícil implementación.

Algoritmo Bellman-Ford

Algoritmo Bellman-Ford (I)

Suposiciones:

- Pesos positivos y negativos.
- Ciclos no negativos.

Objetivo:

- Encontrar el camino más corto de una fuente hasta el resto de nodos.

Variables:

$D_i^{(h)}$, longitud del camino más corto desde la fuente (suponiendo que es el nodo 1) y el nodo i con número de saltos (*hops*) $\leq h$.

Inicialización:

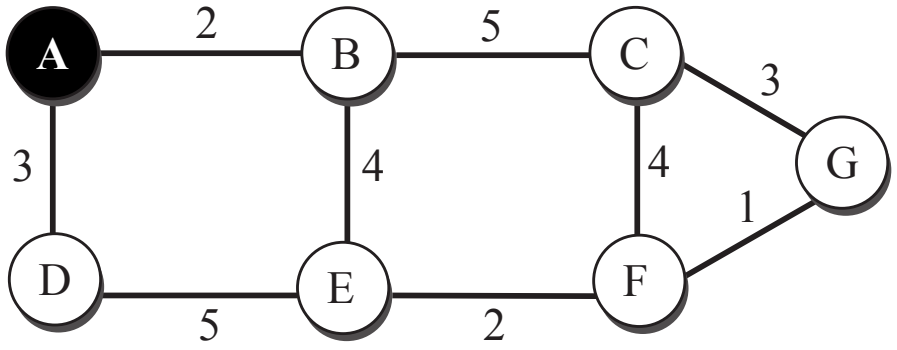
$$D_1^{(h)} = 0 \quad \forall h$$ Distancia a sí mismo, es 0

$$D_i^{(0)} = \infty \quad \forall h \neq 1$$ A salto 0, la distancia al resto de nodos, es ∞

Iteración:

$$D_i^{(h+1)} = \text{mín}\left[D_i^{(h)}, \min_{\forall j}\left(D_j^{(h)} + d_{ji}\right)\right]$$

Algoritmo Bellman-Ford: nodo A (tabla)



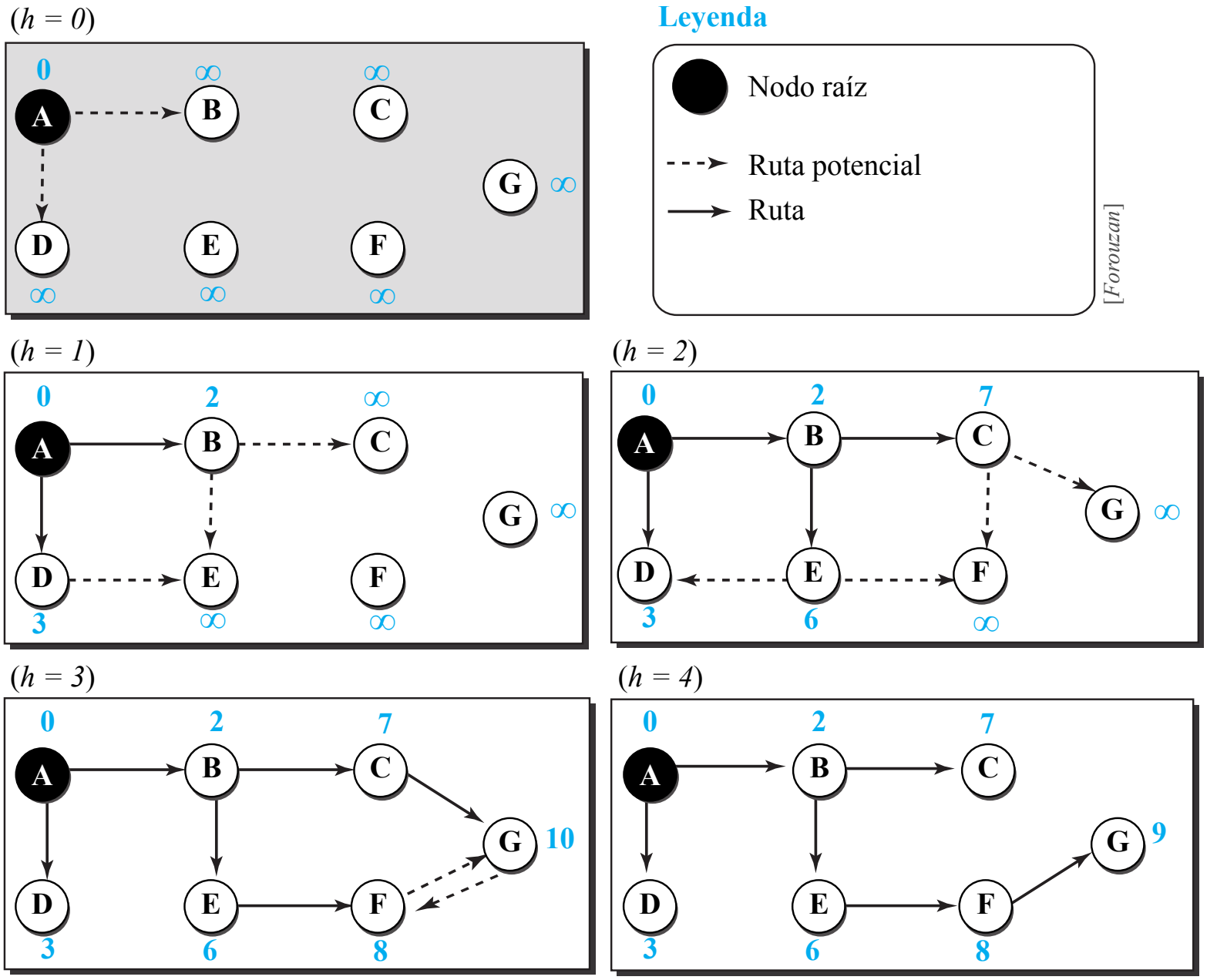
Grafo ponderado

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

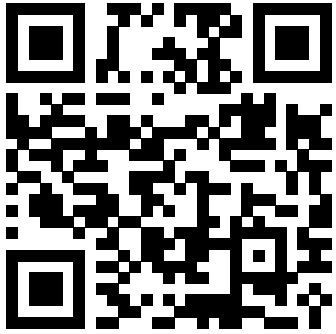
Base de datos de estado del enlace (LSDB)

h	$L_h(B)$	Path	$L_h(C)$	Path	$L_h(D)$	Path	$L_h(E)$	Path	$L_h(F)$	Path	$L_h(G)$	Path
0	∞	-	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	AB	∞	-	3	AD	∞	-	∞	-	∞	-
2	2	AB	7	ABC	3	AD	6	ABE	∞	-	∞	-
3	2	AB	7	ABC	3	AD	6	ABE	8	ABEF	10	ABCG
4	2	AB	7	ABC	3	AD	6	ABE	8	ABEF	9	ABEFG

Algoritmo Bellman-Ford: nodo A (grafo)



Algoritmo Dijkstra



Algoritmo Dijkstra

Suposiciones:

- Pesos positivos.

Objetivo:

- Encontrar el camino más corto de una fuente (1) hasta el resto de los otros nodos.

Inicialización:

$$P = \{1\},$$

$$D_1 = 0, \quad D_j^{(0)} = d_{1j} \quad \forall j \neq 1$$

$d_{ij} = \infty$ si el arco $i - j$ no existe.

Iteración:

1. Encontrar $i \in (N - P)$:

$$D_i = \min_{j \in (N-P)} D_j$$

y establecer:

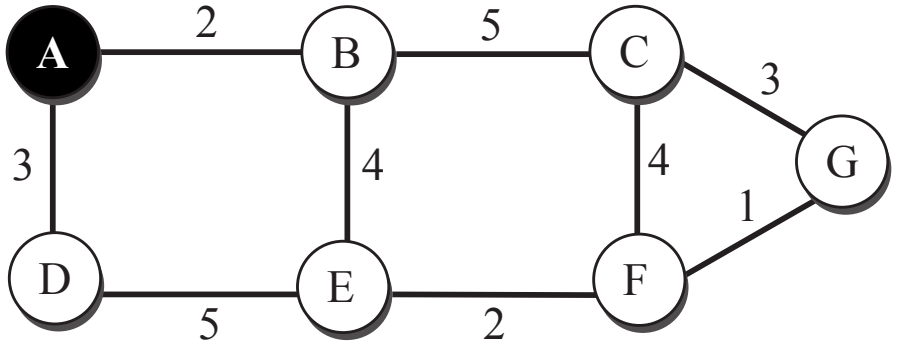
$$P := P \cup \{i\}. \text{ Si } P = N, \text{ entonces } \textit{STOP}.$$

2. Para cada $j \in (N - P)$ vecino de cualquier nodo en P, establecer:

$$D_j = \min \left[D_j, \min_k \left(D_k + d_{kj} \right) \right]$$

3. Ir a Paso 1.

Algoritmo de Dijkstra: nodo A (tabla)



Grafo ponderado

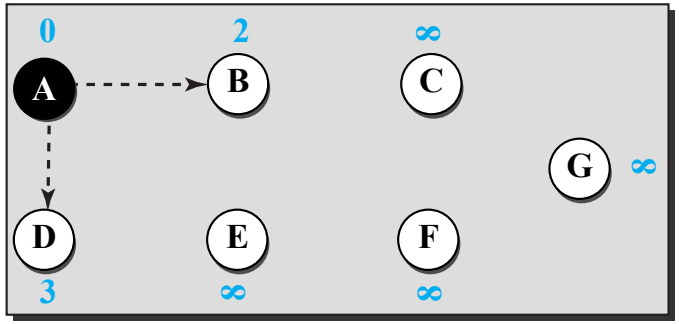
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

Base de datos de estado del enlace (LSDB)

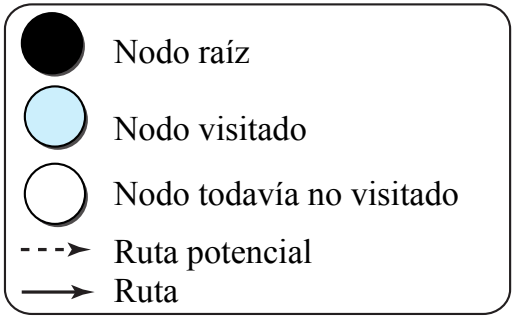
I	T	L(B)	Path	L(C)	Path	L(D)	Path	L(E)	Path	L(F)	Path	L(G)	Path
Inic.	{A}	2	AB	∞	-	3	AD	∞	-	∞	-	∞	-
1	{AB}	2	AB	7	ABC	3	AD	6	ABE	∞	-	∞	-
2	{ABD}	2	AB	7	ABC	3	AD	6	ABE	∞	-	∞	-
3	{ABDE}	2	AB	7	ABC	3	AD	6	ABE	8	ABEF	∞	-
4	{ABDEC}	2	AB	7	ABC	3	AD	6	ABE	8	ABEF	10	ABCG
5	{ABDECF}	2	AB	7	ABC	3	AD	6	ABE	8	ABEF	9	ABEFG
6	{ABDECFG}	2	AB	7	ABC	3	AD	6	ABE	8	ABEF	9	ABEFG

Algoritmo de Dijkstra: nodo A (grafo)

Inicialización

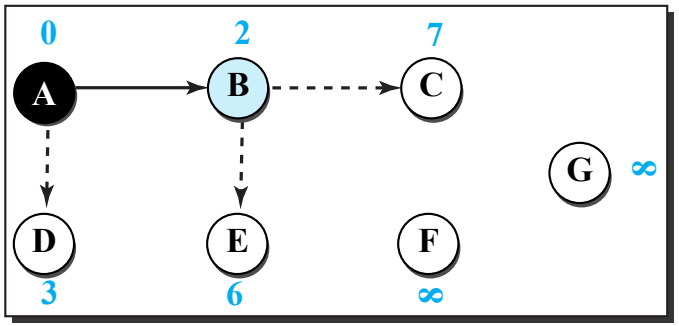


Leyenda

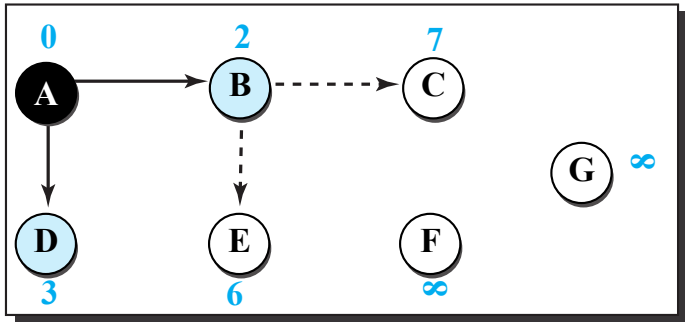


[*Forouzan*]

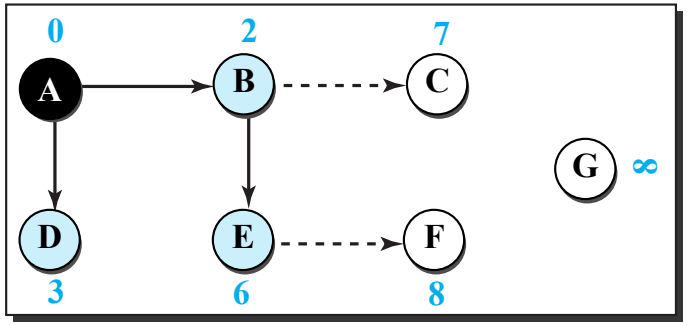
Iteración 1



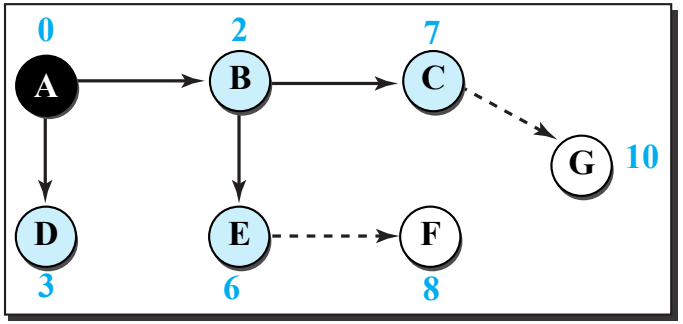
Iteración 2



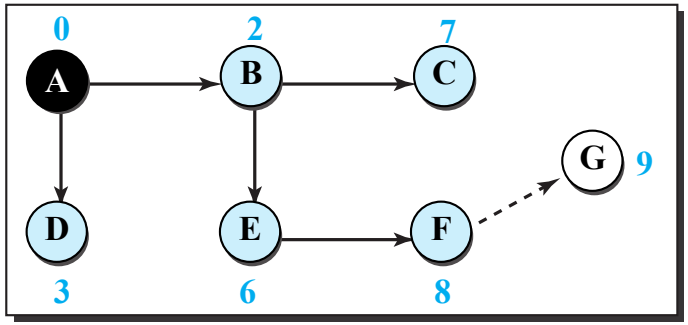
Iteración 3



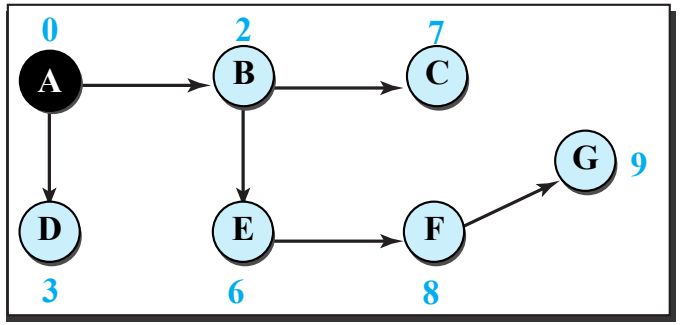
Iteración 4



Iteración 5



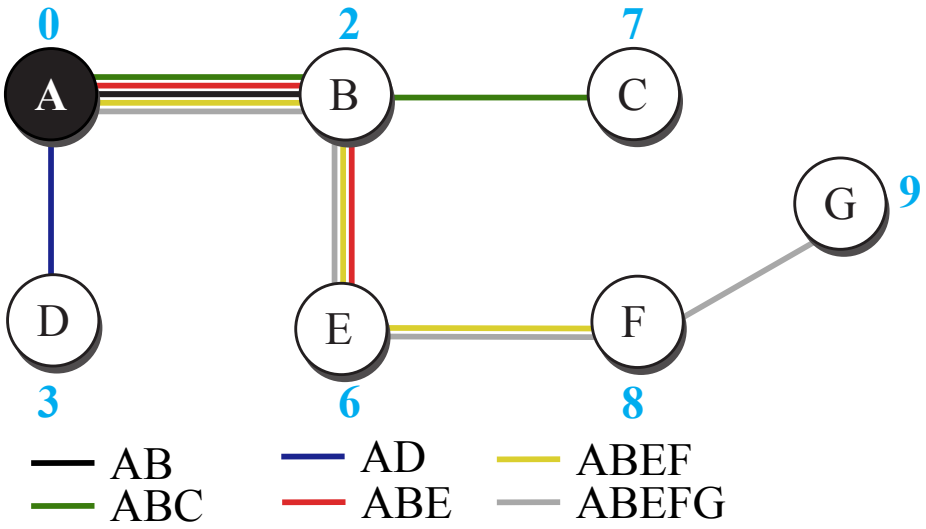
Iteración 6



Árbol de mínimo coste

Al finalizar los algoritmos, tanto Dijkstra como Bellman-Ford, la solución obtenida corresponde al árbol compuesto por los **árboles de mínimo coste**, en este caso, desde A, hasta el resto de nodos de la red.

Para $N = 7$ nodos, se obtienen 6 árboles de coste mínimo:



- Indica el coste acumulado desde el nodo A hasta el resto de nodos.
- A la vista del árbol, se concluye que, los caminos `DE`, `CF` y `CG`, nunca serán utilizados.
- Además, se obtiene información adicional: trayectorias, necesidades de ancho de banda, requisitos de vecinos, etc.
- Evidentemente, ambos algoritmos proporcionan el mismo resultado.

OSPF (*Open Shortest Path First*)

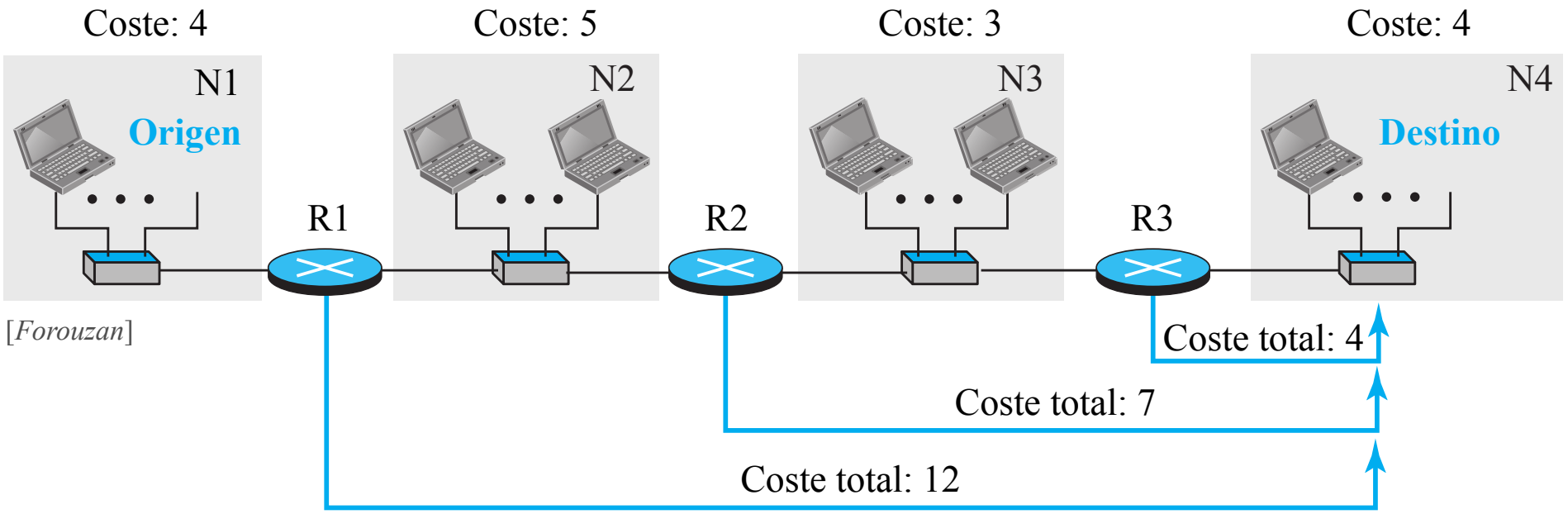
Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) es un protocolo de enrutamiento intradominio.

Basado en el algoritmo de **enrutamiento por estado del enlace**.

Es un protocolo abierto por lo que la especificación es un documento público.

OSPF: métrica



Igual que en RIP, el coste de alcanzar un destino desde un host es calculado desde el router origen hasta la red destino.

Cada enlace (red) tiene un **peso** asignado: throughput, RTT, seguridad, prioridad, coste, etc.

También se pueden aplicar criterios de saltos.

OSPF: tablas de reenvío

Tabla re-envío para R1

Red Destino	Siguiente router	Coste
N1	——	4
N2	——	5
N3	R2	8
N4	R2	12

Tabla re-envío para R2

Red Destino	Siguiente router	Coste
N1	R1	9
N2	——	5
N3	——	3
N4	R3	7

Tabla re-envío para R3

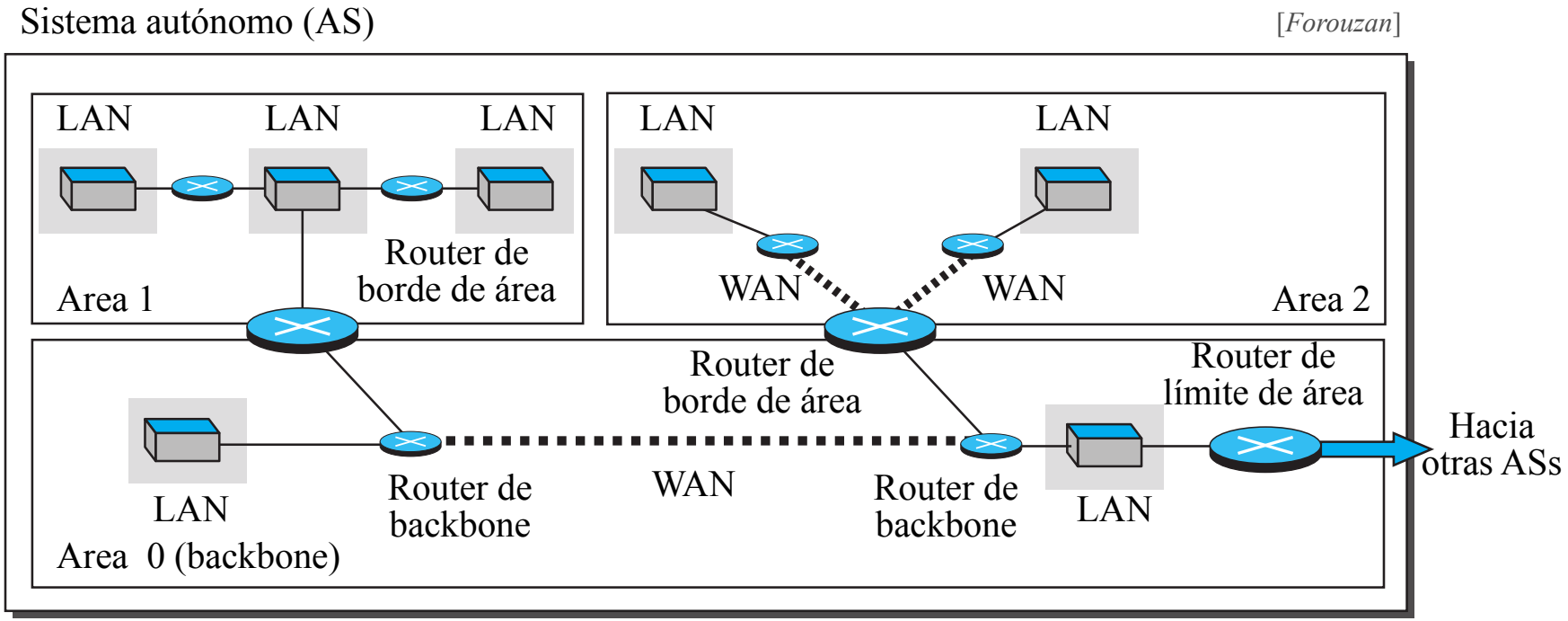
Red Destino	Siguiente router	Coste
N1	R2	12
N2	R2	8
N3	——	3
N4	——	4

[*Forouzan*]

OSPF crea una tabla de reenvío a partir de encontrar el árbol de camino más corto entre él y el destino, utilizando el **algoritmo de Dijkstra**.

Similar a las tablas de RIP, pero con un valor de *coste*.

OSPF: áreas



OSPF, a diferencia de RIP, es apropiado para el enrutamiento en sistemas más grandes.

En OSPF, todos los routers inundan con sus LSP para crear la LSDB global, lo que puede generar problemas de tráfico.

Para evitar saturación por tráfico, se organiza en **áreas**, que actúan como dominios independientes con sus propias inundaciones de LSP.

Cada router sólo conoce información de su área.

El backbone es el responsable de interconectar las áreas y pasar información entre ellas. Por defecto, es el sistema 0.

OSPF: resumen

Actualización de mensajes:

- Los mensajes de estado del enlace en OSPF tienen un formato más complejo que RIP.
- Utilizan un mecanismo de inundación.
- Si se trata de sistemas grandes, los mensajes pueden provocar problemas de tráfico y utilizar un gran ancho de banda.

Convergencia de las tablas de reenvío:

- Una vez se ha realizado la inundación de LSPs, cada router puede crear su propio árbol de camino más corto y tabla de reenvío.
- La convergencia es bastante rápida.
- No obstante, cada router necesita ejecutar el algoritmo de Dijkstra, donde el coste temporal puede ser significativo.

Robustez:

- OSPF es más **robusto** que RIP, ya que una vez recibida por completo la LSDB, cada router es independiente y no depende del resto de routers del área.
- Fallos en routers adyacentes no afectan a otros routers, como sucede en RIP.

Cuestiones de revisión

El estudiante debería ser capaz de responder a las siguientes cuestiones:

¿Cómo se modela un servicio orientado/no orientado a la conexión?

¿Cómo funciona la conmutación de paquetes frente a conmutación de circuitos?

¿Cómo se gestiona un espacio de direcciones IPv4?

¿Cómo se realiza el subnetting en IPv4, tanto fijo, como variable?

¿Cómo se construye una tabla de reenvío y cómo la utilizan los routers?

¿Cómo se realiza la agregación de rutas?

¿Cómo se realiza la fragmentación a nivel red?

¿Cómo funciona y para qué se utilizan ARP y Ping?

¿Para qué se utilizan la dirección física (MAC) y lógica (IP)?

¿Cómo funciona el enrutamiento por vector distancias?

¿Cómo funciona el enrutamiento por estado del enlace?

¿Cómo se aplican los algoritmos Dijkstra y Bellman-Ford, así como las diferencias entre ambos?

¿Cómo se obtiene el árbol de coste mínimo?

UNIVERSITAS
Miguel Hernández