# Capítulo 4: la ruta

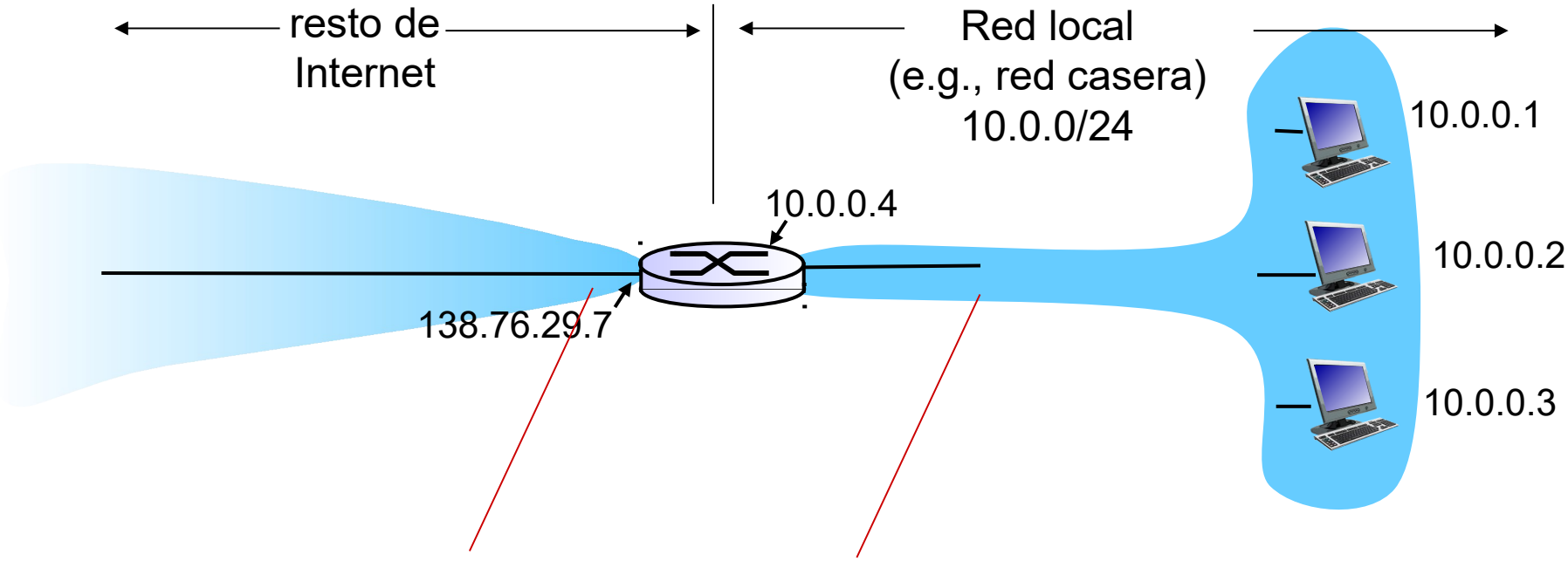
4.1 Introducción
4.2 Circuito virtual y redes de datagramas
4.3 ¿Qué hay dentro de un enrutador?
4.4 IP: Protocolo de Internet
- Formato del datagrama
- Direccionamiento IPv4
- Protocolo ICMP
- Direccionamiento IPv6
- NAT

4.5 Algoritmos de enrutamiento
- Estado del enlace
- Vector de distancia
- Enrutamiento jerárquico

4.6 Enrutamiento en Internet
- RIP
- OSPF
- BGP

4.7 enrutamiento de difusión y multidifusión

# NAT: network address translation




*todos* los datagramas *dejando la red* local comparten la misma dirección IP pública del NAT: 138.76.29.7, con distintos puertos fuente

Los datagramas con fuente o destino en la red local tendrán direcciones IP: 10.0.0.x

# NAT: network address translation

*motivación:* la red local utiliza solo una dirección IP en lo que respecta al mundo exterior:

- rango de direcciones no necesarias para el ISP: sólo una dirección IP para todos los dispositivos
- puede cambiar direcciones de dispositivos en la red local sin notificar al mundo exterior
- puede cambiar el ISP sin cambiar las direcciones de los dispositivos en la red local
- dispositivos dentro de la red local no explícitamente direccionables o visibles por el mundo exterior (un plus de seguridad)

# NAT: network address translation

*implementación*: un enrutador NAT debe:

- *A los datagramas salientes: remplazar:* dirección IP fuente, puerto # de cada datagrama al formato: dirección IP NAT, nuevo puerto #

  . . . clientes remotos/servidores responderán usando: dirección IP NAT, nuevo puerto #) como dirección IP destino y puerto destino

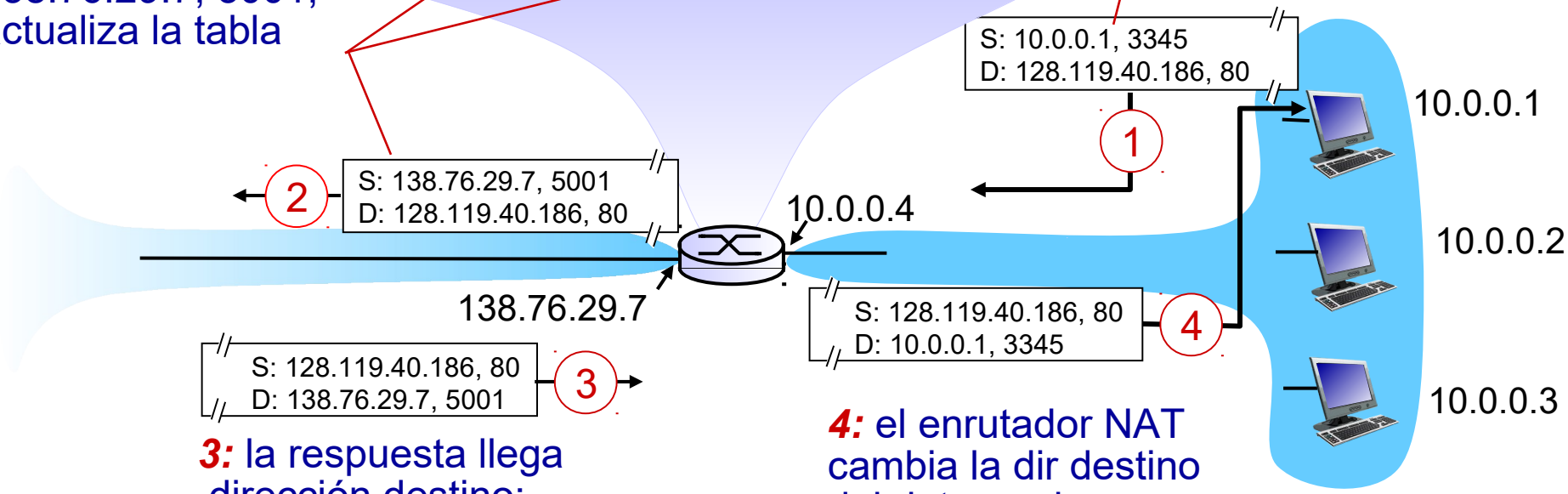
- *recordar (en una tabla de traducción NAT)* para cada pareja: dirección fuente, puerto #, a dirección IP NAT, nuevo puerto #)
- *A los datagramas salientes: remplazar:* dirección IP NAT, nuevo puerto #) en el campo destino de cada paquete entrante con su correspondiente: dirección IP fuente, puerto #) almacenado en la tabla de traducción NAT

# NAT: network address translation

| Tabla de traducción NAT | |
|---|---|
| la WAN de dir. | Lado LAN de dir. |
| 138.76.29.7, 5001 | 10.0.0.1, 3345 |
| …… | …… |

**2:** el enrutador NAT cambia la dirección fuente del datagrama de 10.0.0.1, 3345 a 138.76.29.7, 5001, actualiza la tabla

**1:** equipo 10.0.0.1 envía un paquete a 128.119.40.186, 80




**3:** la respuesta llega dirección destino: 138.76.29.7, 5001

**4:** el enrutador NAT cambia la dir destino del datama de 138.76.29.7, 5001 a 10.0.0.1, 3345

# NAT: network address translation

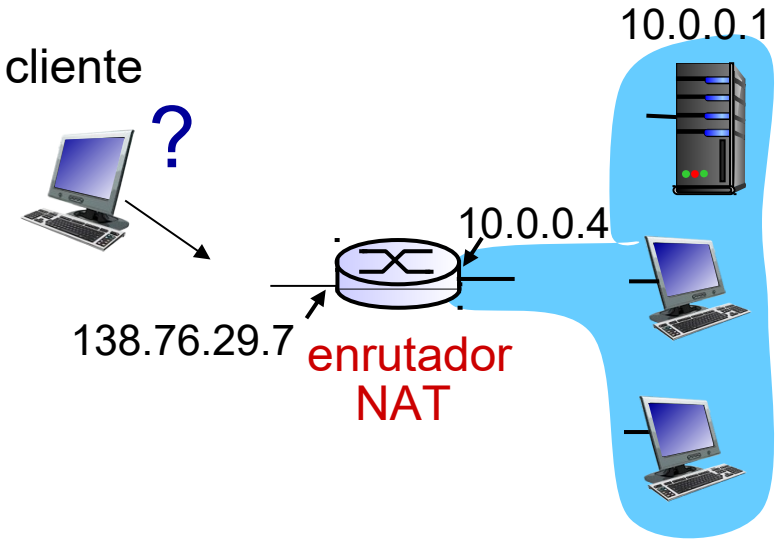
- campo de 16-bit puerto-número:
  - 60,000 conexiones simultáneas con una sola cara LAN!
- NAT es controversial:
  - enrutadores solo deben procesar hasta la capa 3
  - Viola el argumento de conexión extremo-a-extremo
    - La posibilidad de un NAT debe ser tomada en cuenta por los diseñadores y desarrolladores de apps
  - la escasez de direcciones debe ser resuelta por IPv6



*Kurose and Ross, Computer Networking: A Top Down Approach,* 6th edition, Addison-Wesley, 2012

# NAT: problema transversal

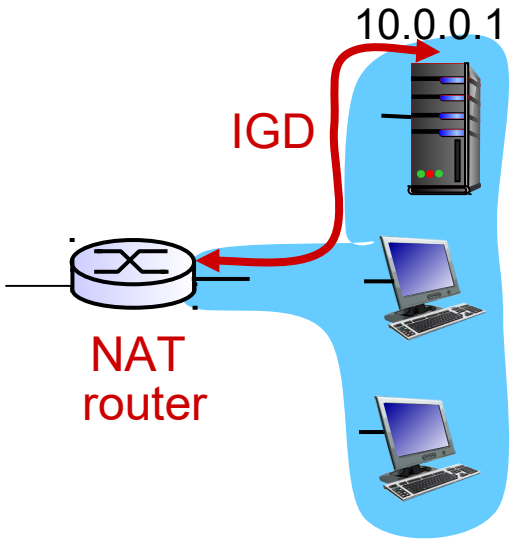
- el cliente quiere conectarse al servidor con la dirección IP 10.0.0.1
  - dirección del servidor 10.0.0.1 local a LAN (el cliente no puede usarla como dirección IP destino)
  - Sólo hay una dirección IP posible desde fuera de la LAN: 138.76.29.7
- *solución 1:* configurar estáticamente en el NAT para reenviar solicitudes de conexión entrantes en un puerto determinado al servidor
  - e.g., (123.76.29.7, puerto 2500) siempre reenvía solicitudes al 10.0.0.1 puerto 25000



# NAT: problema transversal

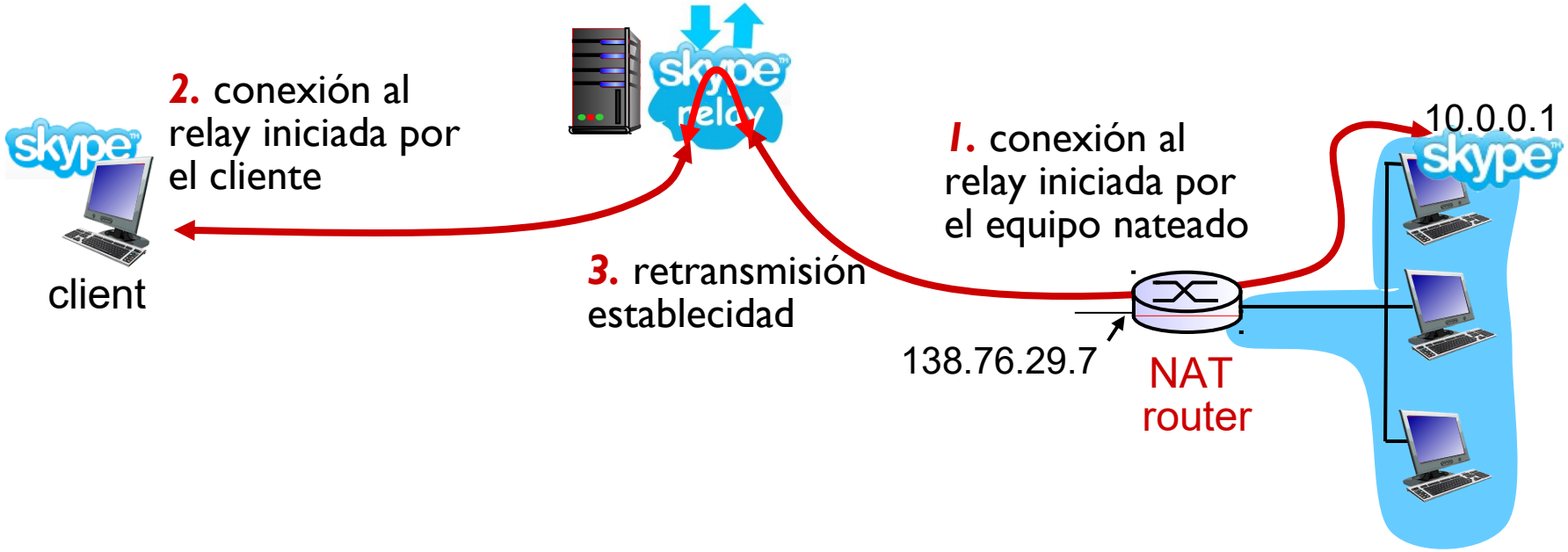
- *solución 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Permite que el equipo nateado:
  - Aprenda la dirección IP pública (138.76.29.7)
  - agregue/elimine mapeo de puertos (con tiempos de arrendamiento)

ejemplo., automatiza la configuración de la tabla de traducción estática NAT



# NAT: problema transversal

- *solución 3:* retransmisión - relaying (usado por Skype)
  - Los clientes nateados establecen conexiones retransmitidas
  - Los clientes externos se conectan al retransmisor-relay
  - El relay hace un puente para los paquetes que van y vienen de la conexión



# Capítulo 4: la ruta

4.1 Introducción
4.2 Circuito virtual y redes de datagramas
4.3 ¿Qué hay dentro de un enrutador?
4.4 IP: Protocolo de Internet
- Formato del datagrama
- Direccionamiento IPv4
- Protocolo ICMP
- Direccionamiento IPv6

4.5 Algoritmos de enrutamiento
- Estado del enlace
- Vector de distancia
- Enrutamiento jerárquico

4.6 Enrutamiento en Internet
- RIP
- OSPF
- BGP

4.7 enrutamiento de difusión y multidifusión

# ICMP: internet control message protocol

- utilizado por equipos y enrutadores para comunicar información de nivel de red
  - informe de errores: equipos inalcanzable, red, puerto, protocolo
  - solicitud / respuesta de eco (utilizada por ping)
- capa de red justo por encima del protocolo IP:
  - Los mensajes ICMP son transportados por los datagramas IP
- mensajes ICMP: tipo, código, más 8 primeros bytes de la IP que está causando el error

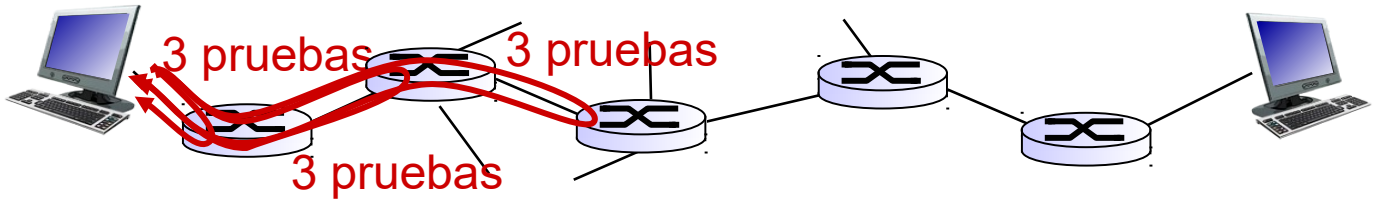
| Tipo | Código | descripción |
|---|---|---|
| 0 | 0 | echo reply (ping) |
| 3 | 0 | dest. network unreachable |
| 3 | 1 | dest host unreachable |
| 3 | 2 | dest protocol unreachable |
| 3 | 3 | dest port unreachable |
| 3 | 6 | dest network unknown |
| 3 | 7 | dest host unknown |
| 4 | 0 | source quench (congestion control - not used) |
| 8 | 0 | echo request (ping) |
| 9 | 0 | route advertisement |
| 10 | 0 | router discovery |
| 11 | 0 | TTL expired |
| 12 | 0 | bad IP header |

# Traceroute e ICMP

- La fuente envía una serie de segmentos UDP al destino
  - El primerto tiene un TTL =1
  - El segundo tiene un TTL=2, etc.
- Cuando *n conjunto de datagramas llegan al enrutador* n:
  - El enrutador descarta los datagramas
  - Y envía un mensaje a la fuente con el tipo 11 y el código 0
  - El mensaje ICMP incluye el nombre del enrutador y la dirección IP
- Cuando los mesajes de ICMP llegan Los registros RTTs se guardan en la fuente

*Criterio de parada:*

- El segmento UDP eventualmente llega el equipo destino
- El destino envía el ICMP "port unreachable" mensaje (tipo 3, código 3)
- La fuente para



# IPv6: motivación

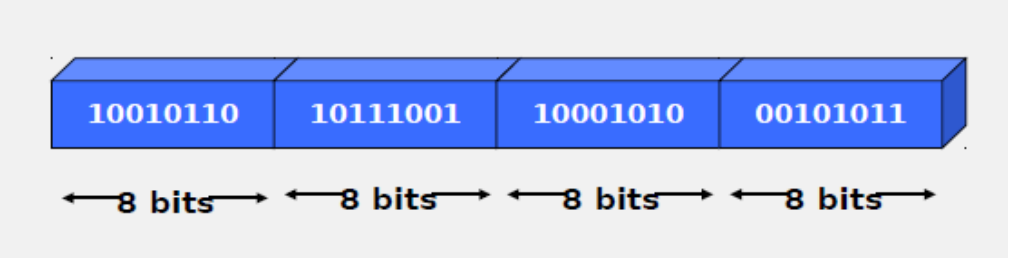
- *protocolo propuesto como reemplazo del protocolo de Internet actualmente en uso (IP versión 4)*
- *motivación inicial: el espacio de direcciones de* 32-bit está pronto será asignado completamente.
- motivación adicional:
  - el formato de la cabecera ayuda a la velocidad de procesamiento y reenvío
  - los cambios en la cabecera facilitan el QoS

*formato del datagrama IPv6:*

- longitud fija de 40 bytes
- no se permite la fragmentación

# IPv6

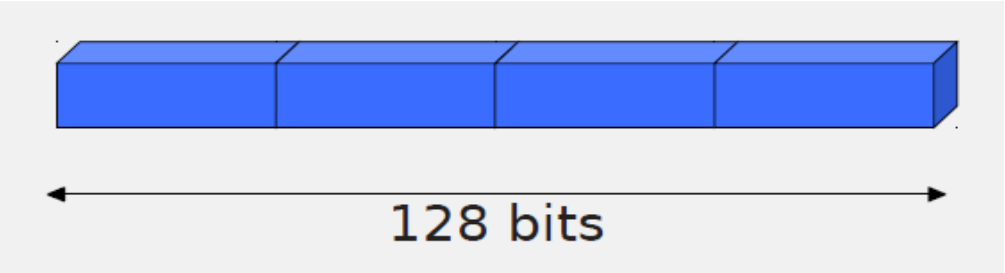
- *IPv4: 32 bits.*




- *Máximo teórico de direcciones disponibles:*

  *4.294.967.296*

- *IPv6: 128 bits.*




- *Máximo teórico:*

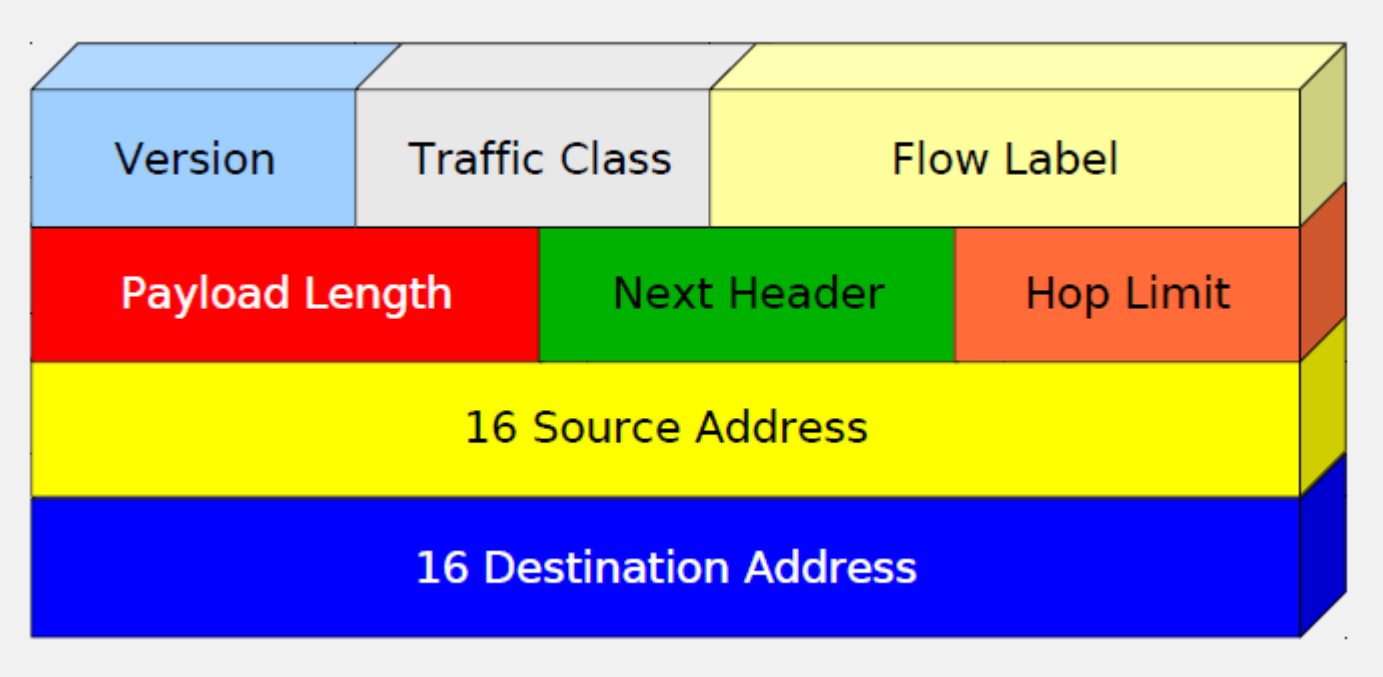
  *340282366920938463463374607431768211456*

# IPv6: formato de la cabecera

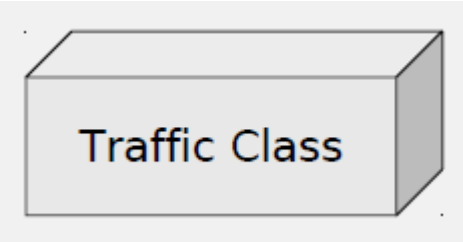
*prioridad:* identifica la prioridad entre los datagramas en el flujo

*etiqueta de flujo:* identifica datagramas en el mismo "flujo". (concepto de "flujo" no bien definido).

*siguiente cabecera:* identifica el protocolo de capa superior para datos

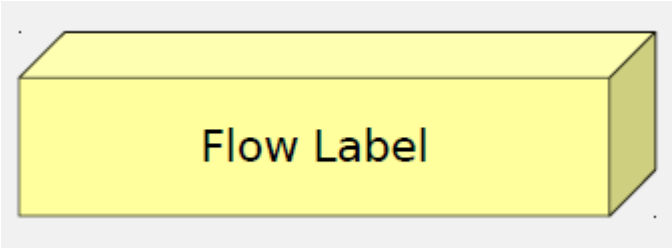


# IPv6: formato de la cabecera

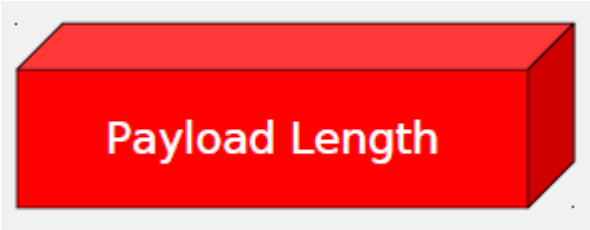



Este campo es utilizado para distinguir paquetes con diferente requisitos de tiempo real de envío.
Fue utilizado raras veces en IPv4 por los enrutadores pero actualmente es utilizado mejorar el envío de tráfico multimedia.

# IPv6: formato de la cabecera




Es un campo experimental que será utilizado para que el origen y el destino puedan establecer una pseudo conexión con requisitos específicos y propiedades particulares. Por ejemplo, cuando un par de nodos necesiten cierto ancho de banda pueden levantar esta bandera y los enrutadores pueden darle tratamiento especial.



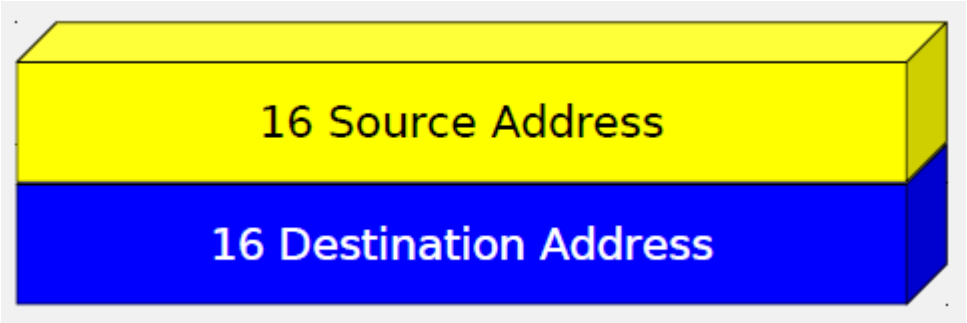

Este campo indica cuantos bytes contiene el datagrama luego de la cabecera de 40 bytes

# IPv6: formato de la cabecera

Indica el tipo de encabezado que le sigue a este.
Hay 6 tipos de encabezados:
Por ejemplo, si el tipo es última IP entonces el siguiente encabezado será el tipo de protocolo de transporte utilizado: TCP o UDP

Evita que los paquetes vivan por siempre y es equivalente al TTL de IPv4

# IPv6: formato de la cabecera




Las direcciones de origen y destino en Ipv6 están conformadas por 8 conjuntos de 16 bits cada uno.

La representación de cada conjunto se hace con cuatro dígitos hexadecimales

Ejemplo:

12AB:0000:0000:CD30:0123:4567:89AB:0DEF

Los ceros continuos pueden ser compactados:

12AB:0:0:CD30:123:4567:89AB:0DEF

12AB::CD30:123:4567:89AB:DEF

Las direcciones Ipv4 se pueden representar así:

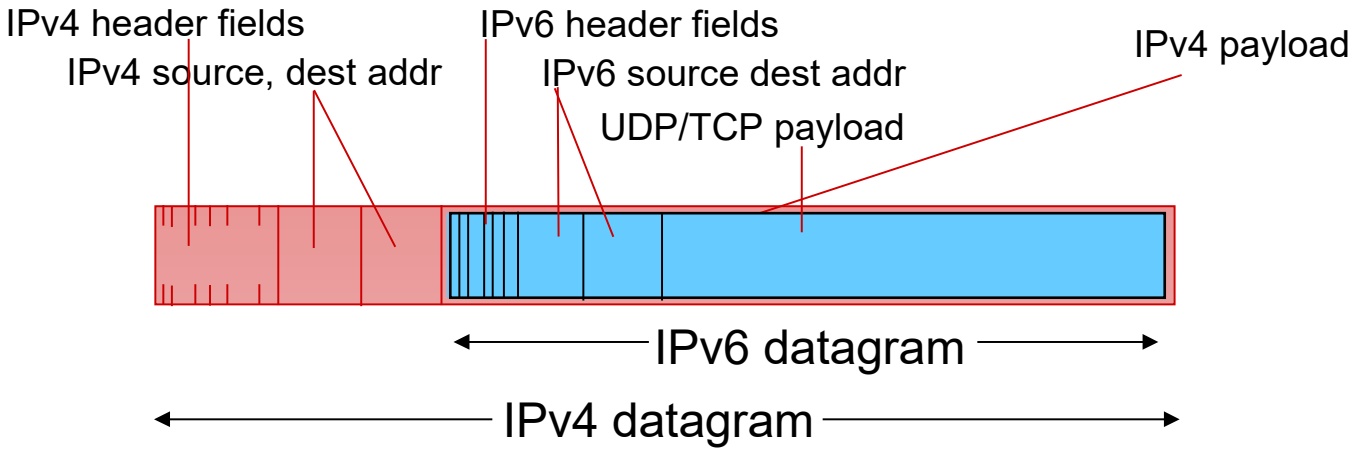
::150.185.128.12

# Otros cambios con respecto a IPv4

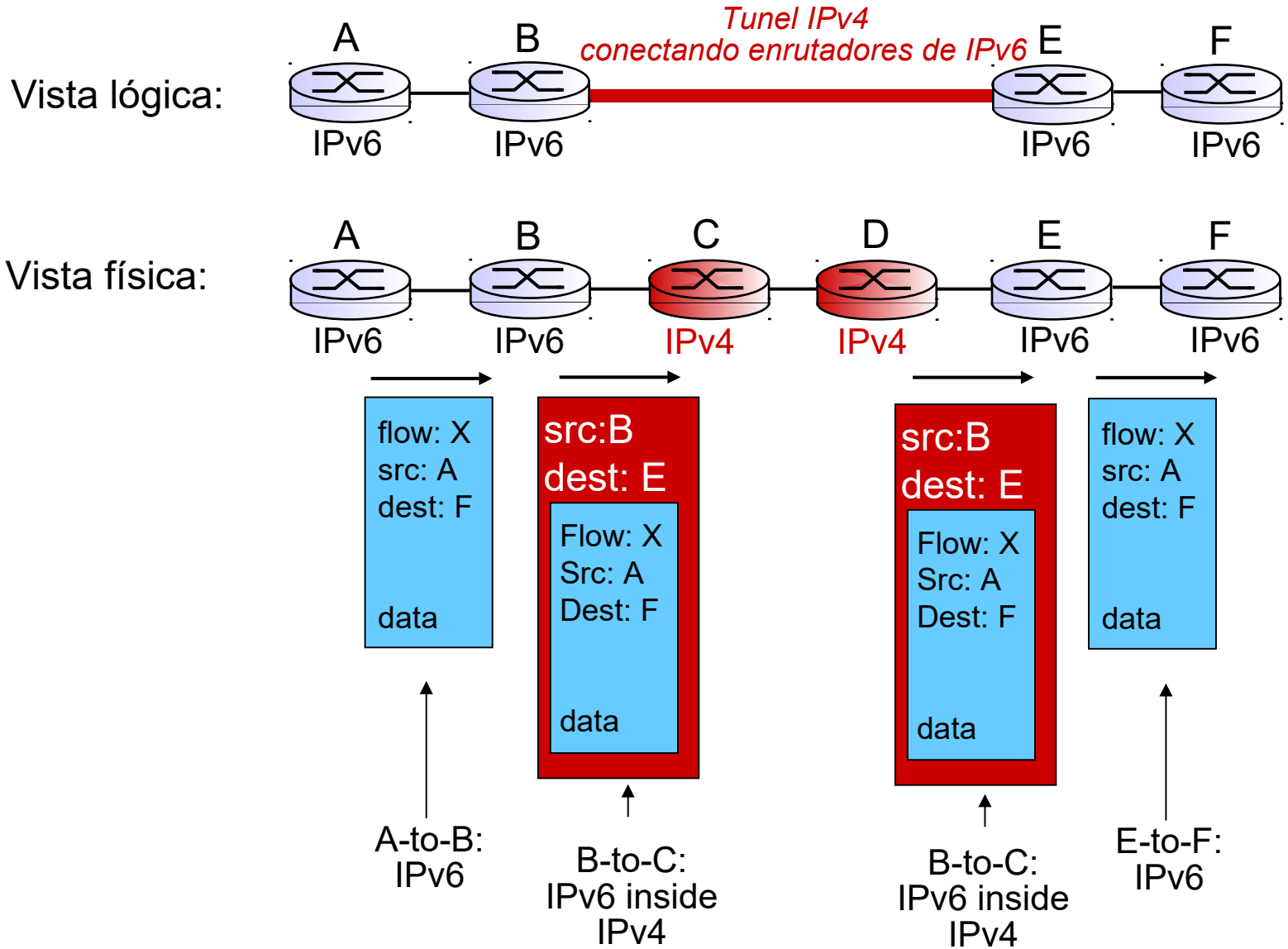
- *suma de comprobación: eliminada por completo para reducir el tiempo de procesamiento en cada salto*
- *opciones:* permitido, pero fuera del encabezado, indicado por el campo "Siguiente encabezado"
- *ICMPv6:* nueva versión de ICMP
  - tipos de mensajes adicionales, ejemplo “Packet Too Big”
  - funciones de gestión de grupo de multidifusión

# Transición de IPv4 a IPv6

- No todos los enrutadores puede adaptarse simultáneamente
  - no hay un "día cero"
  - ¿Cómo la red puede operar en un ambiente mixto con enrutadores IPv4 y IPv6?
- *tuneles:* IPv6 va hospedado cómo un dato (*payload)* en el datagrama IPv4 entre enrutadores IPv4 o viceversa



# Tunel de IPv6



# IPv6: adopción

- El instituto nacional de estándares de los EUA estimó para el 2013:
  - ~3% migración para las direcciones IP de los enrutadores comerciales
  - ~11% migración para los enrutadores en el sector gubernamental

- *un largo (largoooo!) tiempo para la migración total*
  - 20 años y quizás más!
  - Sustituir las direcciones IPv4 a nivel de usuario y apps de último nivel tomará mucho tiempo, hay muchos factores:
    - La IPv4 sigue funcionando, porqué eliminarla…
    - Necesitas que tu ISP anuncie los bloques de IPv6 (en Vzla, CANTV aún no lo hace, y es el ISP más grande del país) en la ULA ya probamos nuestros bloques a través de túneles

# Capítulo 4: la ruta

4.1 Introducción
4.2 Circuito virtual y redes de datagramas
4.3 ¿Qué hay dentro de un enrutador?
4.4 IP: Protocolo de Internet
- Formato del datagrama
- Direccionamiento IPv4
- Protocolo ICMP
- Direccionamiento IPv6

4.5 Algoritmos de enrutamiento
- Estado del enlace
- Vector de distancia
- Enrutamiento jerárquico

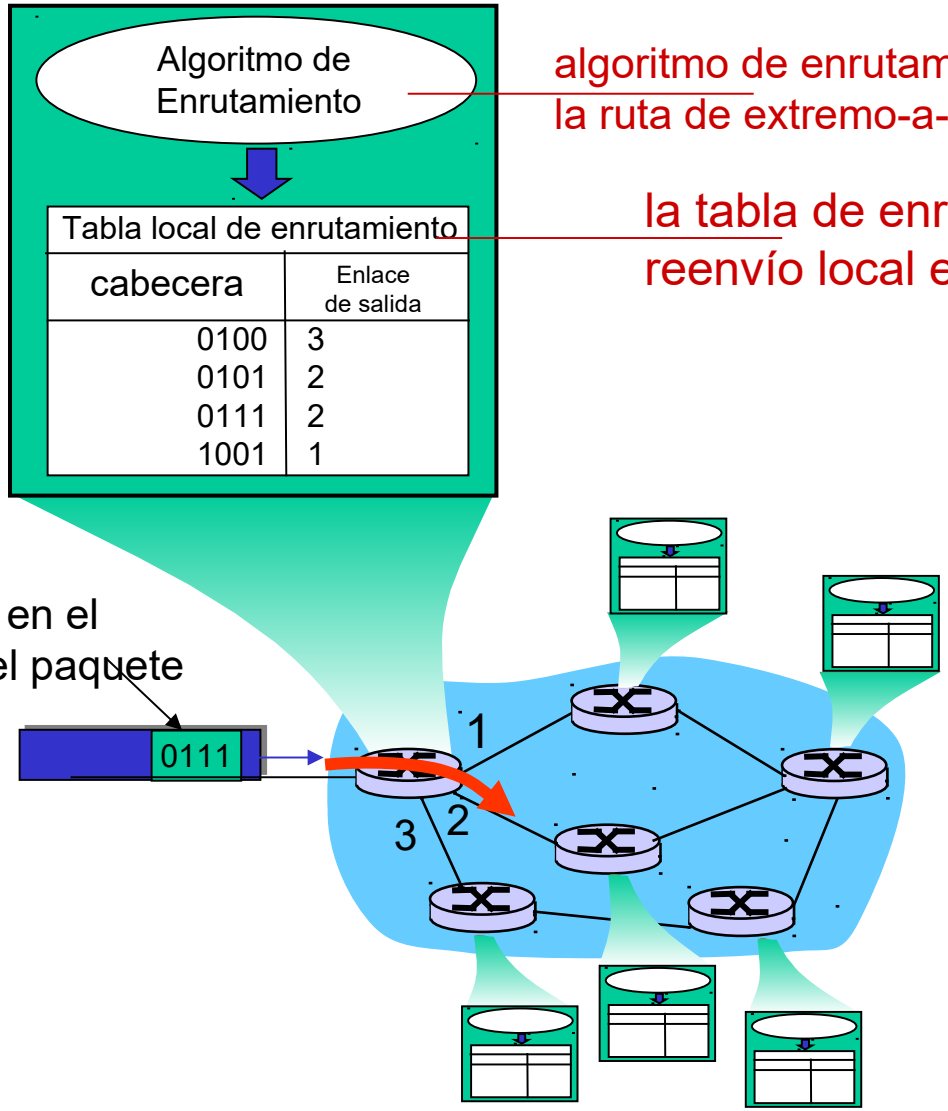
4.6 Enrutamiento en Internet
- RIP
- OSPF
- BGP

4.7 enrutamiento de difusión y multidifusión



*Kurose and Ross, Computer Networking: A Top Down Approach,* 6th edition, Addison-Wesley, 2012

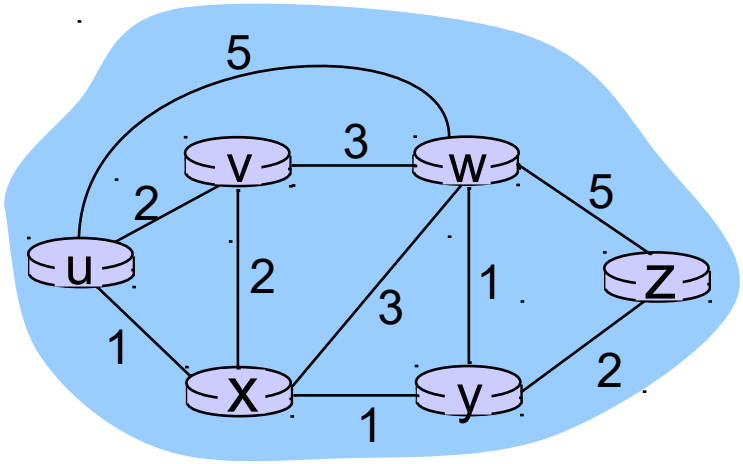
# Interacción entre enrutamiento, reenvío




algoritmo de enrutamiento determina la ruta de extremo-a-extremo a través de la red

la tabla de enrutamiento determina reenvío local en este enrutador

# Abstracción de grafos




grafo: G = (N,E)

N = conjunto de enrutadores = { u, v, w, x, y, z }

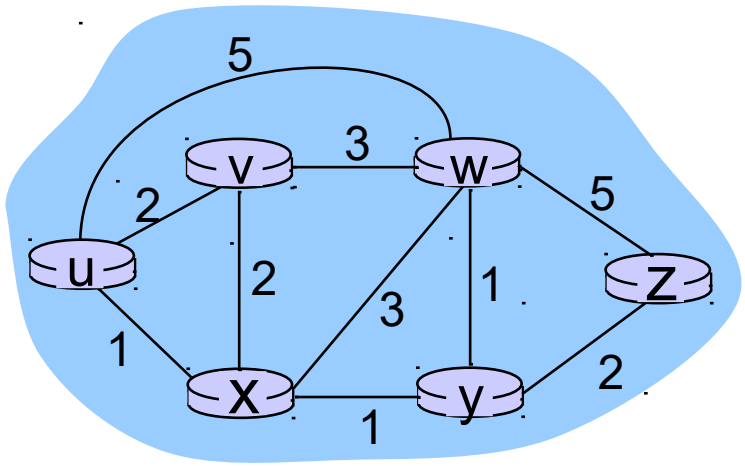
E = conjunto de enlaces ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

*aparte:* la abstracción de grafos es útil en otros contextos de red, por ejemplo, P2P, donde N es un conjunto de pares y E es un conjunto de conexiones TCP



*Kurose and Ross, Computer Networking: A Top Down Approach,* 6th edition, Addison-Wesley, 2012

# Abstracción de grafos: costos




c(x,x') = cost of link (x,x')
ejemplo., c(w,z) = 5

el costo siempre puede ser 1, o inversamente relacionado con el ancho de banda,
o inversamente relacionado con la congestión

costo de la ruta $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

*Pregunta clave:* ¿Cuál es la ruta de menor costo entre u y z?
*Algoritmo de enrutamiento:* el algoritmo de enrutamiento encontrará la ruta de menor costo

# Clasificación de el algoritmos de enrutamiento

*P: información global o descentralizada?*

*global:*

- todos los enrutadores tienen la topología completa, información de costo del enlace
- Algoritmos del "estado del enlace"

*descentralizada:*

- el enrutador conoce a los vecinos conectados físicamente, vincula los costos a los vecinos
- proceso iterativo de computación, intercambio de información con los vecinos
- Algoritmos del "vector de la distancia"

*P: estático o dinámico?*

*estático:*

- las rutas cambian lentamente con el tiempo

*dinámico:*

- las rutas cambian más rápido
  - actualización periódica
  - en respuesta a los cambios en los costos del enlace

# Capítulo 4: la ruta

4.1 Introducción
4.2 Circuito virtual y redes de datagramas
4.3 ¿Qué hay dentro de un enrutador?
4.4 IP: Protocolo de Internet
- Formato del datagrama
- Direccionamiento IPv4
- Protocolo ICMP
- Direccionamiento IPv6

4.5 Algoritmos de enrutamiento
- Estado del enlace
- Vector de distancia
- Enrutamiento jerárquico

4.6 Enrutamiento en Internet
- RIP
- OSPF
- BGP

4.7 enrutamiento de difusión y multidifusión



*Kurose and Ross, Computer Networking: A Top Down Approach,* 6th edition, Addison-Wesley, 2012

# Algoritmo de enrutamiento: estado del enlace

*Algoritmo de Dijkstra*

- topología de red, costos de enlace conocidos para todos los nodos
  - logrado a través de "difusión estatal de enlace"
  - todos los nodos tienen la misma info
- calcula rutas de menor costo desde un nodo ('fuente') a todos los otros nodos
  - da la tabla de enrutamiento para ese nodo
- iterativo: después de k iteraciones, conoce la ruta de menor costo para el destino k

*notación:*

- $c(x,y)$: lcosto del enlace del nodo x a y; $= \infty$ si no tiene vecinos directos
- $D(v)$: valor actual del costo de la ruta de origen a destino v
- $p(v)$: nodo predecesor a lo largo de la ruta de la fuente a v
- $N'$: conjunto de nodos cuya ruta de menor costo definitivamente conocida



*Kurose and Ross, Computer Networking: A Top Down Approach,* 6th edition, Addison-Wesley, 2012

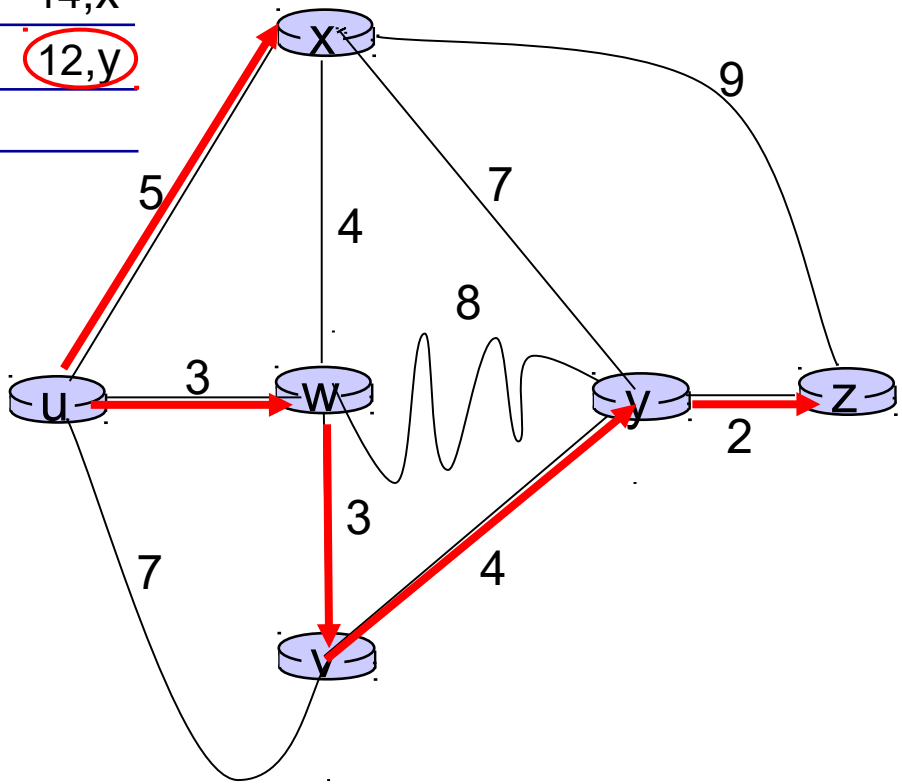
# Algoritmo de Dijsktra

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
until all nodes in N'
```

# Algoritmo de Dijkstra: ejemplo

| Step | N' | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u | 7,u | 3,u | 5,u | ∞ | ∞ |
| 1 | uw | 6,w | | 5,u | 11,w | ∞ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

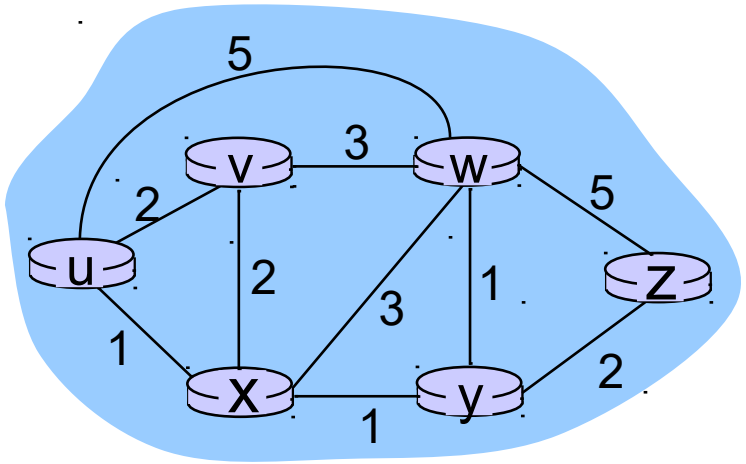



*notas:*

- construir el árbol de ruta más corta mediante el seguimiento de los nodos predecesores
- los empates pueden existir (se pueden romper arbitrariamente)

# Algoritmo de Dijkstra: otro ejemplo

| paso | N' | D(v),p(v) | D(w),p(w) | D(x),p(x) | D(y),p(y) | D(z),p(z) |
|---|---|---|---|---|---|---|
| 0 | u | 2,u | 5,u | 1,u | ∞ | ∞ |
| 1 | ux | 2,u | 4,x | | 2,x | ∞ |
| 2 | uxy | 2,u | 3,y | | | 4,y |
| 3 | uxyv | | 3,y | | | 4,y |
| 4 | uxyvw | | | | | 4,y |
| 5 | uxyvwz | | | | | |



# Algoritmo de Dijkstra: ejemplo (2)

el árbol de ruta más corto resultante de u:

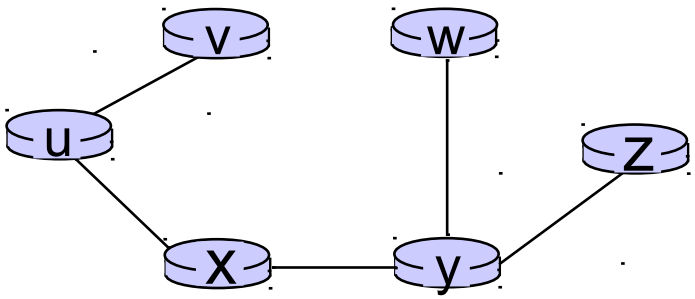



tabla de enrutamiento resultante para u:

| destino | enlace |
|---|---|
| v | (u,v) |
| x | (u,x) |
| y | (u,x) |
| w | (u,x) |
| z | (u,x) |

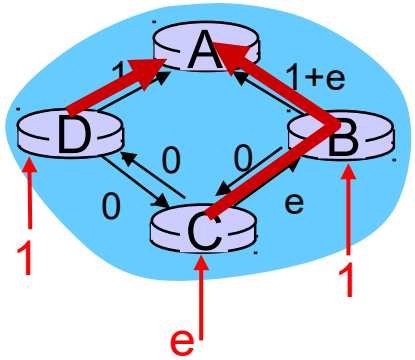
# Algoritmo de Dijkstra, discusión

*Complejidad del algoritmo:* n nodos

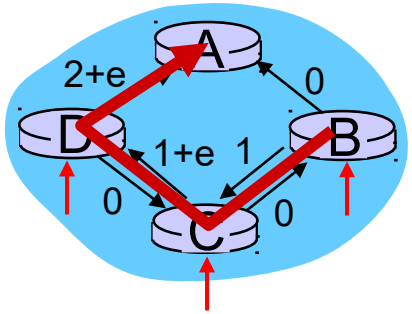
- Cada iteración: necesita revisar todos los nodos, w, no en N
- n(n+1)/2 comparisons: $O(n^2)$
- Existen implementaciones más eficientes: O(nlogn)

*posible oscilaciones:*

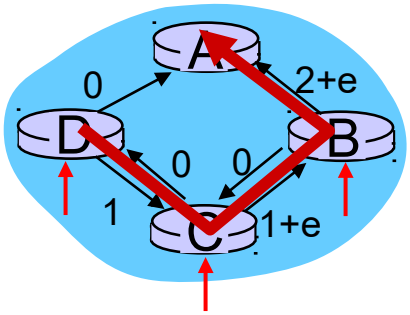
- ejemplo., el costo del enlace de soporte es igual a la cantidad de tráfico transportado:



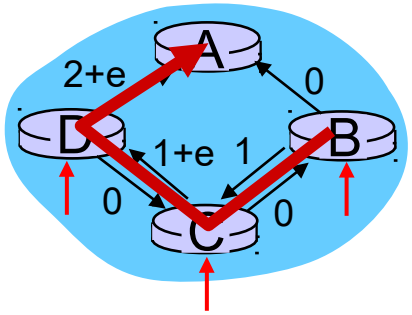

inicialmente




dados estos costos, consigue el nuevo enrutamiento…. resultando en nuevos costos




dados estos costos, consigue el nuevo enrutamiento…. resultando en nuevos costos




dados estos costos, consigue el nuevo enrutamiento…. resultando en nuevos costos

# Capítulo 4: la ruta

4.1 Introducción
4.2 Circuito virtual y redes de datagramas
4.3 ¿Qué hay dentro de un enrutador?
4.4 IP: Protocolo de Internet
- Formato del datagrama
- Direccionamiento IPv4
- Protocolo ICMP
- Direccionamiento IPv6

4.5 Algoritmos de enrutamiento
- Estado del enlace
- Vector de distancia
- Enrutamiento jerárquico

4.6 Enrutamiento en Internet
- RIP
- OSPF
- BGP

4.7 enrutamiento de difusión y multidifusión

# Algoritmo de vector de distancia

*Ecuación de Bellman-Ford (programación dinámica)*

dado

$d_x(y)$ := costo del camino de menor costo de x a y

luego

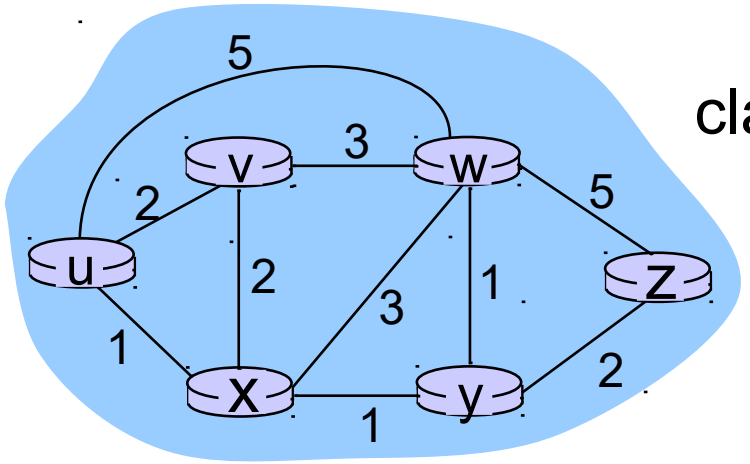
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

costo desde el vecino v hasta el destino y

costo hasta el vecino v

*min* tomado entre todos los vecinos v de x

# Bellman-Ford: ejemplo




claramente, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

la ecuación de B-F dice:

$$
\begin{aligned}
d_u(z) &= \min \{ c(u,v) + d_v(z), \\
&\qquad c(u,x) + d_x(z), \\
&\qquad c(u,w) + d_w(z) \} \\
&= \min \{2 + 5, \\
&\qquad 1 + 3, \\
&\qquad 5 + 3\} \ = 4
\end{aligned}
$$

nodo que alcanza el mínimo es el próximo salto en el camino más corto, utilizado en la tabla de enrutamiento

# Algoritmo de vector de distancia

- $D_x(y)$ = estima el menor costo de x a y
  - x mantien el vector de distancia $\mathbf{D}_x = [D_x(y): y \in N]$
- nodo x:
  - sabe el costo para cada vecino v: $c(x,v)$
  - mantiene los vectores de distancia de sus vecinos. Para cada vecino v, x, mantiene $\mathbf{D}_v = [D_v(y): y \in N]$

# Algoritmo de vector de distancia

*Idea clave:*

- de vez en cuando, cada nodo envía su propia estimación del vector de distancia a los vecinos
- cuando x recibe un nuevo estimado DV de su vecino, actualiza su propio DV usando la ecuación de B-F:

$$D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- en condiciones naturales menores, la estimación $D_x(y)$ *convergen al costo mínimo real* $d_x(y)$



*Kurose and Ross, Computer Networking: A Top Down Approach,* 6th edition, Addison-Wesley, 2012

# Algoritmo de vector de distancia

*iterativo, asincrónico:* cada iteración local es causada por:

- cambios locales en los costos de los enlaces
- DV actualiza el mensaje por un vecino

*distribuido:*

- cada nodo notifica a su vecino sólo cuando DV cambia
  - luego los vecinos notifican a su vecinos y así sucesivamente

*cada nodo:*

*espera* por (cambio en el costo del enlace local o mensaje de vecino)

↓

*recalcula* estimaciones

↓

si DV para cualquier destino cambia, *notifica* a sus vecinos

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

**node x table**

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | ∞ | ∞ | ∞ |
| z | ∞ | ∞ | ∞ |

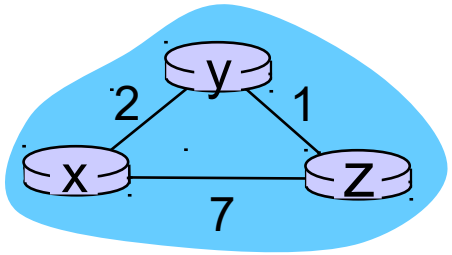
| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

**node y table**

| from \ cost to | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

**node z table**

| from \ cost to | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 7 | 1 | 0 |




tiempo

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

**node x table**

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | ∞ | ∞ | ∞ |
| z | ∞ | ∞ | ∞ |

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node y table**

| from \ cost to | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

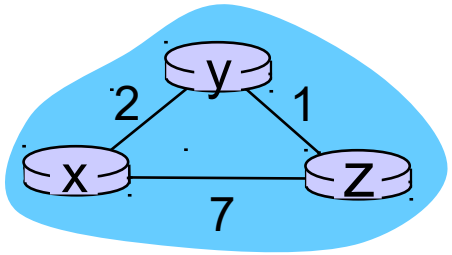
| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node z table**

| from \ cost to | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 7 | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

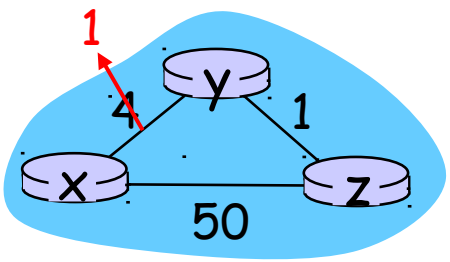



tiempo

# Vector de distancia: cambios en el costo de los enlaces

*cambios en los costos de enlace:*

- nodo detecta cambio de costo de enlace local
- actualiza la información de ruta, recalcula vector de distancia
- si el DV cambia, notifica a los vecinos




"buenas noticias viajan rápido"

$t_0$: y detecta cambios en el costo del enlace, actualiza su DV, informa a sus vecinos.

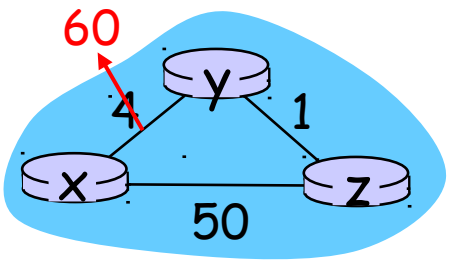
$t_1$: z recibe la actualización de y, actualiza su tabla, calcula el nuevo costo mínimo para x, envía a sus vecinos su DV.

$t_2$: y recibe la actualización de z, actualiza su tabla de distancia. Los costos mínimos de y no cambian, por lo que y no envía un mensaje a z

# Vector de distancia: cambios en el costo de los enlaces

*cambios en el costo de los enlaces:*

- nodo detecta cambio de costo de enlace local
- *malas noticias viajan lento* – el problema de "contar hasta el infinito"!
- 44 iteraciones antes de que el algoritmo pueda establizarse: más detalle en el libro




*reverso envenenado:*

- si quieres ir desde Z a través de Y para llegar a X:
  - Z le dice a Y que su distancia (Z's) para llegar a X es infinita (así que Y no tomará la ruta a X vía Z)
- resolverá esto por completo hasta el problema de contar hasta el infinito?

# Comparación entre los algoritmos de LS y DV

## *Complejidad del mensaje*

- ***LS:*** con n nodos, E enlaces, O(nE) mensajes enviados
- ***DV:*** intercambio entre vecinos solamente
  - el tiempo de convergencia varía

## *velocidad de la convergencia*

- ***LS:*** $O(n^2)$ el algoritmo require de O(nE) mensajes
  - puede presentar oscilaciones
- ***DV:*** tiempo de convergencia varía
  - pueden generarse bucles de enrutamiento
  - Problema de contar hasta el infinito

## *Robustez:* ¿Qué pasa si el enrutador funciona mal?

*LS:*

- nodo puede anunciar un costo de enlace incorrecto
- cada nodo computa solo su propia tabla

*DV:*

- el nodo DV puede anunciar un costo de ruta incorrecta
- cada tabla de nodo es usada por los demás
  - El error se propagará por la red

# Capítulo 4: la ruta

4.1 Introducción
4.2 Circuito virtual y redes de datagramas
4.3 ¿Qué hay dentro de un enrutador?
4.4 IP: Protocolo de Internet
- Formato del datagrama
- Direccionamiento IPv4
- Protocolo ICMP
- Direccionamiento IPv6

4.5 Algoritmos de enrutamiento
- Estado del enlace
- Vector de distancia
- Enrutamiento jerárquico

4.6 Enrutamiento en Internet
- RIP
- OSPF
- BGP

4.7 enrutamiento de difusión y multidifusión

# Enrutamiento jerárquico

nuestro estudio de enrutamiento hasta ahora es de idealización

- Todos los enrutadores son idénticos
- La red es "plana"

… *en la práctica esto no es así!*

*escala:* con más de 600 millones de destinos:

- no puedo almacenar todos los destinos en tablas de enrutamiento!
- ¡el intercambio de la tabla de enrutamiento podría inundar los enlaces!

*autonomía administrativa*

- internet = red de redes
- cada administrador de red puede querer controlar el enrutamiento en su propia red

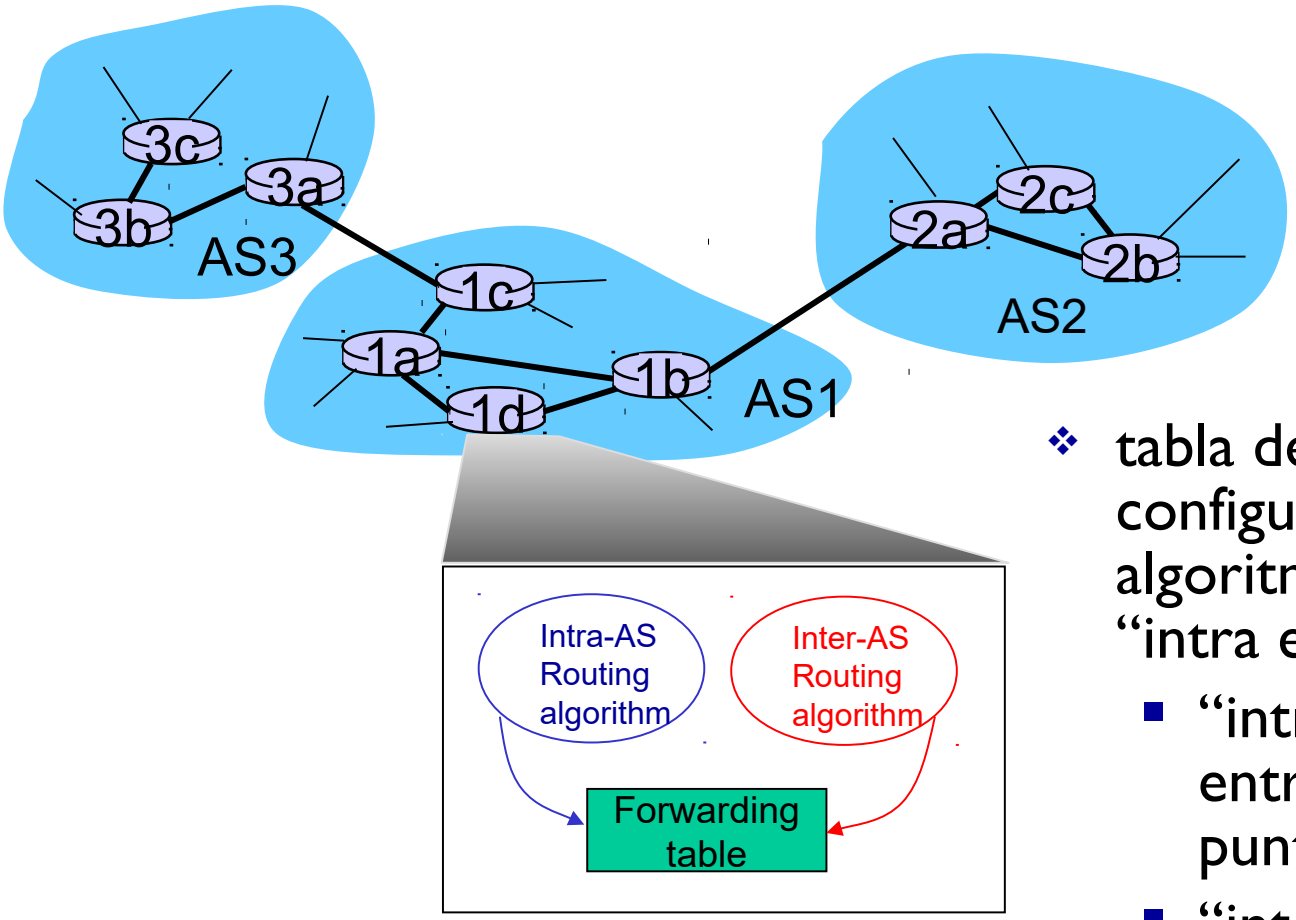
# Enrutamiento jerárquico

- agregar enrutadores a regiones, conocidas cómo: "autonomous systems" (AS)
- los enrutadores en el mismo AS ejecutan el mismo protocolo de enrutamiento
  - "intra-AS" protocolo de enrutamiento
  - los enrutadores en diferentes AS pueden ejecutar diferentes protocolos de enrutamiento "intra-AS"

*enrutador de puerta de enlace:*

- en el "borde" de su propio AS
- tiene un enlace al enrutador en otro AS

# ASs interconectados



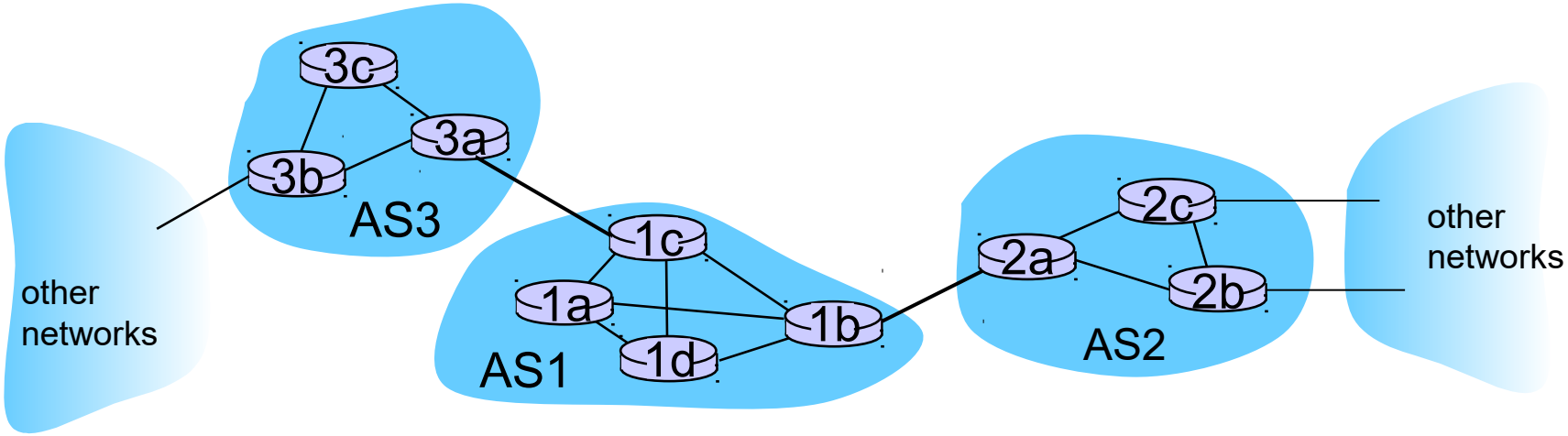

- tabla de enrutamiento es configurada por el algoritmo de enrutamiento "intra e inter-AS"
  - "intra-AS" establece entradas para los puntos internos
  - "inter-AS" establece entradas para los puntos externos

# Tareas del Inter-AS

- Supongamos que el router en AS1 recibe un datagrama destinado fuera de AS1:
  - el enrutador debe reenviar el paquete al enrutador de la puerta de enlace, pero ¿cuál?

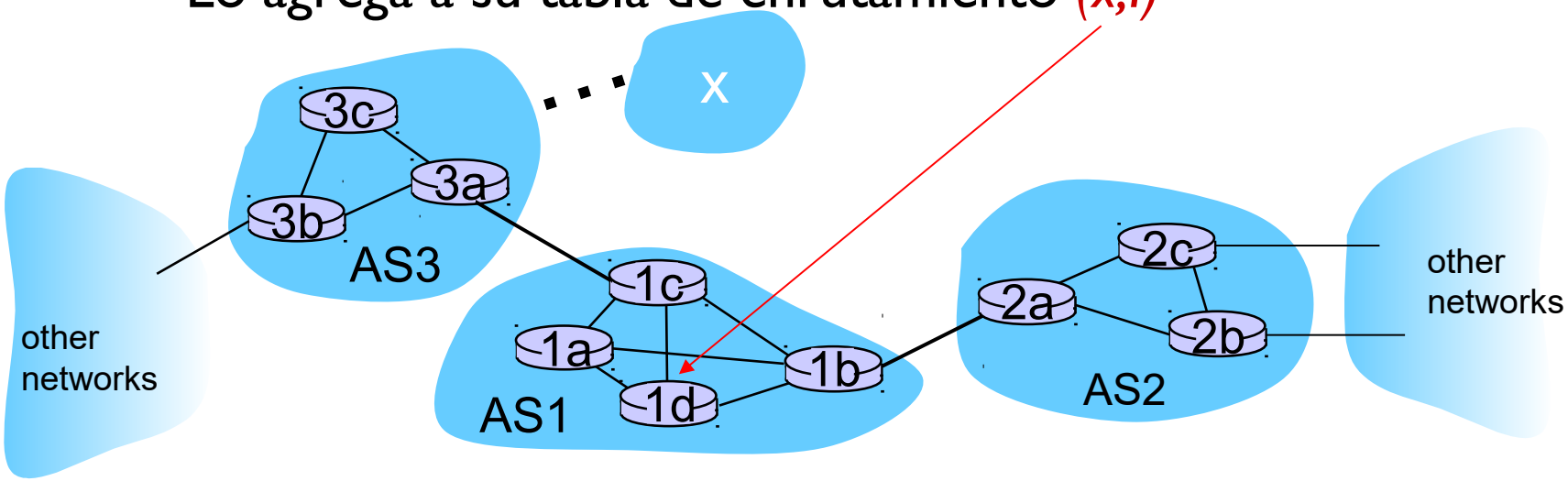
*AS1 debe:*

1. aprender qué destinos son alcanzables a través de AS2 y destinos a través de AS3
2. propagar esta información de accesibilidad a todos los enrutadores en AS1



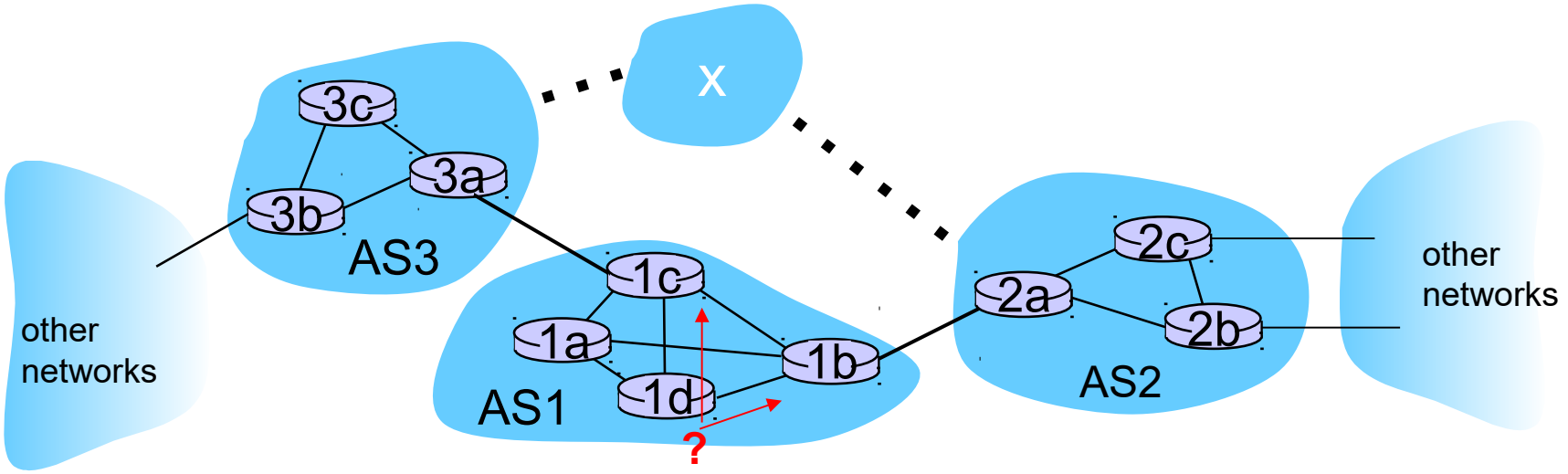
## Ejemplo: configurando la tabla de enrutamiento en el enrutador 1d

- Supongamos que AS1 aprende (vía protocolo inter-AS) que la subred x es accesible por la vía del AS3 (a través del gateway 1c), pero no a través del AS2
  - A través del protocolo inter-AS se propaga la información de accesabilidad de todos los enrutadores internos
- El enrutador 1d determina por la información intra-AS que la interfaz 1 es la del menor costo de ruta para llegar a 1c
  - Lo agrega a su tabla de enrutamiento *(x,l)*



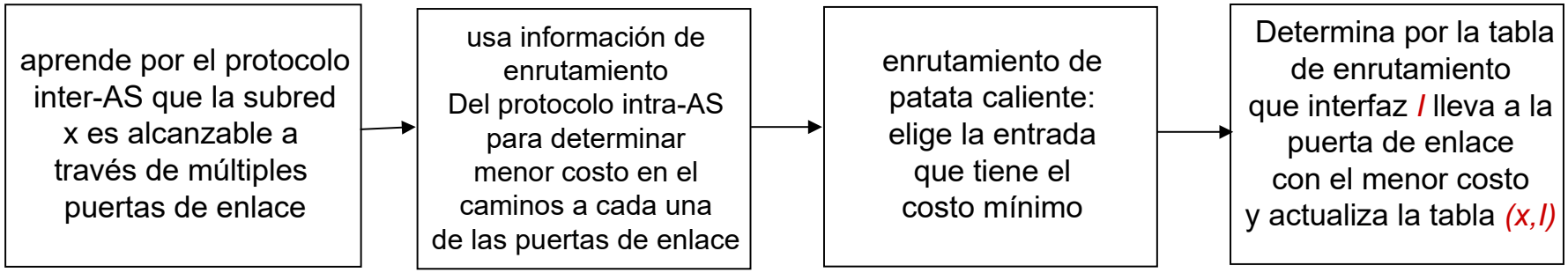
# Ejemplo: escogiendo entre múltiples ASes

- Ahora supongamos que ASI aprende por el protocol inter-AS que la subred x es accesible desde AS3 y AS2
- Para construir la tabla de enrutamiento, el enrutador 1d debe determinar cual gateway debe re-enviar los paquetes al destino x
  - Esto es trabajo del protocolo inter-AS!



# Ejemplo: escogiendo entre múltiples ASes

- Ahora supongamos que ASI aprende por el protocol inter-AS que la subred x es accesible desde AS3 y AS2
- Para construir la tabla de enrutamiento, el enrutador 1d debe determinar cual gateway debe re-enviar los paquetes al destino x
  - Esto es trabajo del protocolo inter-AS!
- *enrutamiento de la papa caliente: envía paquetes a los 2 enrutadores más cercanos*



# Capítulo 4: la ruta

4.1 Introducción
4.2 Circuito virtual y redes de datagramas
4.3 ¿Qué hay dentro de un enrutador?
4.4 IP: Protocolo de Internet
- Formato del datagrama
- Direccionamiento IPv4
- Protocolo ICMP
- Direccionamiento IPv6

4.5 Algoritmos de enrutamiento
- Estado del enlace
- Vector de distancia
- Enrutamiento jerárquico

4.6 Enrutamiento en Internet
- RIP
- OSPF
- BGP

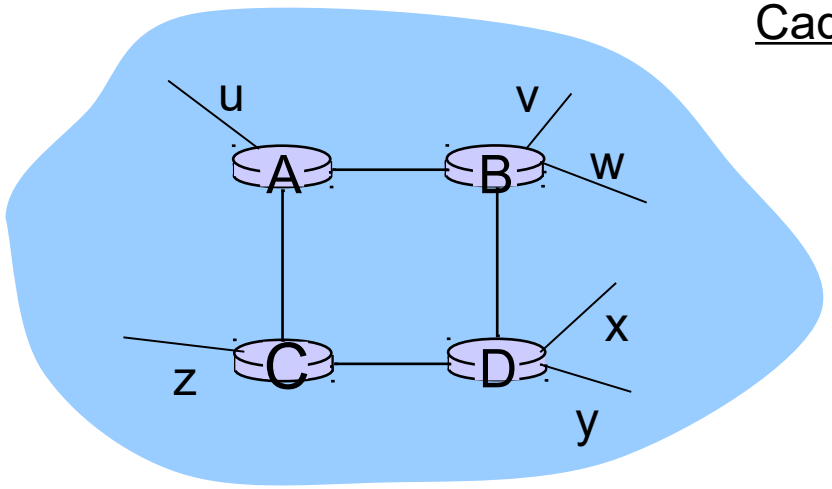
4.7 enrutamiento de difusión y multidifusión

# Enrutamiento Intra-AS

- También conocidos cómo protocolos para gateways internos, *interior gateway protocols (IGP)*
- Los protocolos intra-AS más comunes son:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

# RIP ( Routing Information Protocol)

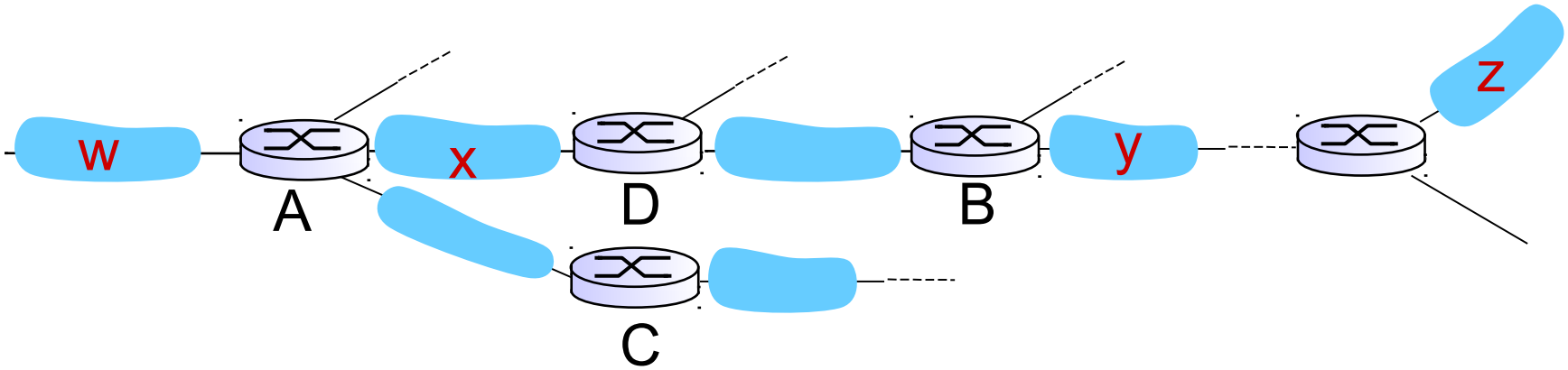
- Incluido en la distribución BSD-UNIX en 1982
- Usa un algoritmo de vector de distancia (DV)
  - metrica de distancia: # hops (max = 15 saltos), cada enlace tiene un costo de 1
  - DVs intercambia con sus vecinos cada 30 segundos la información de sus tablas (aka anuncios)
  - Cada anuncio: lista hasta 25 subredes destino *(en formato de direcciones IP)*




Cada enrutador A para su destino *subredes:*

| subred | saltos |
|---|---|
| u | 1 |
| v | 2 |
| w | 2 |
| x | 3 |
| y | 3 |
| z | 2 |

# RIP: ejemplo



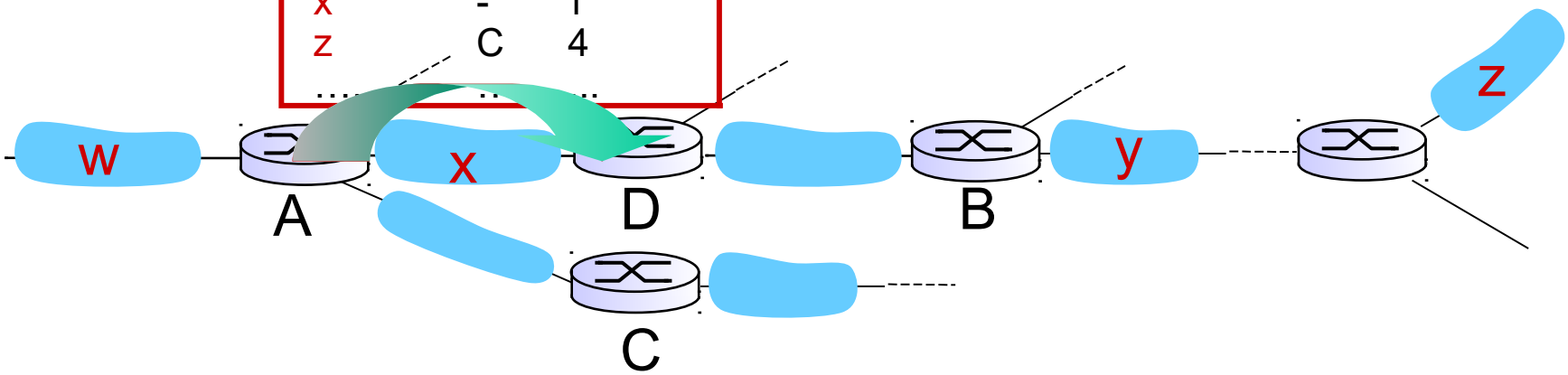

routing table in router D

| destination subnet | next router | # hops to dest |
|---|---|---|
| w | A | 2 |
| y | B | 2 |
| z | B | 7 |
| x | -- | 1 |
| …. | …. | …. |

# RIP: ejemplo

A-to-D advertisement

| dest | next | hops |
|---|---|---|
| w | - | 1 |
| x | - | 1 |
| z | C | 4 |
| .... | .... | .... |



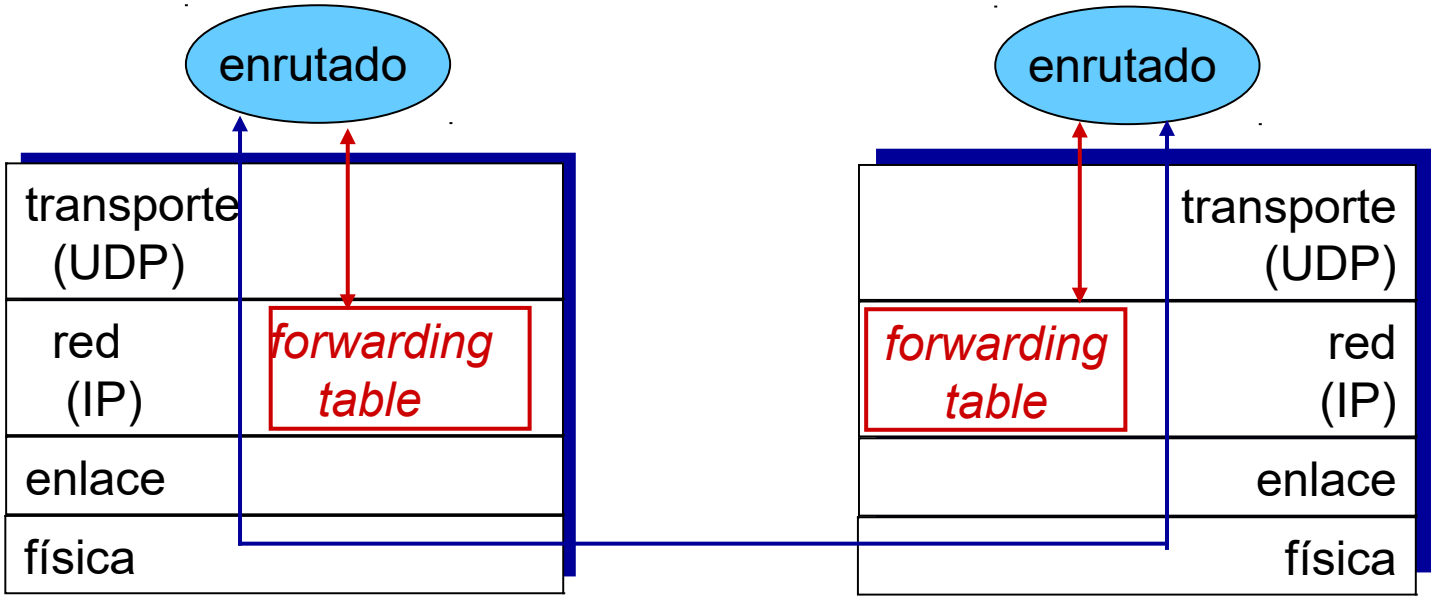

routing table in router D

| destination subnet | next router | # hops to dest |
|---|---|---|
| w | A | 2 |
| y | B | 2 |
| z | ~~B~~ A | ~~7~~ 5 |
| x | -- | 1 |
| .... | .... | .... |

# RIP: falla de enlace, recuperación

Si no escucho anuncios después de 180 segundos →el enlace vecino se declara "muerto"

- Las rutas a ese vecino son invalidades
- Se envían nuevos anuncios al resto de los vecinos
- Los vecinos en su turno envían nuevos anuncios (si sus tablas cambiaron)
- La información sobre la falla de un enlace se propaga rápidamente a la red
- *Envenenamiento reverso* se usa para previnir enlaces de ping-pong (distancia infinita = 16 hops)

# RIP procesamiento de la tabla

- La gestión de la tabla de enrutamiento de RIP en manejada por un proceso del nivel de aplicación llamado route-d (daemon)
- Los anuncios se envian en paquetes UDP y son repetidos periódicamente