REDES DE COMPUTADORAS 1

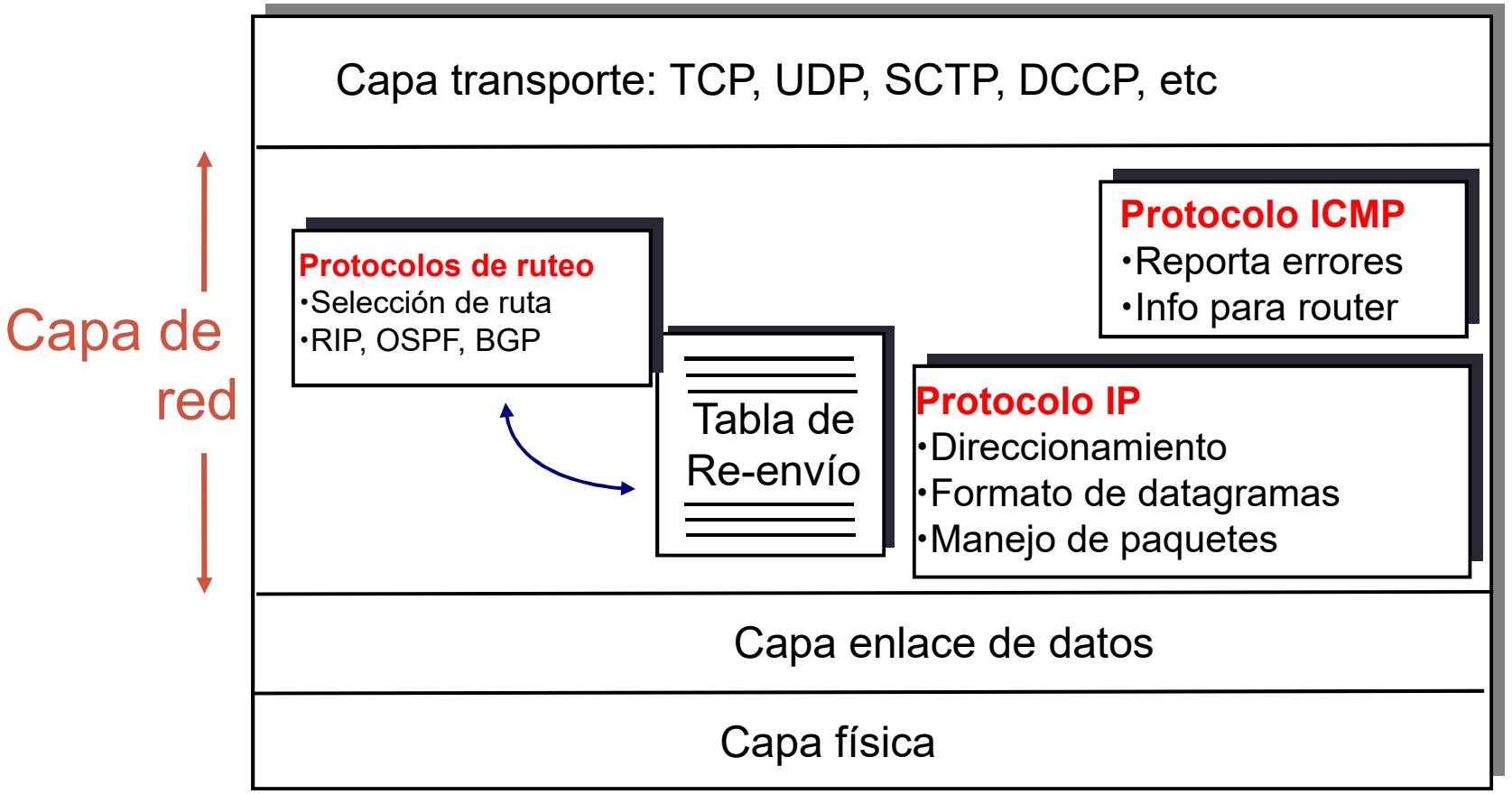
Clase 5

La Capa de Red

- Funciones claves / Modelos de servicio.
- Redes de Circuitos virtuales / Datagramas
- Interior de un Router
- IP: Internet Protocol
- Algoritmos de ruteo
  - Estado de enlace
  - Vector de Distancias
  - Ruteo Jerárquico
- Ruteo en Internet
  - RIP
  - OSPF
  - BGP
- Ruteo Broadcast y multicast

Capa de red en Internet

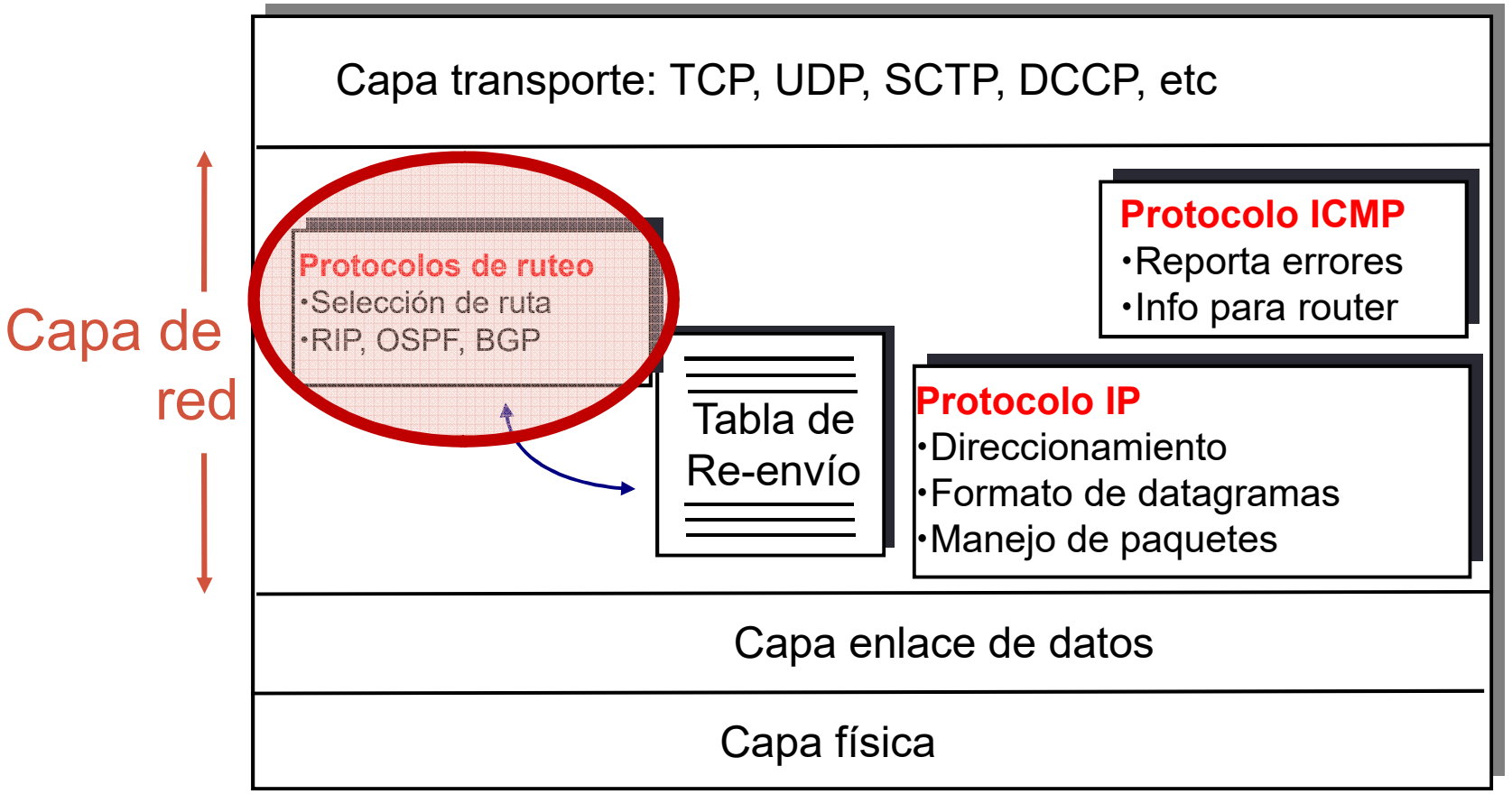
- Funciones de la capa de red en host y router :



SCTP: Stream Control Transmission Protocol (año 2000)
DCCP: Datagram Congestion Control Protocol (año 2006)

Capa de red en Internet

- Funciones de la capa de red en host y router :



SCTP: Stream Control Transmission Protocol (año 2000)
DCCP: Datagram Congestion Control Protocol (año 2006)

Algoritmo de enrutamiento

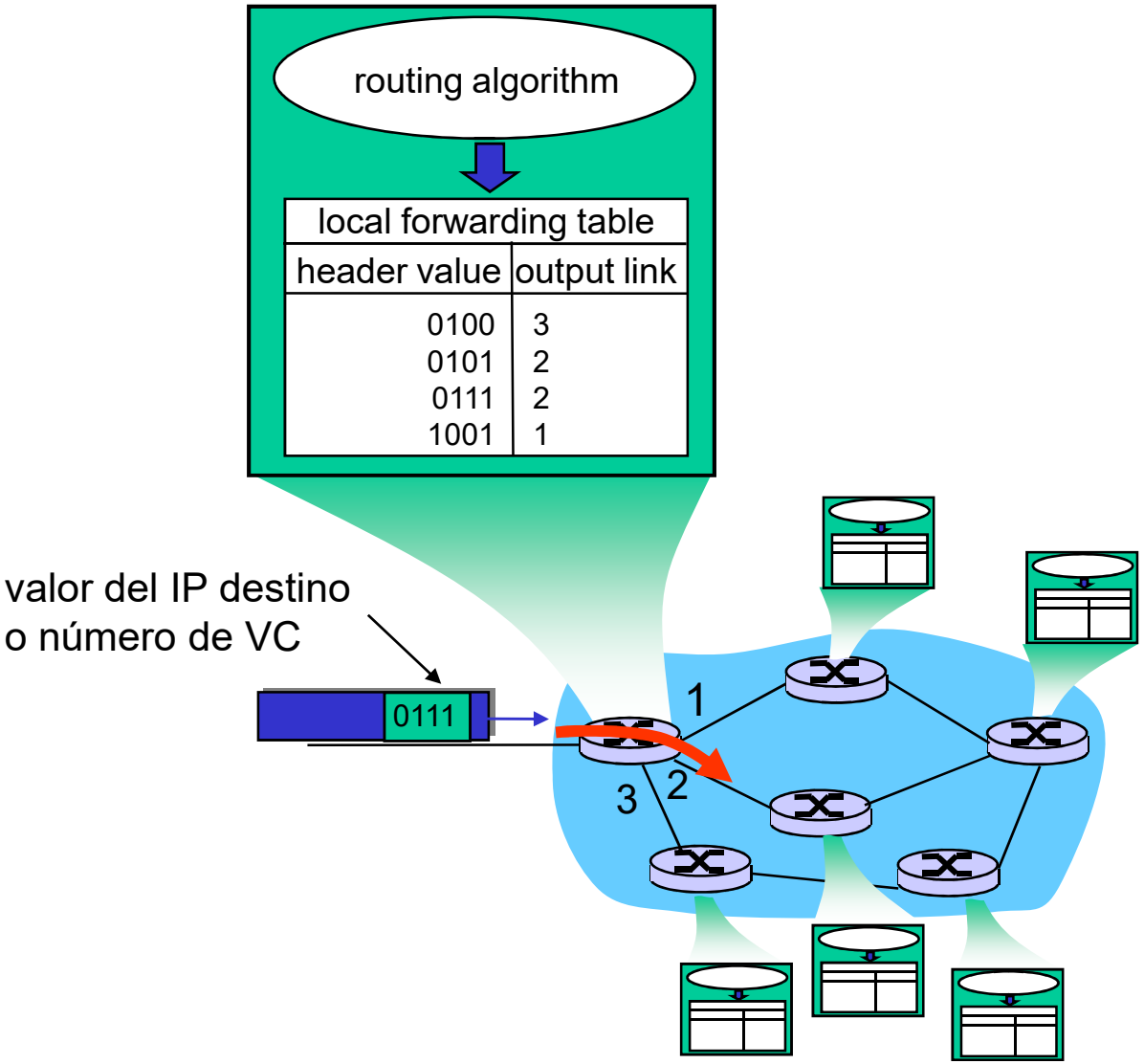
Su tarea consiste en

- Determinar buenas rutas desde los emisores hasta los receptores a través de la red de routers.
- Completar las tablas de reenvío de los routers

Tanto para …

- Redes de datagramas ➔ diferentes paquetes pueden tomar diferentes rutas.
- Redes de circuitos virtuales ➔ todos los paquetes seguirán la misma ruta.

Algoritmo de enrutamiento



Abstracción de la red vía un Grafo

- Para formular los problemas de enrutamiento se utilizan ***grafos*** - ***G*=(N,E)**
  - Conjunto de nodos (N)
  - Conjunto de aristas o uniones entre nodos (E)
- En el contexto de enrutamiento:
  - Nodo ➔ Routers
  - Aristas ➔ Enlaces físicos

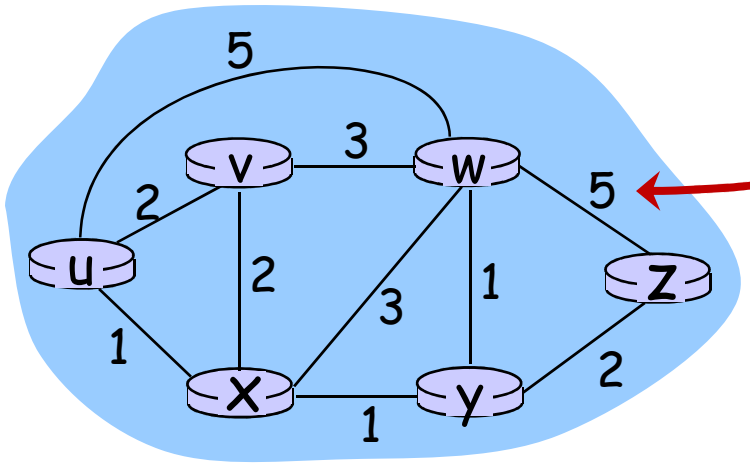
Abstracción de la red vía un Grafo

Grafo: **G = (N,E)**

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

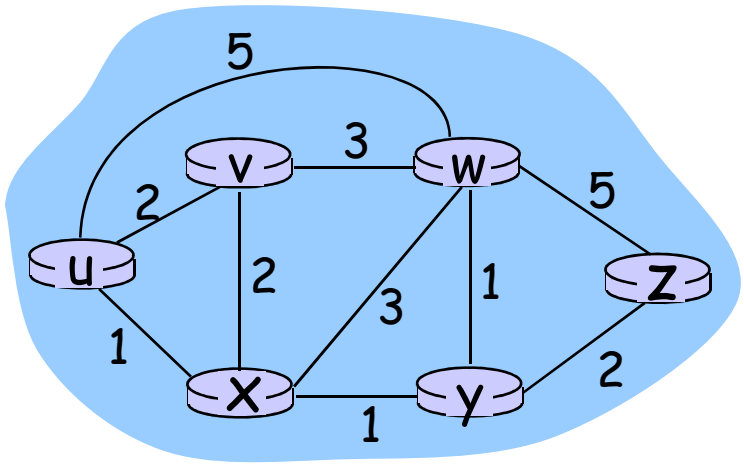
E = conjunto de enlaces ={ (u,v), (u,x), (u,w), (v,x), (v,w), (x,w), (w,y), (w,z), (y,z) }

Observación: Se considera que: (*x,y*) = (*y,x*)



Costo del enlace: longitud física del enlace, velocidad del enlace, costo monetario asociado, etc.

Abstracción de Grafos: costos



- Costo de enlace: *c(x, y)*- ej: c(w,z)=5
- Si no hay enlace físico: *c(x, y)*= ∞
- Enlaces no dirigidos*: c(x, y)=c(y, x)*
- Los nodos x e y se dicen **vecinos** si (x,y) pertenece a E.
- costo puede ser 1, inversamente relacionado al ancho de banda o directamente relacionado a 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: ¿Cuál es la ruta de mínimo costo entre u y w ?

Algoritmo de ruteo: algoritmo que encuentra el costo mínimo

Clasificación de los algoritmos de ruteo

- Global
- Descentralizado
- Estático
- Dinámico

Clasificación de los algoritmos de ruteo

- Global
  - Se basa en el conocimiento global y completo de la red (conectividad y costos de los enlaces).
  - Requiere que el algoritmo adquiera esta información antes de realizar el cálculo.
  - Se denominan algoritmos de “estado de enlace” (*link state* - LS)
- Descentralizado
- Estático
- Dinámico

Clasificación de los algoritmos de ruteo

- Global
- Descentralizado
  - Cada nodo conoce a priori **sólo** { los nodos vecinos y / los costos a los nodos vecinos.
  - Cada nodo calcula la ruta de costo mínimo a cada posible destino.
  - Proceso iterativo de cómputo e intercambio de información con sus vecinos
  - Algoritmo de “vector de distancia”
- Estático
- Dinámico

Clasificación de los algoritmos de ruteo

- Global
- Descentralizado
- Estático
  - Rutas cambian muy lentamente con el tiempo.
- Dinámico

Clasificación de los algoritmos de ruteo

- Global
- Descentralizado
- Estático
- Dinámico
  - Se modifican los caminos a medida que el tráfico o la topología de la red cambian.
  - Las actualizaciones pueden ser:
    - → Periódicas
    - → En respuesta a cambios en la red (costos/topología)

Clasificación de los algoritmos de ruteo

- Global
- Descentralizado
- Estático
- Dinámico

Los algoritmos dinámicos:

- ➔ Responden mejor a cambios en la red.
- ➔ Son más susceptibles a problemas (bucles u oscilaciones)

Algoritmo Dijkstra (LS)

- Algoritmo de estado de enlace (***Algoritmo Global***)
  - Topología y costos conocidos por todos los nodos
  - Se logra vía “difusión de estado de enlace”
  - Todos los nodos tienen la misma información
- Se calcula el camino de costo mínimo desde un nodo (fuente) a todos los demás nodos de la red
  - Iterativo.
  - Después de la ***k*-ésima** iteración se conoce el camino de menor costo a **k** nodos de destino.
  - Determina la tabla de re-envío para ese nodo.

Algoritmo Dijkstra (LS)

- A partir de un nodo de origen **u**, se definen:

  c(v,w): costo del enlace desde nodo v a w

  c(v,w) = ∞ → si no v y w no son vecinos directos

  D(v): valor actual del costo del camino desde origen (**u**) a destino **v**

  p(v): nodo anterior a **v** (vecino a **v**) en la ruta desde el origen (**u**) hasta **v**

  N': subconjunto de nodos cuyos caminos de costo mínimo ya se conocen

- Composición del Algoritmo:

  → Paso de inicialización

  → Bucle: se ejecuta tantas veces como nodos tenga la red.

Algoritmo Dijkstra (LS)

Algoritmo de LS para el nodo de origen u

Inicialización:

```
N' = {u}
for todos los nodos v
    if v es vecino de u
    then D(v) = c(u,v)
else D(v) = ∞
```

Notación:

c(v,w): costo del enlace desde nodo v a w.

D(v): valor actual del costo del camino desde fuente al destino v.

p(v): nodo anterior a v (vecino a v) en la ruta desde el origen a v.

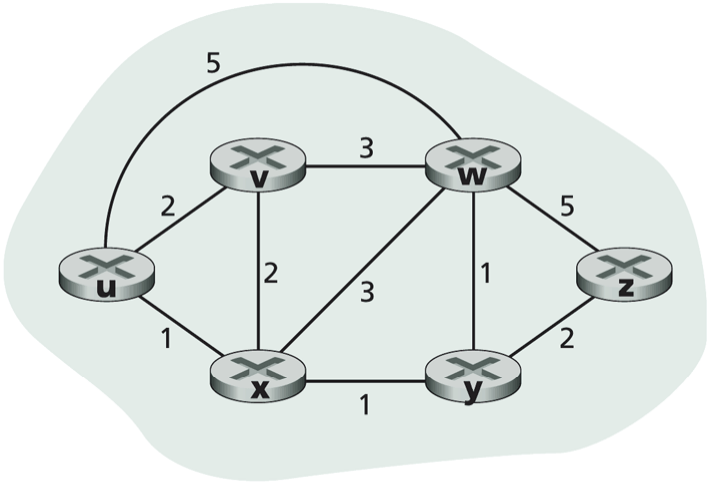
N': subconjunto de nodos cuyos caminos de costo mínimo ya se conocen.

Loop

```
encontrar w no perteneciente a N' tal que D(w) es un mínimo y agregar w a N'
actualiza D(v) para cada vecino v de w, que no está en N':
    D(v) = min( D(v), D(w) + c(w,v) )
    /* el nuevo costo a v es o bien el antiguo costo a v o el costo
    del camino más corto conocido a w más el costo de w a v */
```

until N' = N

Algoritmo de Dijsktra



Al finalizar el algoritmo se tiene:

- Para cada nodo su predecesor a lo largo de la ruta de costo mínimo.
- Se puede reconstruir el camino completo de dicha ruta.

step	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					

Table 4.3 ♦ Running the link-state algorithm on the network in Figure 4.25

Algoritmo de Dijsktra

Tabla de reenvío del nodo u:

Destino	Enlace
v	(u,v)
w	(u,x)
x	(u,x)
y	(u,x)
z	(u,x)

step	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					

Table 4.3 ♦ Running the link-state algorithm on the network in Figure 4.25

Algoritmo de Dijsktra

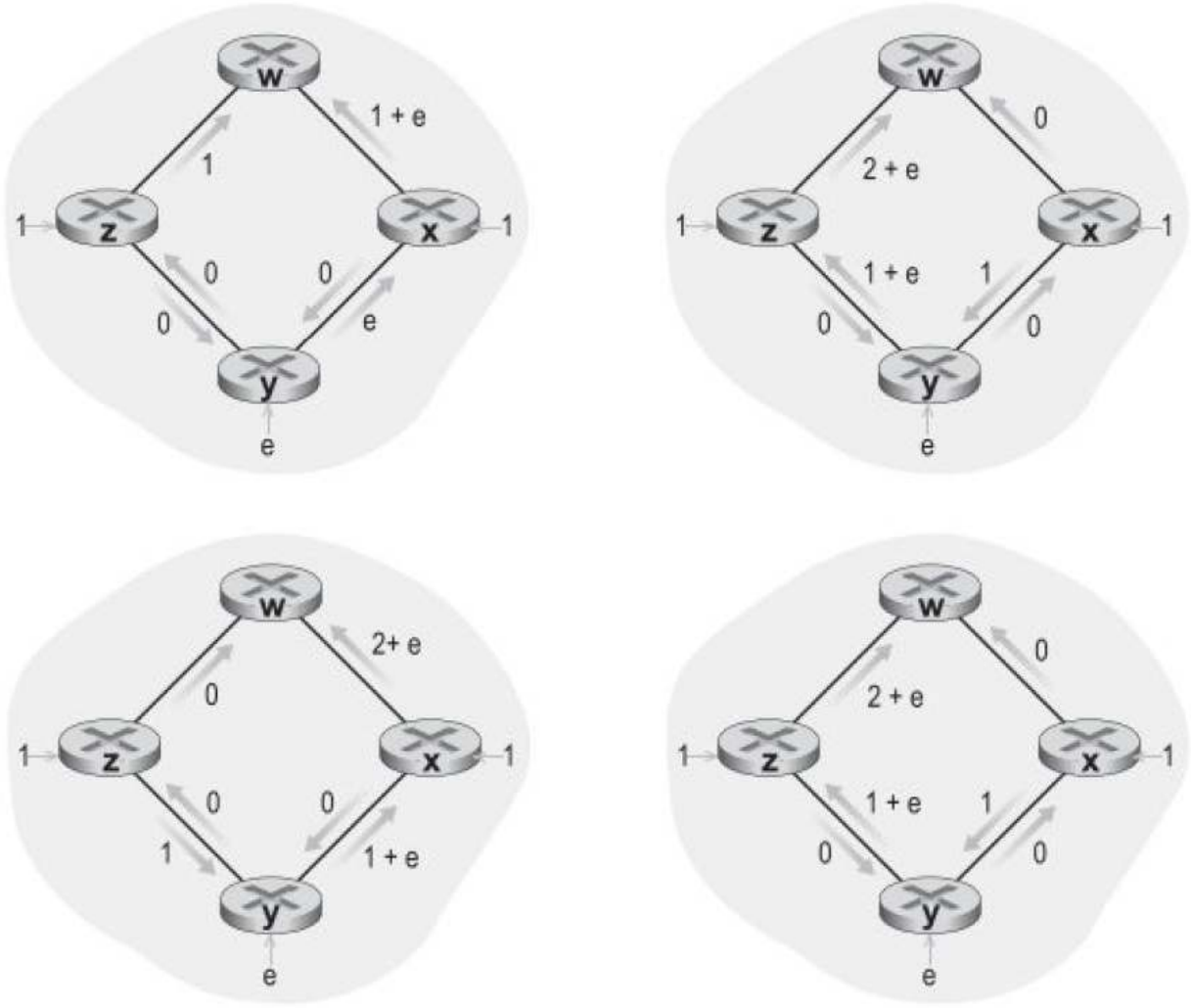
- Complejidad para n nodos
  - En cada iteración busca entre todos los nodos, no pertenecientes a N, el de menor costo.
  - En total se deben realizar n(n+1)/2 comparaciones: $O(n^2)$
  - Otras implementaciones son posibles: O(nlogn)
- Posibles oscilaciones en cálculos
  - Si costo enlace = cantidad de tráfico (carga) del enlace

Algoritmo de Dijsktra - Oscilaciones

Ejemplo

Costos no simétricos

Enrutamiento inicial ➔



Algoritmo Vector de Distancias (DV)

A diferencia del algoritmo de estado de enlace (LS), el algoritmo de vector distancia es:

Distribuido:

- Cada nodo recibe información de sus nodos vecinos.
- Cada nodo realiza sus cálculos y distribuye los resultados (solo a sus vecinos).

Iterativo:

- El proceso continúa hasta que no hay más información para ser intercambiada (finaliza por sí mismo).

Asincrónico:

- No requiere que todos los nodos operen sincronizados entre sí.

Algoritmo Vector de Distancias (DV)

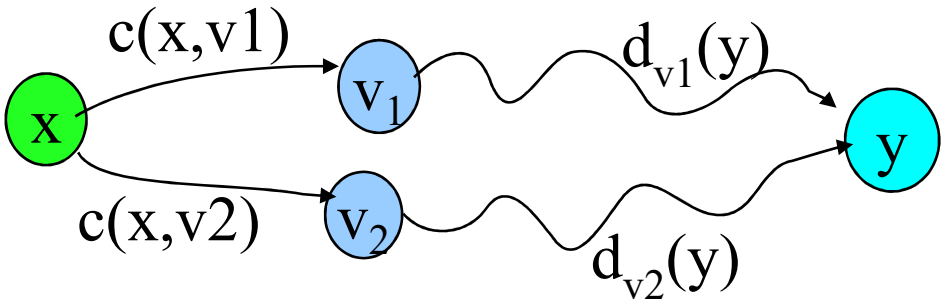
Ecuación de Bellman-Ford

$d_x(y)$:= costo de la ruta de menor costo desde x hasta y

Entonces:

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

de todos los vecinos (v) de x



Algoritmo Vector de Distancias (DV)

Costo de la ruta de menor costo desde **u** hasta **z**:

$$\mathbf{d_u(z) = \min_v \{c(u,v) + d_v(z)\}}$$

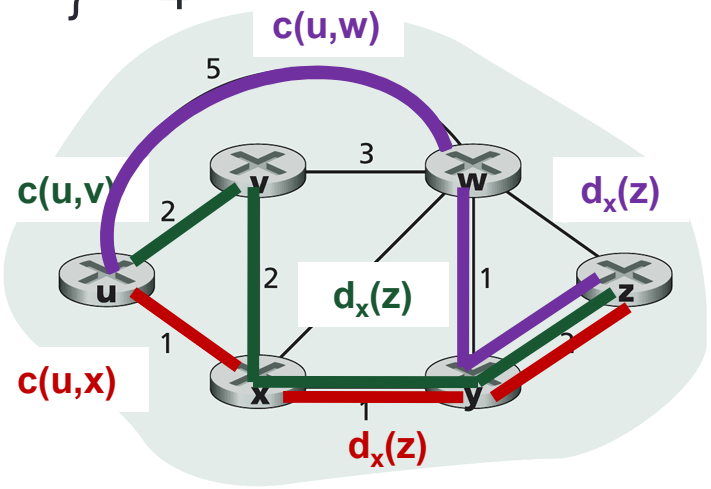
Vecinos de u: x, v y w ➔ observamos $d_x(z)=3$; $d_v(z)=5$; $d_w(z)=3$

A partir de la ecuación de Bellman-Ford:

$d_u(y)= \min \{c(u,x)+d_x(z)\ \ c(u,v)+d_v(z)\ \ c(u,w)+d_w(z)\}=$

min { 1+3 ; 2+5 ; 5+3 } =

min { 4 ; 7 ; 8 } = 4



Algoritmo Vector de Distancias (DV)

Idea básica: *Cada nodo x:*

- Comienza con una estimación del costo mínimo al nodo y (para todo y ϵ N)
- Arma un VD con los costos mínimos estimados a todos los nodos de N.
- Envía periódicamente su VD a sus vecinos.
- Cuando recibe un nuevo VD de un vecino ➔ actualiza el propio (ec de B-F)

$$D_x(y) = \min_v\{c(x,v) + D_v(y)\} \quad \text{para cada vecino } v, \text{ para cada nodo } y \text{ en } N$$

- Si el VD cambia ➔ lo envía a sus vecinos y ellos a su vez actualizan sus VD.
- Bajo condiciones normales, el valor estimado $D_x(y)$ converge al menor costo real $d_x(y)$.

Algoritmo Vector de Distancias (DV)

Nodo x:

- $D_x(y)$ = costo mínimo estimado de x a y
- Vector de distancia: $\mathbf{D}_x = [D_x(y): y \in N]$
- Cada nodo x conoce el costo a todos sus vecinos v: $c(x,v)$
- Cada nodo x mantiene la siguiente información:
  - El costo a todos sus vecinos v: $c(x,v)$
  - Su propio vector distancia: $\mathbf{D}_x = [D_x(y): y \in N]$
  - Los vectores de distancia de sus vecinos v: $\mathbf{D}_v = [D_v(y): y \in N]$

Algoritmo Vector de Distancias (DV)

En cada nodo x:

Inicialización:

for todos los destinos y pertenecientes a N:

$D_x(y) = c(x,y)$ /* si y no es un vecino, entonces $c(x,y) = \infty$ */

for cada vecino w

$D_w(y) = ?$ para todos los destinos y pertenecientes a N

for cada vecino w

enviar vector distancia $\mathbf{D_x} = [D_x(y)$: y perteneciente a N] a w

Loop

wait (hasta variac. del costo de un vecino w o recibir un vector distancia de vecino w)

for cada y pertenecientes a N:

$D_x(y) = \min_v (c(x,v) + D_v(y))$

if $D_x(y)$ varía para cualquier destino y

enviar vector distancia $\mathbf{D_x} = [D_x(y)$: y perteneciente a N] a todos los vecino

forever

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

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

Ejemplo: Vector de Distancias

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

node y table

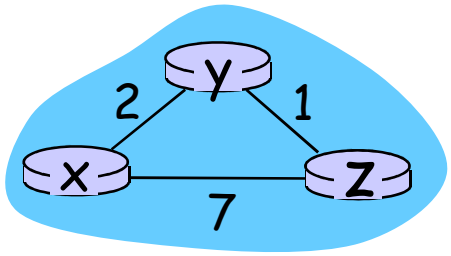
cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



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

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

node x table

from	cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	cost to x	y	z
x			
y	2	0	1
z	7	1	0

node y table

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

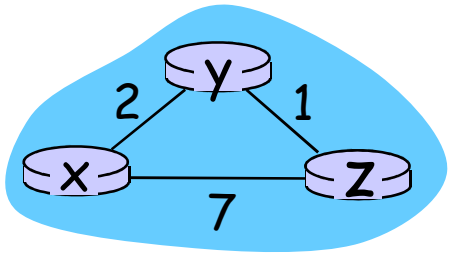
from	cost to x	y	z
x	0	2	7
y			
z	7	1	0

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			

time

Ejemplo: Vector de Distancias



Ejemplo: Vector de Distancias

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

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

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

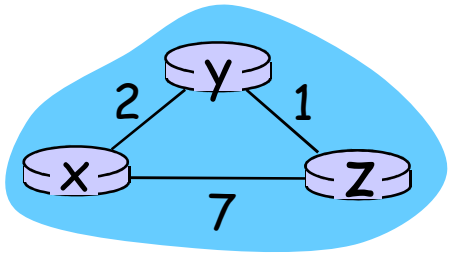
from	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

time



Ejemplo: Vector de Distancias

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

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

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

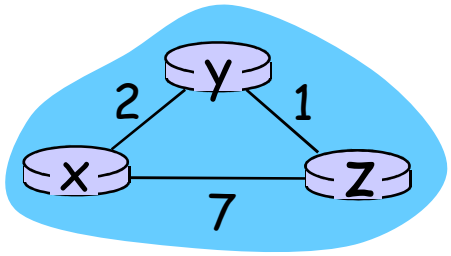
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

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

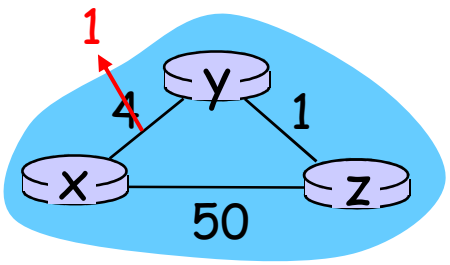
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time



Algoritmo Vector de Distancias (DV)

- Cambios en costos de enlaces:
- nodo detecta un cambio de costo en uno de sus enlaces
- actualiza información de ruteo, recalcula vector de distancia
- si hay cambio en DV notifica a sus vecinos



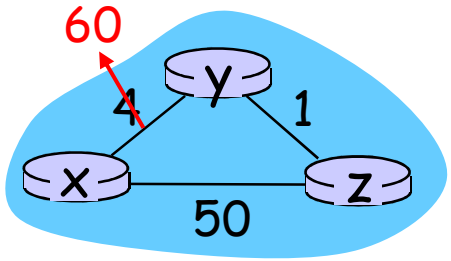
En el tiempo t_0, ***y*** detecta un cambio en costo de enlace, actualiza su DV e informa a sus vecinos.

En el tiempo t_1, ***z*** recibe la información de ***y***, también actualiza su tabla. Calcula un nuevo costo para ***x*** y le envía su Vector a sus vecinos.

En el tiempo t_2, ***y*** recibe la actualización de ***z*** y actualiza su tabla de distancia. Los costos mínimos de ***y*** no cambian, ***y*** no envía ningún nuevo mensaje a ***z***.

Algoritmo Vector de Distancias (DV)

- Cambio en costos de enlaces:
- buenas noticias viajan rápido
- noticias malas viajan lento
- ¿Cómo pasa esto?



Algoritmo Vector de Distancias (DV)

- Inicialmente:

  $D_y(x) = 4 \;-\; D_y(z) = 1 \;-\; D_z(x) = 5 \;-\; D_z(y) = 1$

- En t_0: ➔ cambia el costo de (x,y)

  ➔ el nodo y lo detecta y calcula:

  $D_y(x) = \min \{ c(y,x)+D_x(x) \; ; \; c(y,z)+D_z(x) \} =$

  $= \min \{ 60 + 0 \; ; \; 1 + 5 \} = 6$

Esto ocurre porque la última información que el nodo y tiene es que z llega a x con un costo de 5.

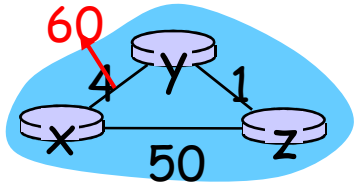
- En t_1: ➔ y tiene un nuevo $D_y(x) = 6$.

  ➔ y informa de su VD a sus vecinos.

node y table

En t_0

from \	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0



node y table

En t_1

from \	x	y	z
x	0	4	5
y	6	0	1
z	5	1	0

Algoritmo Vector de Distancias (DV)

- En t_2: ➔ z recibe el nuevo $\mathbf{D_Y}$ y recalcula $\mathbf{D_z}$.

  $D_Z(x) = \min \{50 + 0, 1 + 6\} = 7$

- En t_3: ➔ z informa a y de su nuevo vector $\mathbf{D_Z}$.

  ➔ y recibe $\mathbf{D_Z}$, recalcula un nuevo $D_y(x)$

  etc...el proceso se repite por 44 iteraciones!

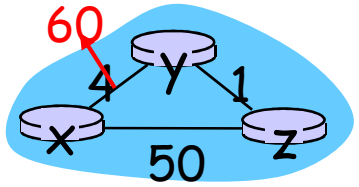
node y table

En t_2		x	y	z
from	x	0	4	5
	y	6	0	1
	z	7	1	0

Esto ocurre porque

➔ y rutea a través de z para llegar a x

➔ z rutea a través de y para llegar a x



Un paquete que cae en un bucle de ruteo va a rebotar entre los dos routers para siempre (en IP muere por TTL).

node y table

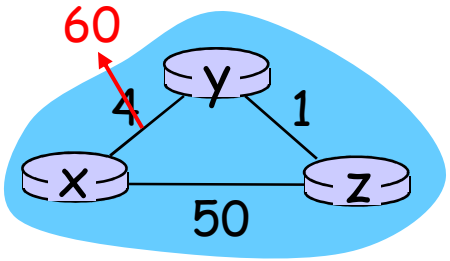
En t_3		x	y	z
from	x	0	4	5
	y	8	0	1
	z	7	1	0

Algoritmo Vector de Distancias (DV)

- Cambio en costos de enlaces:
- ¿Qué pasa si el enlace se cae? Su costo es ∞ ➔ loop infinito.

Reversa envenenada:

- Si Z routea a través de Y para llegar a X:
  - Z le dice a Y que su distancia a X es infinita (para que Y no rutee a X vía Z)
- ¿Resuelve completamente el problema de contar hasta el infinito? Sí, en este caso sencillo!
  - Los bucles de tres o más nodos no son detectados con esta técnica.



Comparación de algoritmos de estado de enlace (LS) y vector de distancias (DV)

Complejidad de mensajes

- LS: (n nodos y E enlaces) O(nE) mensajes son enviados
- DV: sólo intercambios entre vecinos

Rapidez de convergencia

- LS: $O(n^2)$
  - Puede tener oscilaciones
- DV: el tiempo de convergencia varía:
  - Podría entrar en loops
  - Problema de cuenta infinita

Robustez

¿qué pasa si un router funciona mal?

LS:

- Nodos pueden comunicar costo incorrecto del *link*
- Cada nodo computa sólo su propia tabla

DV:

- DV nodo puede comunicar costo de *camino* incorrecto
- La tabla de cada nodo es usada por otros
  - error se propaga a través de la red

Comparación de algoritmos de estado de enlace (LS) y vector de distancias (DV)

Desventaja y ventaja de estado de enlace versus vector de distancias.

- Desventaja: Estado de enlace requiere propagar anticipadamente la información de cada enlace a todos los nodos de la red.
- Ventaja: Estado de enlace converge rápidamente una vez que un enlace cambia su costo y éste ha sido propagado.

Ruteo Jerárquico

- Nuestro estudio del ruteo hasta ahora es idealizado. Suponemos que:
- Todos los routers son idénticos
- La red es “plana”
- … esto *no es* verdad en la práctica

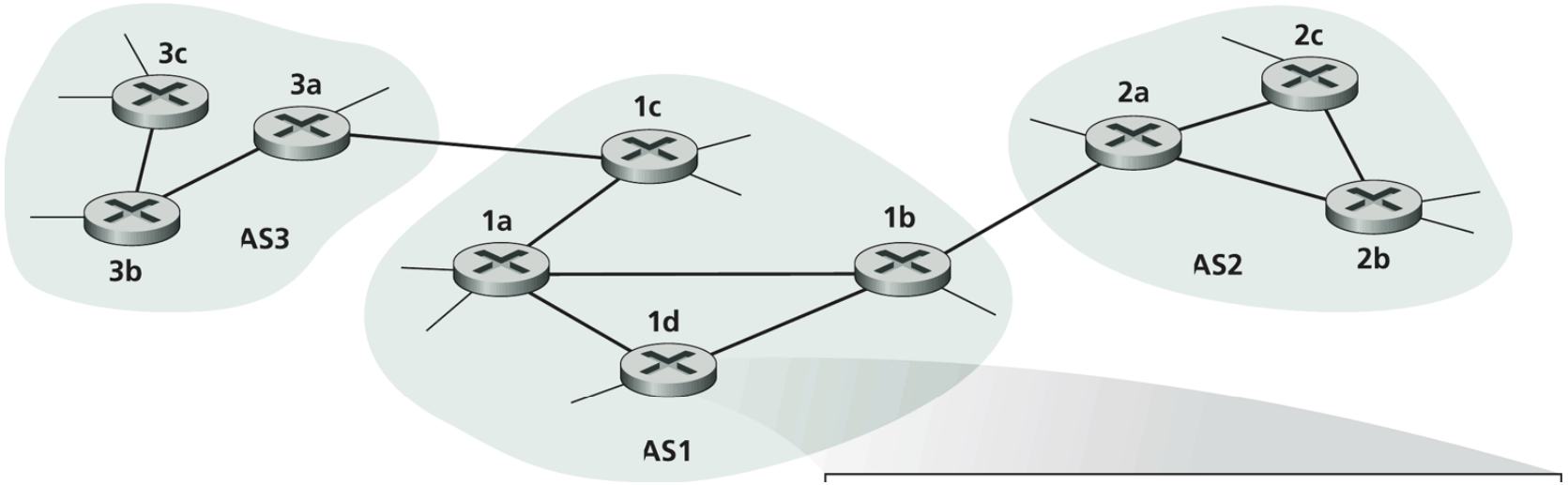
Escala: con 200 millones de destinos:

- No podemos almacenar todos los destinos en tablas de ruteo!
- Los intercambios de tablas de ruteo inundarían los enlaces!

Autonomía administrativa

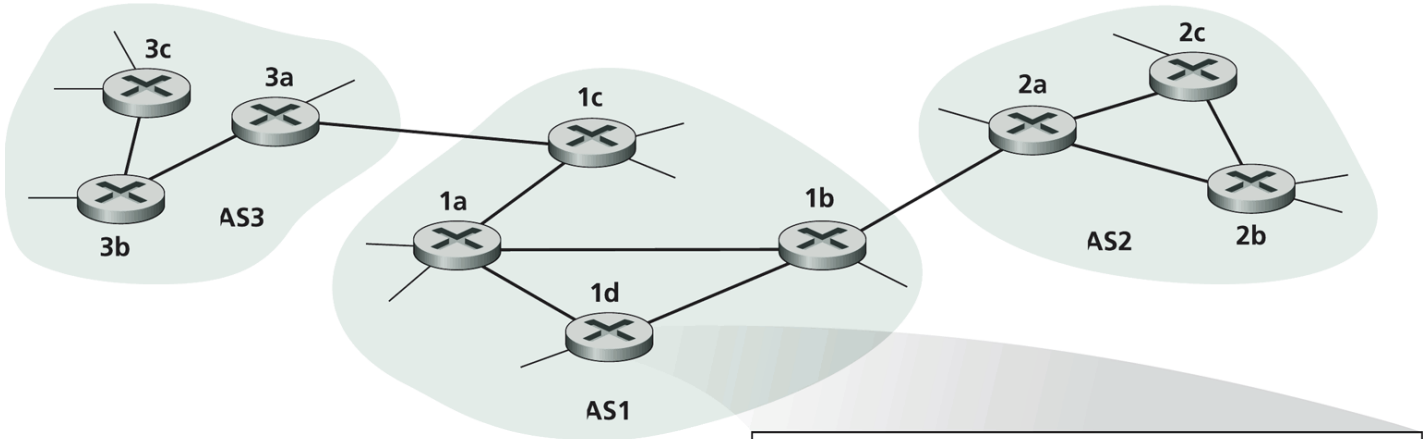
- Internet = red de redes
- Cada administrador de red puede querer controlar el ruteo en su propia red

Ruteo Jerárquico



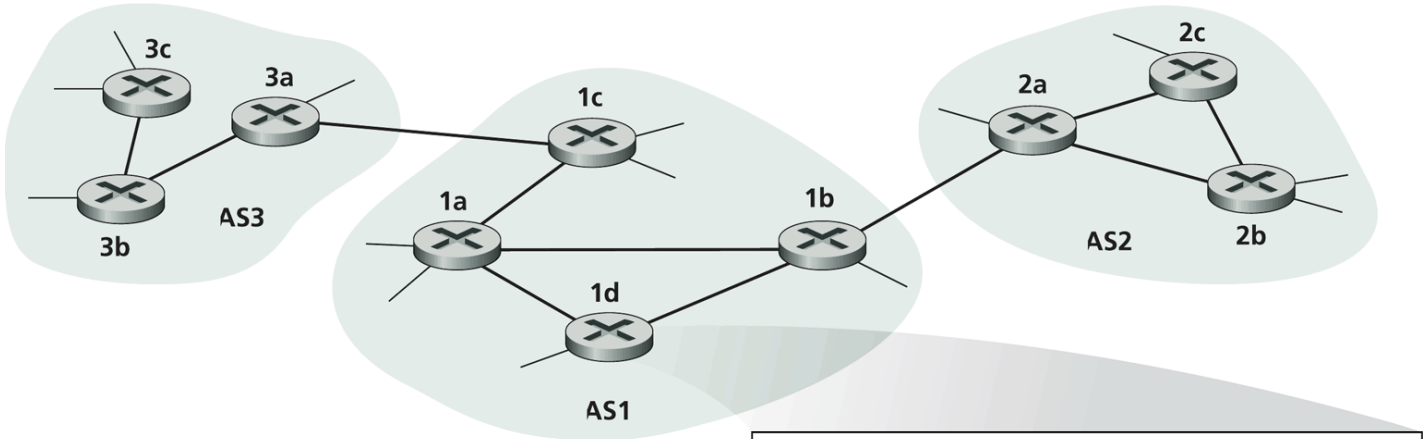
- Agrupar routers en regiones, ***"sistemas autónomos" (AS).***
- Routers en el mismo AS usan el mismo protocolo de ruteo.
  - Protocolo de ruteo ***interno del AS (intra-AS).***
- Routers en diferentes AS pueden correr diferentes protocolos de ruteo.
- Router de borde ***(Gateway router)***
  - Tienen enlace directo a routers en ***otros AS***.

Ruteo Jerárquico



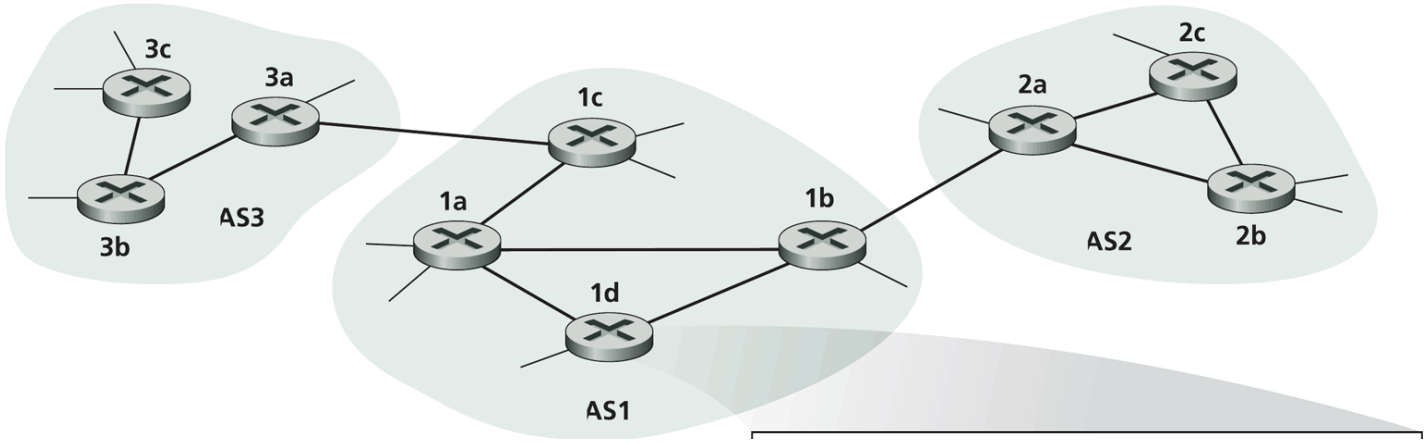
- ¿Cómo puede un router enrutar un paquete con destino fuera de su AS?
  - AS con **1 sólo router de gateway**.
  - AS con **varios routers de gateway**.
    - ➔ Protocolo de enrutamiento ***entre AS (inter-AS).***

Ruteo Jerárquico



- Caso 1: *AS con 1 solo router de gateway*
  - Cada router sabe como reenviar el paquete al router de gateway gracias al algoritmo de enrutamiento interno del AS.
  - El router de gateway reenvía el paquete al enlace que conecta con el exterior del AS.
  - El otro AS toma la responsabilidad de enrutar el paquete.

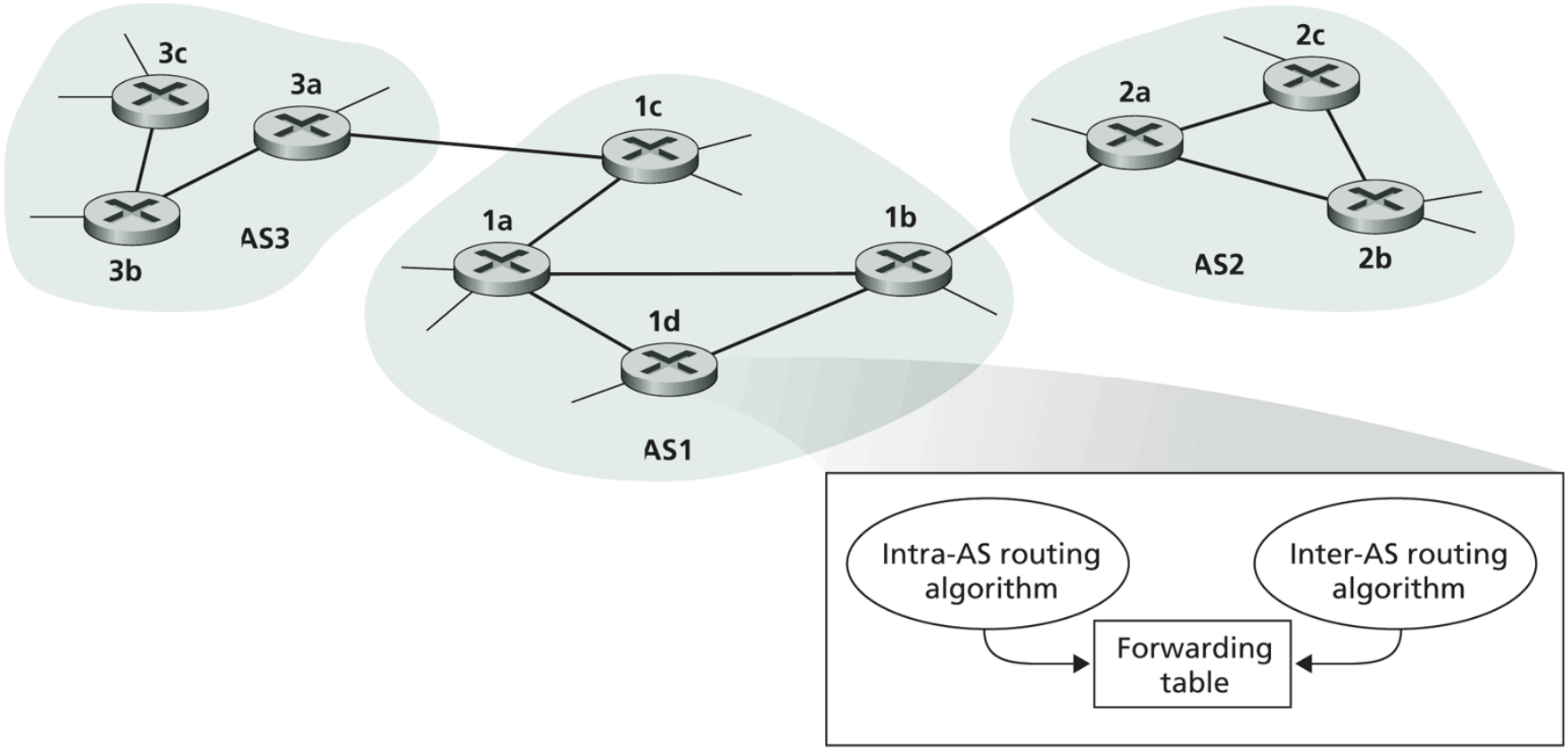
Ruteo Jerárquico



- Caso 2: AS con más de un router de gateway.
  - El problema de saber a dónde enviar el paquete: ***más complicado***.
  - El AS tiene que :
    - Aprender qué destinos son alcanzables a través de cada AS vecino.
    - Propagar esa información a todos los routers internos al AS.

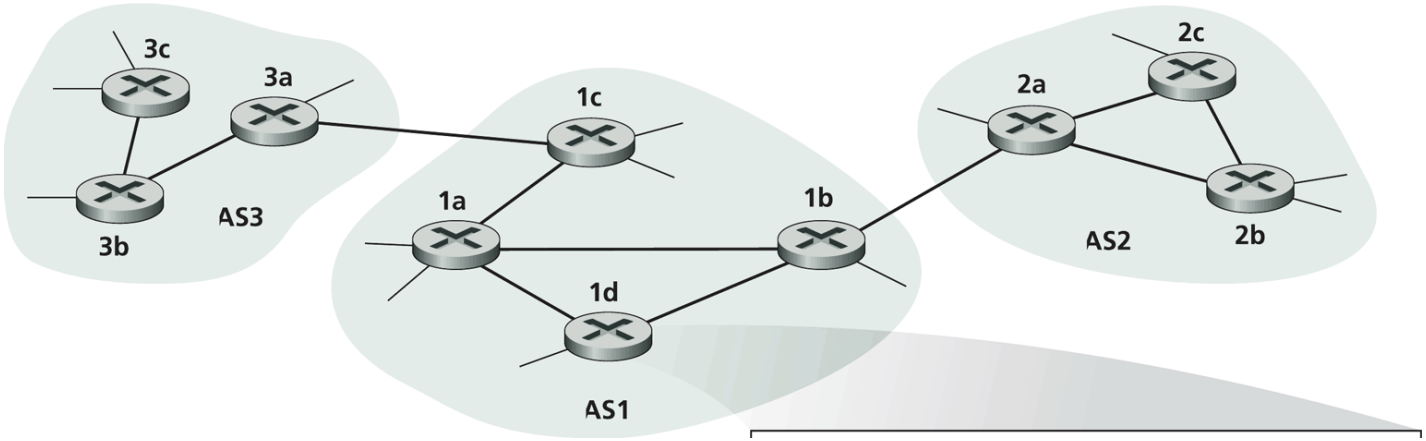
➔ **Protocolo de enrutamiento entre AS *(inter-AS).***

Ruteo Jerárquico



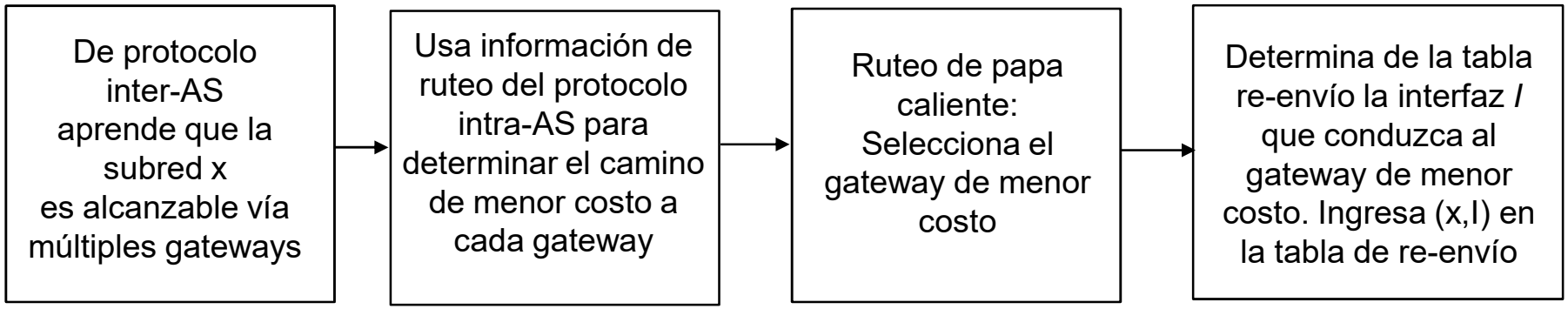
Ejemplo: tabla de re-envío en el router 1d

- Supongamos que AS1 sabe por el protocolo ***inter-AS*** que la subred *x* es alcanzable desde AS3 (gateway 1c) pero no desde AS2.
- El protocolo ***inter-AS*** propaga la información de alcance a todos los routers internos.
- Router **1d** determina de la información de ruteo ***intra-AS*** que su interfaz *I* está en el camino de costo mínimo a 1c.
- Luego éste pone en su tabla de re-envío: *(x, I)*.



Ejemplo: Elección entre múltiples AS

- Ahora supongamos que AS1 sabe por el protocolo inter-AS que la subred *x* es alcanzable desde AS3 *y* desde AS2.
- Para configurar la tabla de re-envío, el router 1d debe determinar hacia qué gateway éste debería re-enviar los paquetes destinados a x.
- Ésta es también una tarea del protocolo de ruteo inter-AS
- Ruteo de la papa caliente (Hot potato routing): enviar el paquete hacia el router más cercano de los dos.

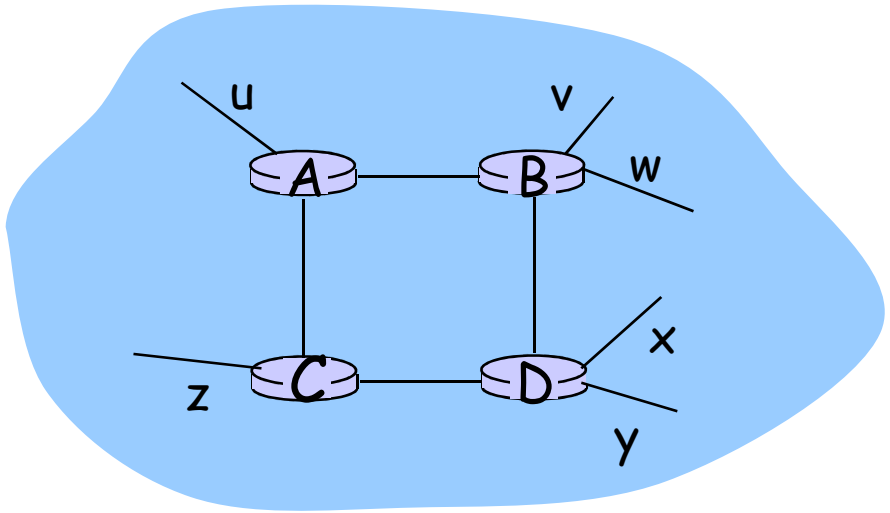


Ruteo Intra-AS

- Ya hemos visto los algoritmos de ruteo, veremos ahora cómo son aplicados en Internet.
- Dentro del sistema autónomo (AS: autonomous systems)
  - También son conocidos como Interior Gateway Protocols (IGP)
- Protocolos de ruteo Internos a los AS más comunes:
  - RIP: Routing Information Protocol (vector-distancia)
  - OSPF: Open Shortest Path First (Estado de enlace - Dijkstra)
  - IGRP: Interior Gateway Routing Protocol (propietario de Cisco)

RIP (Routing Information Protocol)

- Algoritmo de **vector de distancia**
- Incluido en BSD-UNIX en 1982
- Métrica de distancia: # de hops (máx = 15 hops)



Destino desde A	hops
u	1
v	2
w	2
x	3
y	3
z	2

Avisos de RIP

- Vector de Distancia: intercambia avisos entre vecinos cada 30 seg vía mensajes de respuesta RIP (también conocidos como avisos RIP)
- Cada aviso: lista de hasta 25 redes destinos dentro del AS
- La métrica de costo usada es el número de hops, es decir, cada enlace tiene costo unitario.
- Número de hops: es el número de subredes atravesadas desde la fuente a la subred del destino, incluyendo esta última.

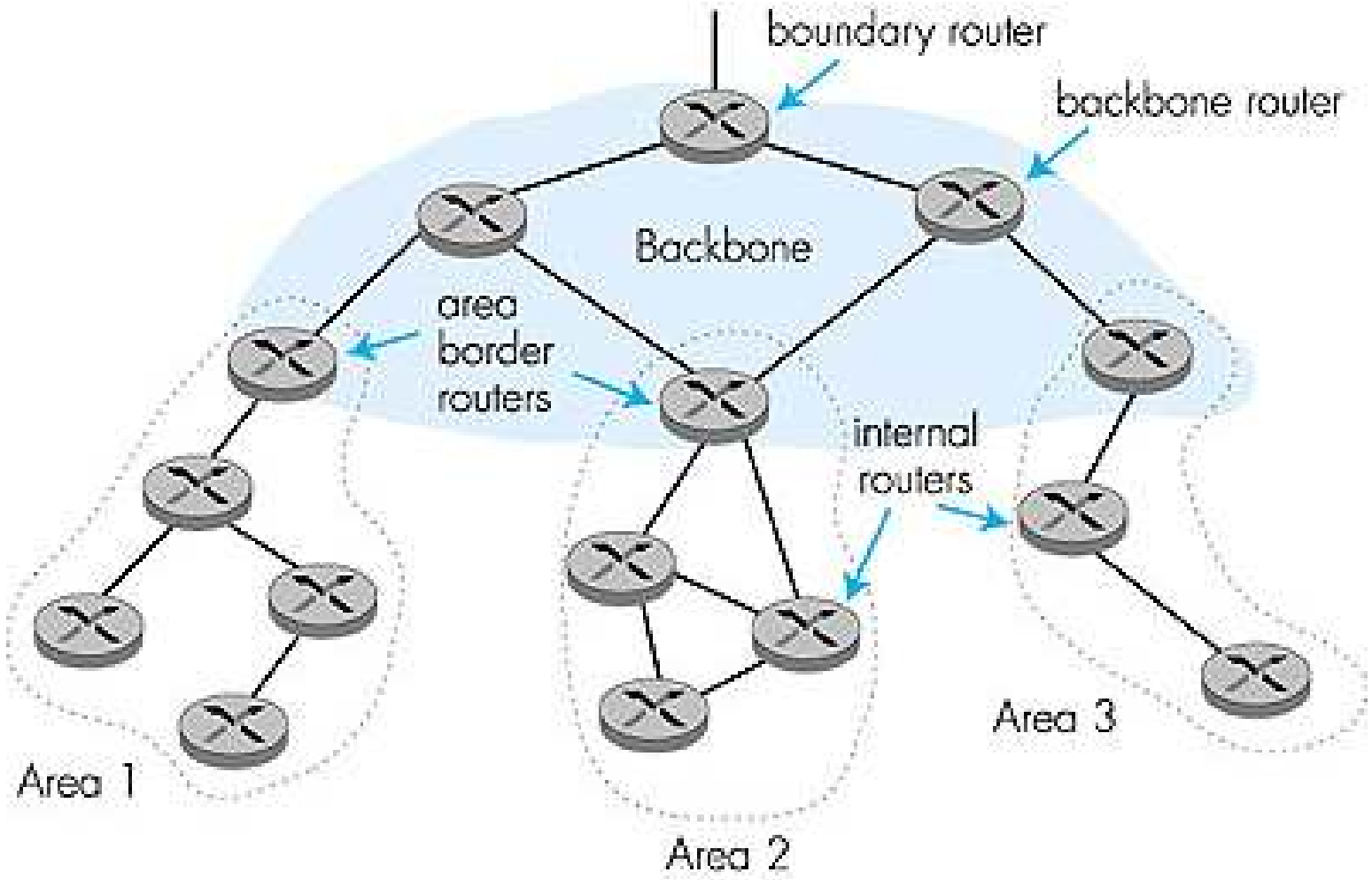
OSPF (Open Shortest Path First)

- "open": públicamente disponible.
- Usa algoritmo de **estado de enlace**.
  - Se difunden paquetes de estado de enlace.
  - Se crea un mapa de la topología en cada nodo.
  - Las rutas se calculan usando el algoritmo de **Dijkstra.**
- Avisos OSPF transportan una entrada por cada router vecino.
- Avisos son difundidos al sistema autónomo entero (vía inundación).
  - Mensajes OSPF son transportados directamente sobre IP (en lugar de TCP o UDP).

OSPF características “avanzadas” (no en RIP)

- Seguridad: todos los mensajes OSPF son autenticados (para prevenir intrusos).
- Múltiples caminos de igual costo son permitidos (sólo un camino en RIP).
- Para cada enlace, hay múltiples métricas de costo para diferentes tipos de servicios (TOS) (e.g., en un enlace satelital se asigna costo “bajo” para servicio de mejor esfuerzo; y costo alto para tiempo real).
- Soporte integrado para uni- y multicast:
  - Multicast OSPF (MOSPF) usa la misma base de datos de la topología que OSPF.
- En dominios grandes se puede usar OSPF Jerárquico.

OSPF Jerárquico

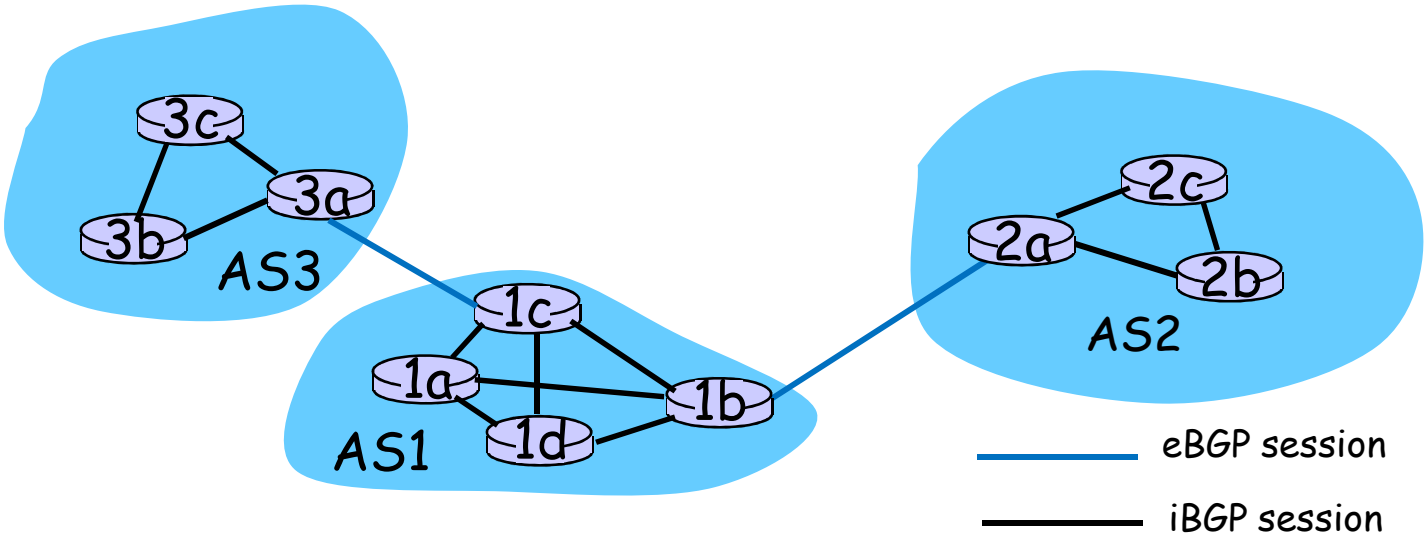


Ruteo inter-AS en internet: BGP

- BGP (Border Gateway Protocol): Estándar por “de facto”
- BGP provee a cada AS un medio para:
  1. Obtener la información de alcanzabilidad de una subred desde sus AS’s vecinos.
  2. Propaga la información de alcanzabilidad a todos los routers internos al AS.
  3. Determina rutas “buenas” a subredes basados en información de alcanzabilidad y políticas.
- Permite a una subred dar aviso de su existencia al resto de la Internet.

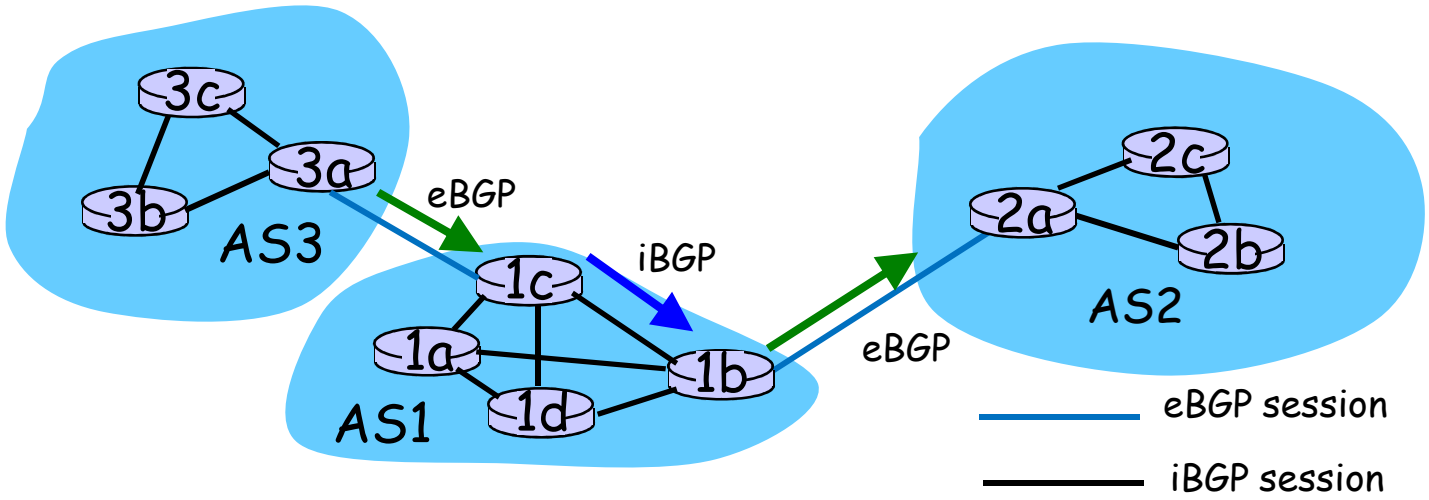
Introducción a BGP

- Pares de routers BGP intercambian información de ruteo sobre conexiones TCP semi-permanentes: sesiones BGP.
- Cuando AS2 anuncia un prefijo a AS1, AS2 está prometiendo que va a reenviar cualquier datagrama destinado a ese prefijo.
- AS2 puede reunir prefijos en su anuncio: se informa prefijo común.



Distribución de información de alcanzabilidad

- Con una sesión eBGP entre 3a y 1c, AS3 envía información de alcanzabilidad de prefijo (e.g 138.16.67/24) a AS1.
- 1c puede usar iBGP para distribuir este nuevo alcance de prefijo a todos los routers en AS1
- 1b puede entonces re-anunciar la información de alcance a AS2 a través de la sesión eBGP entre 1b y 2a
- Cuando un router aprende del nuevo prefijo, crea una entrada para ese prefijo en su tabla de re-envío.



¿Por qué la diferencia entre ruteo Intra- e Inter- AS?

Por política:

- Inter-AS: administradores desean control sobre cómo su tráfico es ruteado y quién rutea a través de su red.
- Intra-AS: administrador único, no se requieren decisiones de política

Escala:

- Ruteo jerárquico ahorra tamaño en tablas, y reduce tráfico en actualizaciones

Desempeño:

- Intra-AS: Se puede focalizar en alto desempeño.
- Inter-AS: políticas pueden dominar sobre desempeño.