# Capítulo 5: Capa Red:Plano de control

## ELO322: Redes de Computadores
Agustín J. González

Este material está basado en:

- Material de apoyo al texto *Computer Networking: A Top Down Approach Featuring the Internet*. Jim Kurose, Keith Ross.

# Capítulo 5: Capa de Red, Plano de Control

Objetivos del Capítulo: Entender los principios del plano de control

- Algoritmo de ruteo tradicional
- Controladores SDN
- ICMP: Internet Control Message Protocol
- Administración de la red

Y sus implementaciones en Internet:

- OSPF, BGP, OpenFlow, controladores ODL y ONOS, ICMP, SNMP
- No todo será cubierto por este curso

# Capítulo 5: Capa de Red: Plano de Control

- 5.1 Introducción
- 5.2 Protocolos de ruteo
  - Estado de enlace
  - Vector de Distancia
- 5.3 Ruteo dentro de sistemas autónomos en la Internet: OSPF
- 5.4 Ruteo entre ISPs: BGP
- 5.5 Plano de control de SDN

Otras secciones del capítulo no son cubiertas en este curso

# Funciones de la capa de red

- Recordar: dos funciones de la capa de red:
- Re-envío (forwarding): mover paquetes desde una entrada del router a la salida apropiada

*Plano de datos*

- Ruteo (routing): determinar la ruta a ser tomada por paquetes desde la fuente al destino

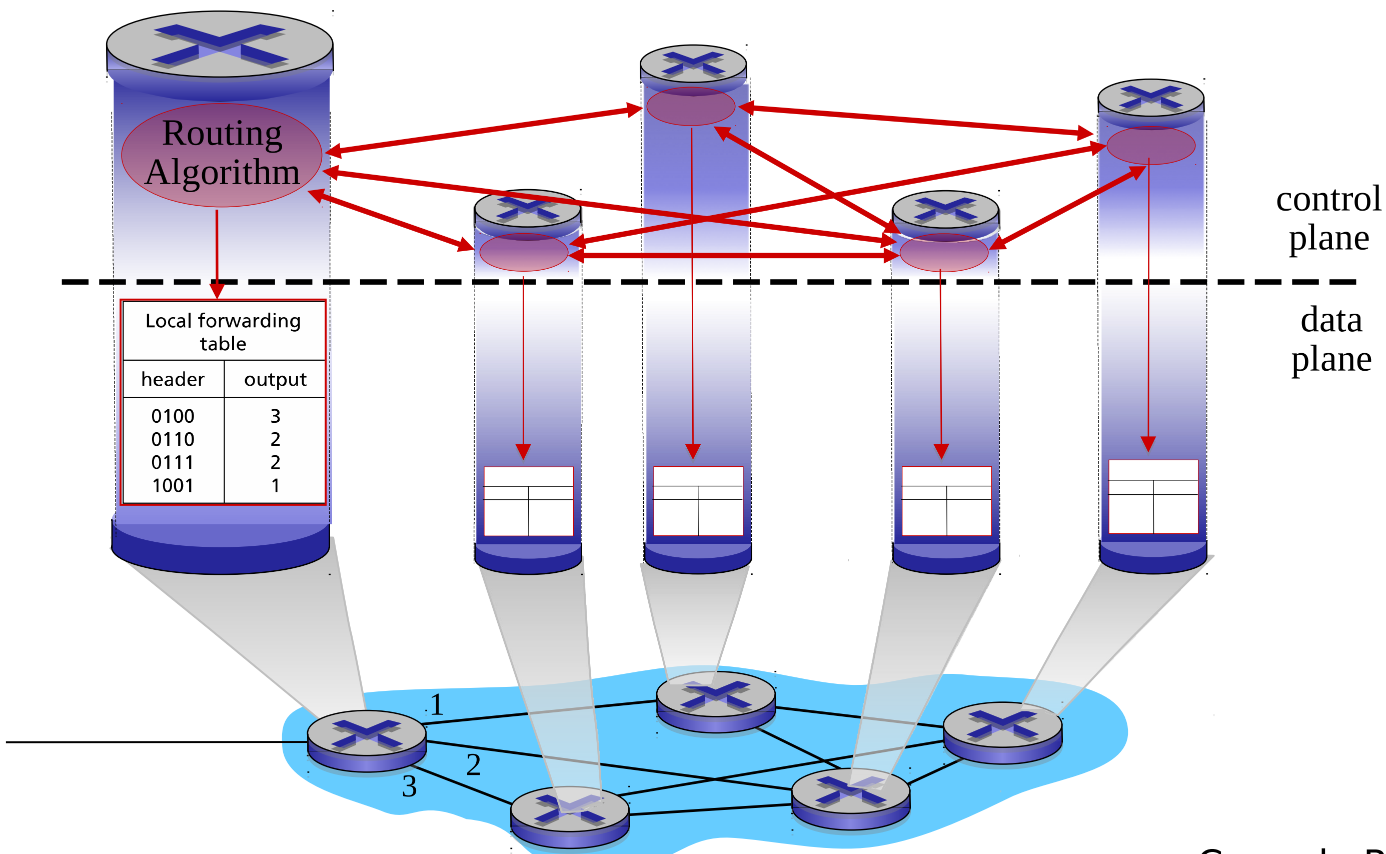
*control plane*

Dos estrategias para estructurar el plano de control:

- Control por router (tradicional)
- Control logicamente centralizado (SDN: Software defined networking)

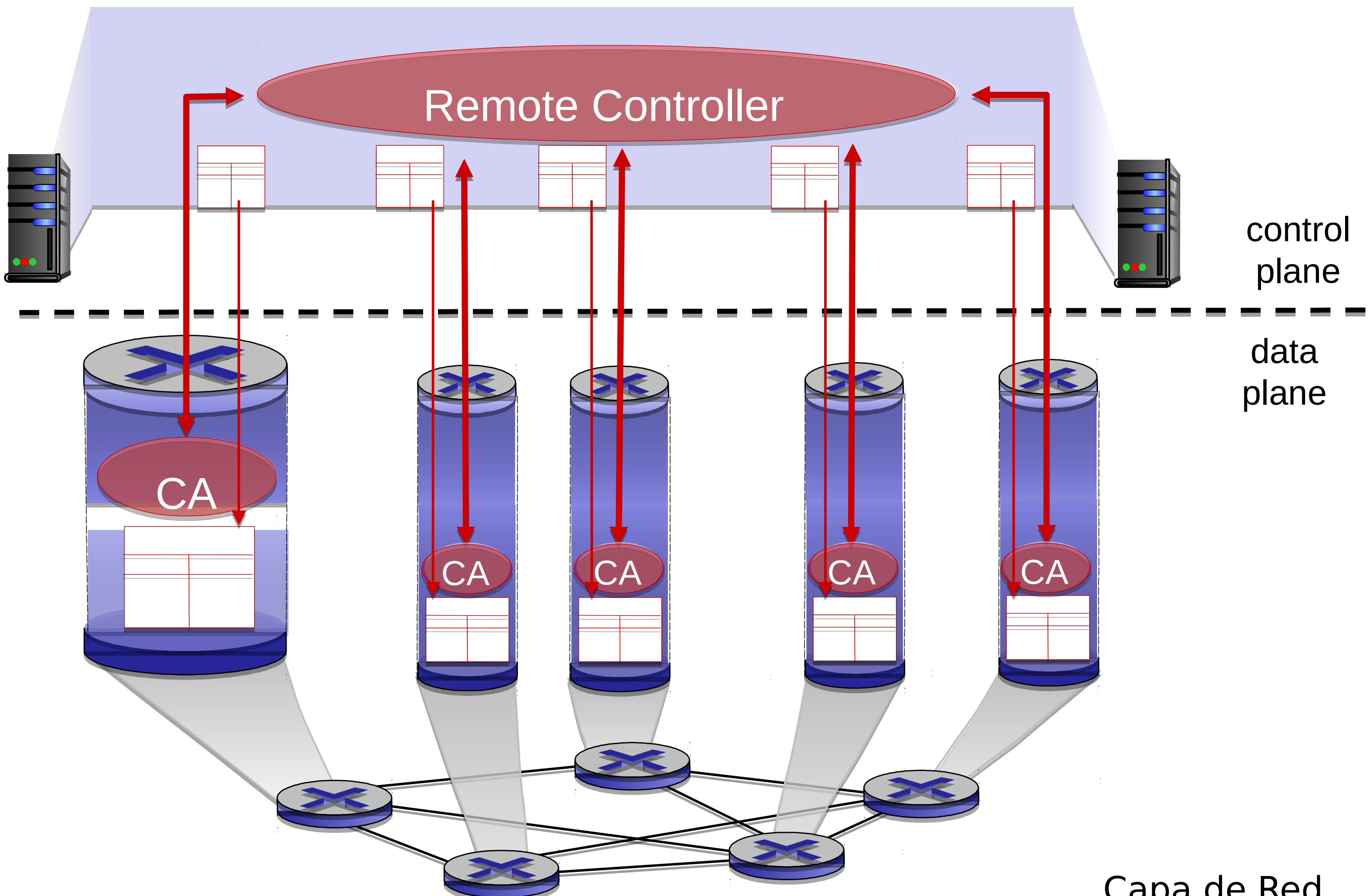
# Plano de control por router

Algoritmo de ruteo individual en cada router interactúa con los otros en el plano de control para determinar la tabla de re-envío



# Plano de control lógicamente centralizado

Un controlador distinto (típicamente remoto) interactúa con agentes de control (Cas) en routers para determinar las tablas de reenvío.



# Capítulo 5: Capa de Red: Plano de Control

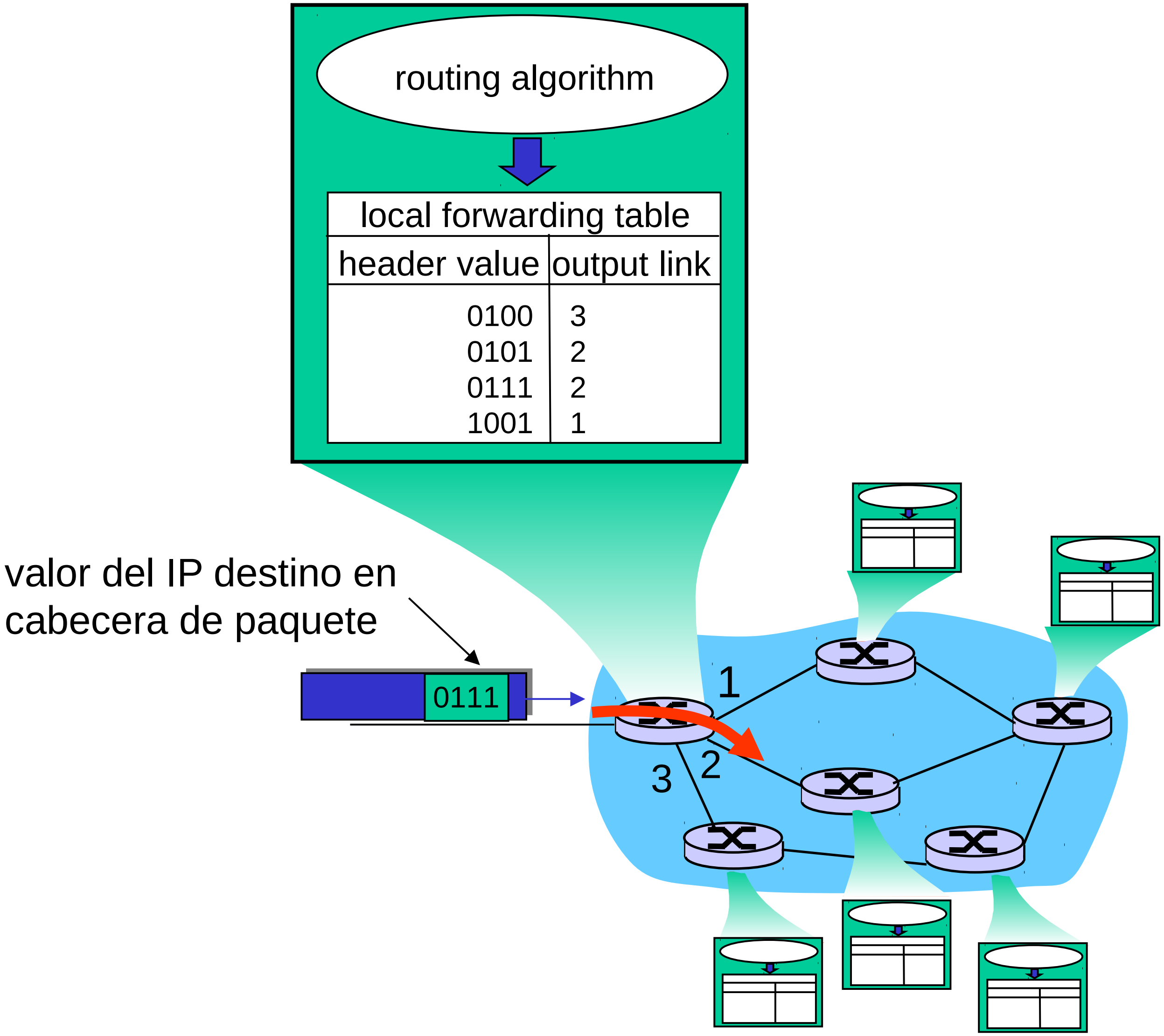
- 5.1 Introducción
- 5.2 Protocolos de ruteo
  - Estado de enlace
  - Vector de Distancia
- 5.3 Ruteo dentro de sistemas autónomos en la Internet: OSPF
- 5.4 Ruteo entre ISPs: BGP
- 5.5 Plano de control de SDN

Otras secciones del capítulo no son cubiertas en este curso

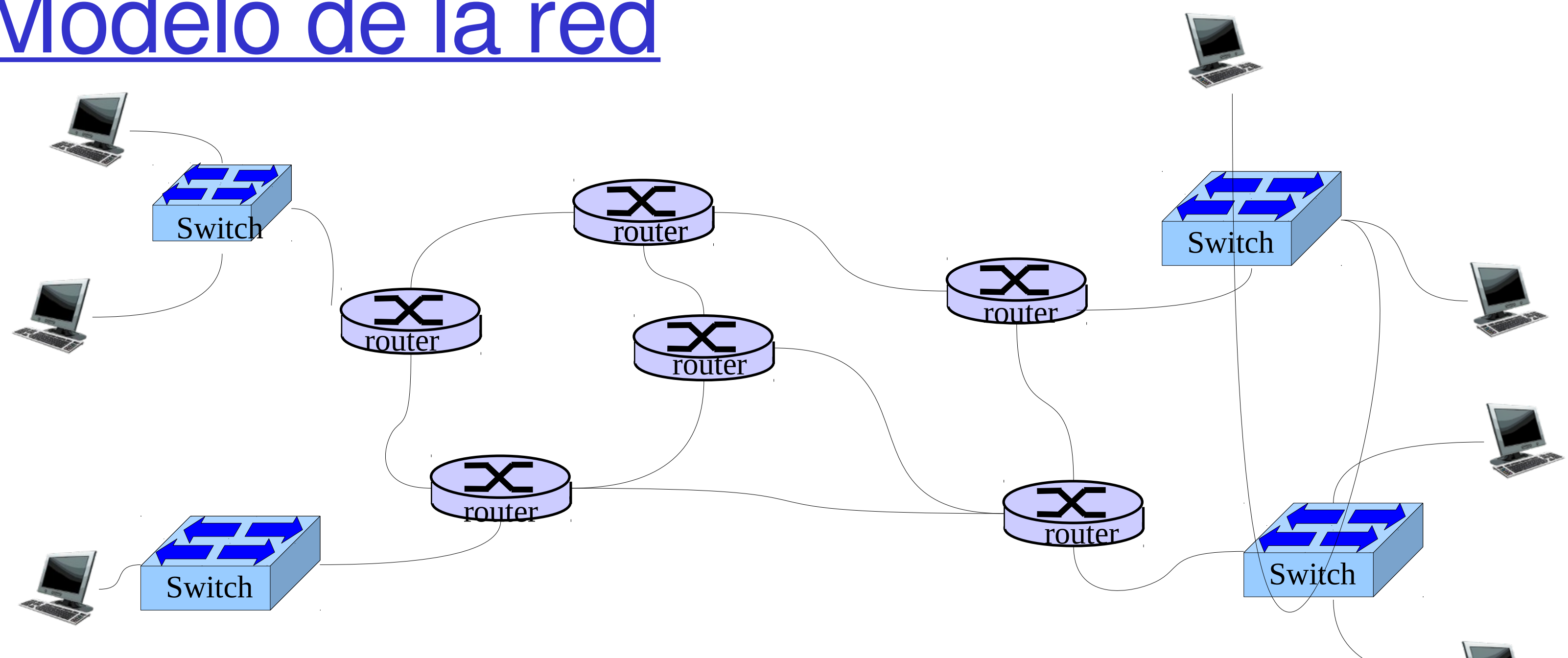
# Protocolos de Ruteo

- Objetivos de Protocolos de Ruteo: determinar "buenos" caminos (equivalentemente, rutas) desde host fuente al host destino a través de la red de routers
- Camino: secuencia de routers que serán recorrridos al ir desde un host fuente a un host de destino final dado.
- "good": "menor costo", "más rápido", "menor congestión"
- Ruteo: uno de los desafíos "top-10" de las redes

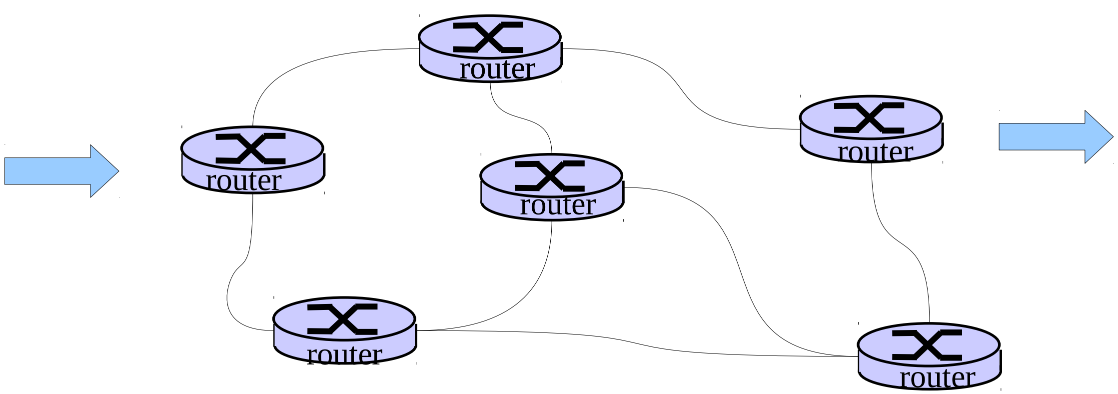
# Interacción de ruteo y re-envío

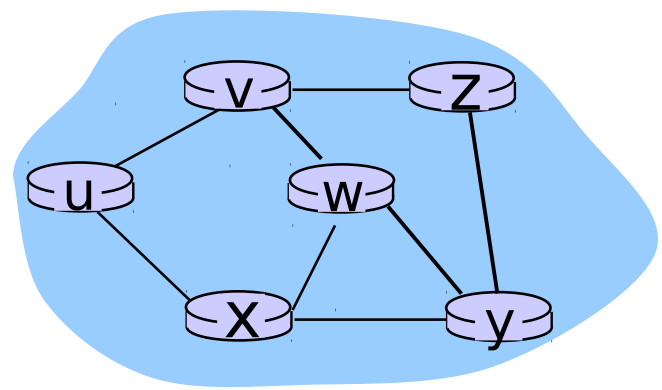


# Modelo de la red

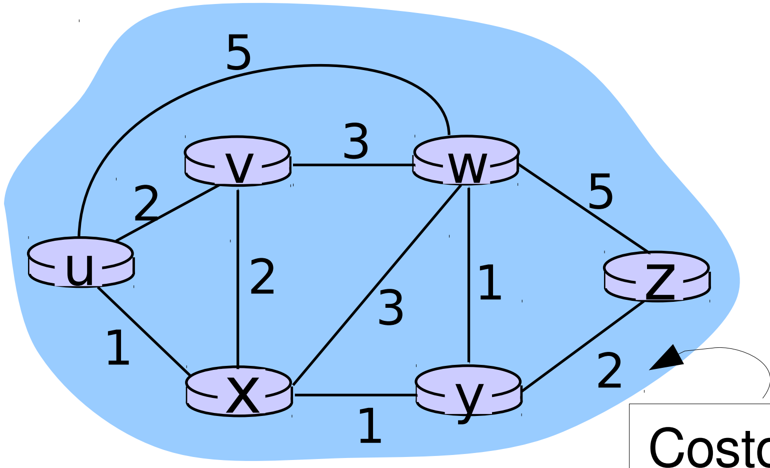



El problema de ruteo, en un routers, se traduce a identificar el router adyacente a quien enviar el paquete para que llegue a la subred destino. El computador de origen tiene sólo una opción => lo dejamos fuera del análisis.



# Abstracción de la red vía un Grafo



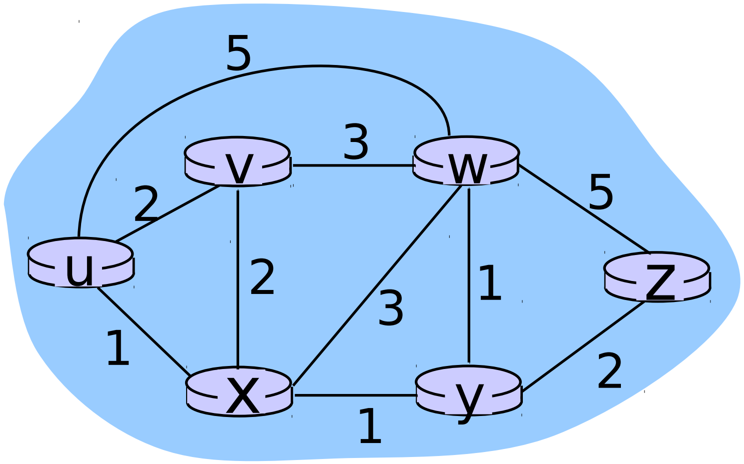

Costo del enlace: retardo, BW, congestión, $

Grafo: G = (N,E)

N = conjunto de routers = { 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)}

# Abstracción de Grafos : costos




- $c(x, y)$ = costo de enlace $(x, y)$

  - e.g., $c(w,z) = 5$

- costo puede ser, por ejemplo, 1, inversamente relacionado al ancho de banda, o directamente relacionado a la congestión

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

Pregunta clave: ¿Cuál es la ruta de mínimo costo entre u y z ?
Algoritmo de ruteo: algoritmo que encuentra ese costo mínimo

# Clasificación de los algoritmos de ruteo

Usa información global o descentralizada?

Global:

- Todos los routers conocen la topología completa y costos de enlaces
- Algoritmos de "estado de enlace" (link state). Creador Edsger W. Dijkstra (1956)

Descentralizada:

- El router conoce vecinos conectados físicamente y el costo del enlace a ellos.
- Proceso iterativo de cómputo e intercambio de información con sus vecinos
- Algoritmos de "vector de distancia" Creadores Bellman y Ford (1958 y 1956)

Es estático o dinámico?

Estático:

- Cuando rutas cambian poco en el tiempo

Dinámico:

- Cuando rutas cambian más rápidamente
    - Actualizaciones periódicas
    - En respuesta a cambios de costos de enlaces

# Capítulo 5: Capa de Red: Plano de Control

- 5.1 Introducción
- 5.2 Protocolos de ruteo
  - Estado de enlace
  - Vector de Distancia
- 5.3 Ruteo dentro de sistemas autónomos en la Internet: OSPF
- 5.4 Ruteo entre ISPs: BGP
- 5.5 Plano de control de SDN

Otras secciones del capítulo no son cubiertas en este curso

# Un Algoritmo de ruteo “estado de enlace”

Algoritmo de Dijkstra

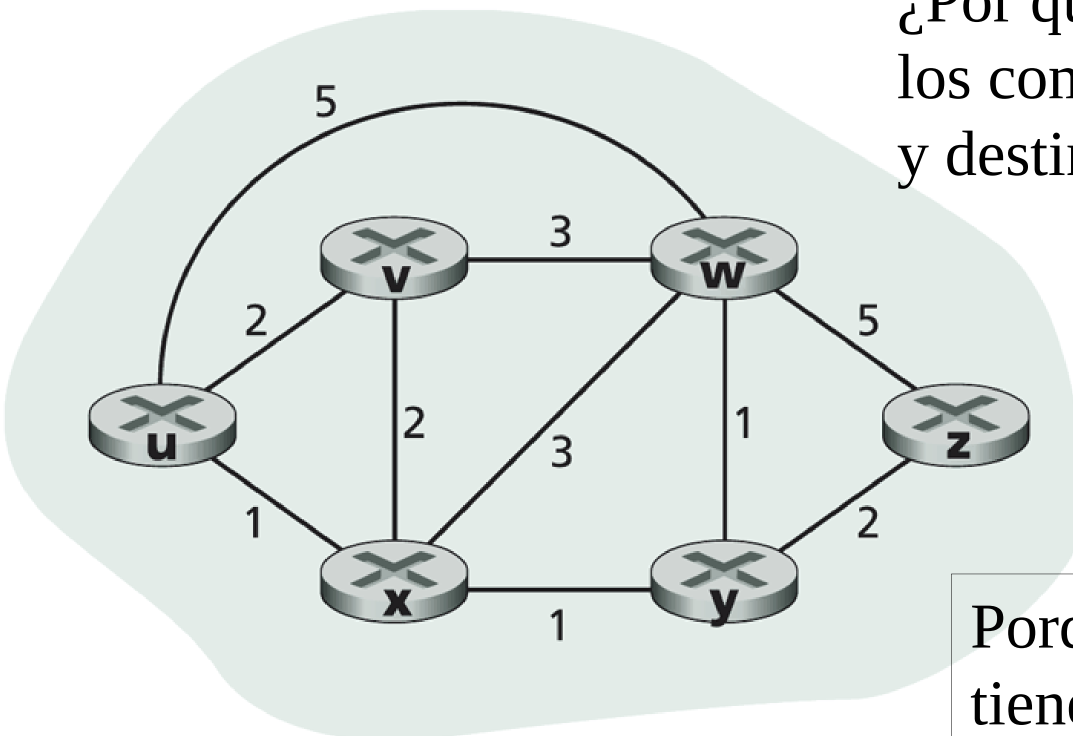
- Supone topología de red y costos de enlaces conocidos a todos los nodos
  - Esto se logra vía “difusión de estado de enlace”
  - Todos los nodos tienen la misma información
- Se calcula el camino de costo menor desde un nodo (fuente) a todos los otros
  - Determina tabla de re-envío para ese nodo
- Iterativo: después de k iteraciones, se conoce el camino de menor costo a k destinos (ver los valores de p(v) en el camino resultante)

## Notación:

- c(x,y): costo del enlace desde nodo x a y; $= \infty$ si no es vecino directo
- D(v): valor actual del costo del camino desde fuente a destino v.
- p(v): nodo predecesor a v en el camino de fuente a v.
- N': conjunto de nodos cuyo camino de costo mínimo ya se conoce

# Modelo abstracto para la red

¿Por qué no se incluyen los computadores fuente y destino?




Porque suponemos que tienen sólo una opción como próximo salto

**Figure 4.25** ♦ Abstract graph model of a computer network

# Algoritmo de Dijsktra

**Notación:**

- c(x,y): costo del enlace desde nodo x a y; = ∞ si no es vecino directo
- D(v): valor actual del costo del camino desde fuente a destino v.
- p(v): nodo predecesor a v en el camino de fuente a v.
- N': conjunto de nodos cuyo camino de costo mínimo (desde origen) ya se conoce

***Inicialización:***

N' = {u}

for todos los nodos v

if v es vecino de u

then D(v) = c(u,v)

else D(v) = ∞

***Loop***

find w not in N' tal que D(w) es un mínimo

agregue w a N'

actualiza D(v) para todo v adyacente a w que no está en N' usando:

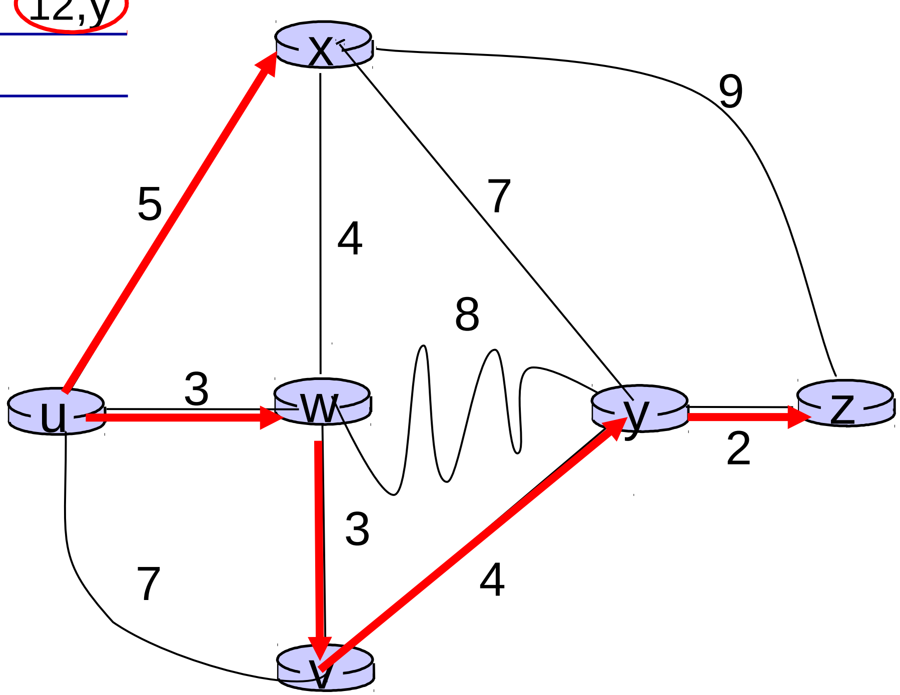
D(v) = min( D(v), D(w) + c(w,v) )

/* nuevo costo a v es el costo del camino actual a v o el costo del camino más corto conocido a w más el costo de w a v*/

***until todos los nodos están en 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 | | | | | |

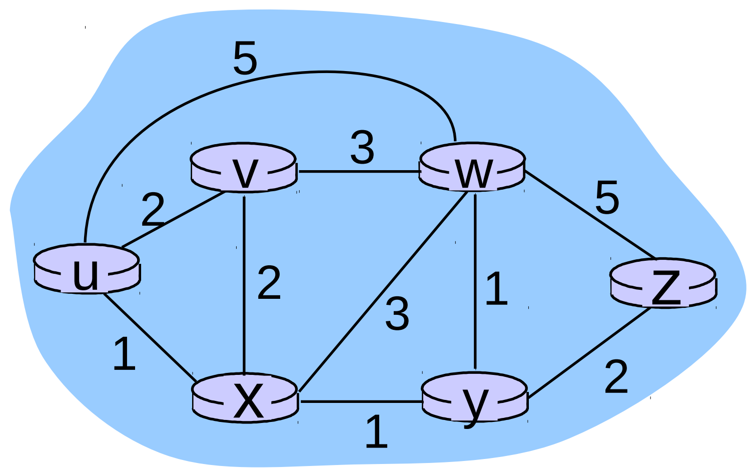



*Notar:*

- Construye el árbol de rutas más cortas recorriendo los nodos predecesores.
- Empates se resuelven arbitrariamente (el árbol no es único)

# 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 | | | | | |




* Hay más ejemplos en: http://gaia.cs.umass.edu/kurose_ross/interactive/

# Algoritmo de Dijkstra: ejemplo (2)

Resultado de árbol de caminos más cortos desde u:

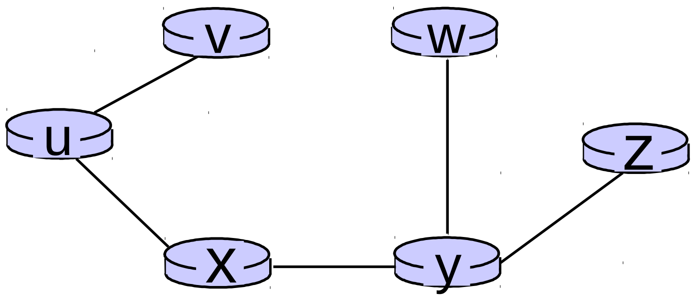



Tabla de re-envío resultante para u:

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

# Algoritmo de Dijkstra, discusión

Complejidad para n nodos

- Cada iteración: ve todos los nodos, w, no presentes en N
- $n(n+1)/2$ comparaciones: $O(n^2)$
- Otras implementaciones son posibles: $O(n \log n)$

Oscilaciones en cálculos son posibles:

- El algoritmo debe ser recalculado periódicamente para sobreponerse a enlaces caídos. Luego puede ocurrir que si costo enlace = cantidad de tráfico enviado por enlace, se tenga una oscilación entre enlaces de menor costo.

# Capítulo 5: Capa de Red: Plano de Control

- 5.1 Introducción
- 5.2 Protocolos de ruteo
  - Estado de enlace
  - Vector de Distancia
- 5.3 Ruteo dentro de sistemas autónomos en la Internet: OSPF
- 5.4 Ruteo entre ISPs: BGP
- 5.5 Plano de control de SDN

Otras secciones del capítulo no son cubiertas en este curso

# Algoritmo Vector de Distancia (1)

## Ecuación de Bellman-Ford

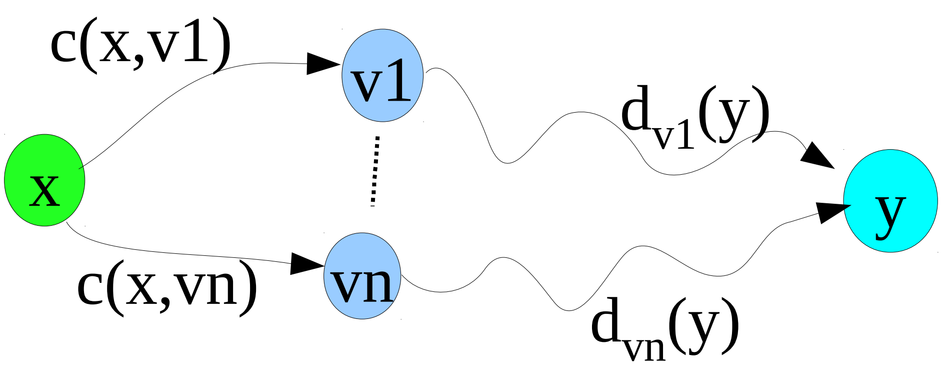
Define

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

Entonces:

$$\boxed{d_x(y) = \min_{v \text{ es vecino de } x} \{c(x,v) + d_v(y)\}}$$

Donde min es tomado sobre todos los vecinos v de x



# Algoritmo Vector de Distancia (2)

- $D_x(y)$ = costo mínimo estimado de x a y
- Vector de distancia: $\mathbf{D}_x = [D_x(y): y \in N]$
- Nodo x conoce el costo a cada vecino v: $c(x,v)$
- Nodo x mantiene $\mathbf{D}_x = [D_x(y): y \in N]$
- Nodo x también mantiene los vectores de distancia de sus vecinos
  - Para cada vecino v, x mantiene $\mathbf{D}_v = [D_v(y): y \in N]$

# Algoritmo Vector de distancia (3)

Idea básica:

- Cada nodo envía periódicamente su vector de distancia estimada a sus vecinos
- Cuando el nodo x recibe un nuevo vector de dist. estimado desde un vecino, éste actualiza su propio vector de dist. usando la ecuación de B-F:

$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ *para cada nodo y en N*

- Si el vector de dist. cambia entonces el nodo x envía su nuevo vector a sus vecinos, y ellos a su vez pueden actualizar sus vectores de distancia
- Bajo condiciones normales, el valor estimado de $D_x(y)$ *converge al menor costo real* $d_x(y)$

# Algoritmo Vector de Distancia (4)

**Iterativo y asincrónico:** cada iteración local es causada por:

- Cambio en costo de enlace local
- Actualización de vector por mensaje de vecino

**Distribuido:**

- Cada nodo notifica a sus vecinos *sólo* cuando su vector cambia
  - Vecinos entonces notifican a sus vecinos si es necesario

**Cada nodo:**

*wait* for (cambio en costo de enlace local o llegada de mensaje desde vecino)

↓

*recalcule* DV estimado

↓

if (DV a cualquier destino ha cambiado)

*notificar* a vecinos

# Ejemplo: Vector de distancia

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

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

**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 |

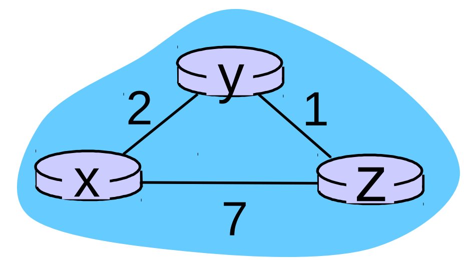
**node y table**

| 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 |

time



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

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

# Ejemplo: Vector de distancia

**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 |

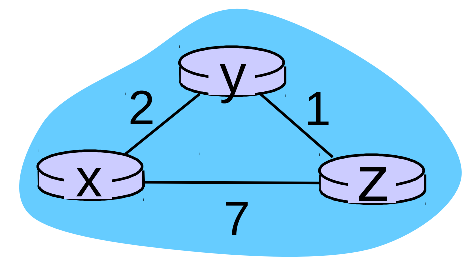
**node z table**

| 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 |

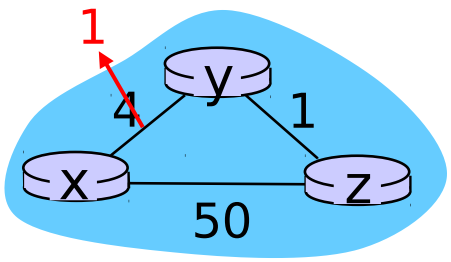
| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

time



# Vector de distancia: cambios en costos de enlaces

- **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



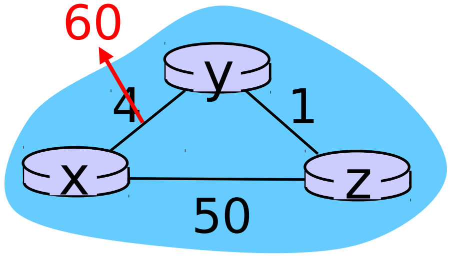

En el tiempo $t_0$, ***y*** detecta un cambio en costo de enlace, actualiza su DV e informa a sus vecinos.

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***.

# Vector de distancia: cambio en costo de enlaces

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

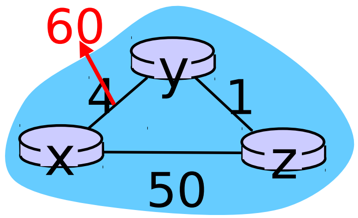


# Vector de distancia: cambio en costo de enlaces (e.g. incremento de costo) $\rightarrow$ Problema!!

- Inicialmente: $D_y(x) = 4$, $D_y(z) = 1$, $D_z(x) = 5$, $D_z(y) = 1$

**node y table**

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

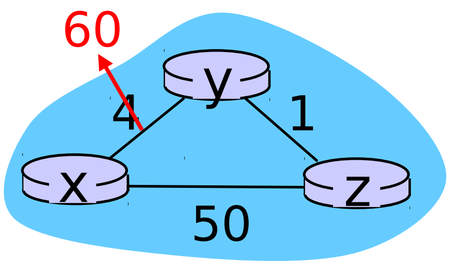



**node y table**

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 6 | 0 | 1 |
| z | 5 | 1 | 0 |

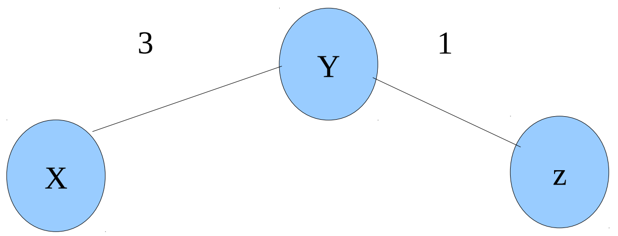
# Vector de distancia: cambio en costo de enlaces

- ¿Qué pasa si el enlace se cae? Su costo es $\infty$. La solución es conocida como "Reversa envenenada":



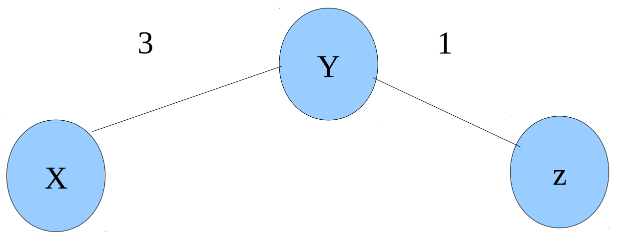

- Si Z routea a través de Y para llegar a X:
  - Z informa a Y que su distancia a X es infinita (para que Y no rutee a X vía Z); es decir, cuando Z informa a Y, Z pretende tener distancia infinita a todos los destinos alcanzables vía Y.
- ¿Resuelve completamente el problema de contar hasta el infinito? No, ¿por qué?

## Si no tenemos “reversa envenenada” explique qué ocurre en la siguiente red (sólo tres routers) cuando el enlace x-y se corta:




- Se produciría un aumento paulatino de la distancia para llegar de Y y Z a X hasta llegar al valor máximo para la distancia.
- Inicialmente, Y llega a X con costo 3 y Z llega a X con costo 4. Cuando el enlace se corta, Y cree tener una ruta a X de distancia 5 vía Z. Luego Z cambia su distancia a X a 6. Esto se repite hasta llegar al valor máximo para la distancia.

# Para este caso, ¿qué ocurre si ocupamos “reversa envenenada” y se corta el enlace x-y?




- El algoritmo converge rápidamente.
- Inicialmente Y llega a X con distancia 3 y Z llega a X con distancia 4; pero como Z lo hace vía Y, Z informó a Y que su distancia a X es “infinita”. Así cuando se cae el enlace, Y no encuentra enlace alternativo a X y actualiza su distancia a X a “infinito” e informa a Z, ante lo cual Z también la actualiza a “infinito”.

# Mencione una desventaja y una ventaja del algoritmo de ruteo “Estado de Enlace” versus el de “Vector de Distancia”.

- 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.

Supongamos que a usted le piden hacer un programa computacional (en el lenguaje que usted maneje) para encontrar la ruta más corta entre dos ciudades. Si la entrada para el programa es una tabla con todos los caminos entre ciudades adyacentes señalado ciudad origen, destino y distancia entre ellas, ¿usaría alguna versión del algoritmo "Estado de Enlace" o "Vector de Distancia"?. Explique.

- Elijo estado de enlace, debido a que el cómputo se debe hacer centralizadamente y en el archivo se cuenta justamente con la información de los nodos y enlaces del grafo donde aplicar el algoritmo de Dijkstra.

# ¿Cuántas sub-redes hay aquí?

Este router sería el único en grafo para obtener tabla de re-envío