Tecnología Digital IV: Redes de Computadoras

Clase 16: Protocolos de Ruteo - Parte 1

Lucio Santi & Emmanuel Iarussi

Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

20 de mayo de 2025

Agenda

- Protocolos de ruteo
  - Introducción
  - Ruteo *link state*
  - Ruteo *distance vector*
- Ruteo intra-ISP: **OSPF**
- Ruteo inter-ISP: **BGP**

Objetivos

- Entender los conceptos detrás del **plano de control** en Internet
  - Algoritmos de ruteo tradicionales
- Entender cómo se instancian e implementan en Internet
  - Protocolos OSPF y BGP

Repaso: nivel de red

Funciones del nivel de red

- *Forwarding*: trasladar paquetes desde los enlaces de entrada de un router hacia el enlace de salida apropiado

plano de datos

- *Routing*: determinar la ruta que siguen los paquetes desde el origen hasta el destino

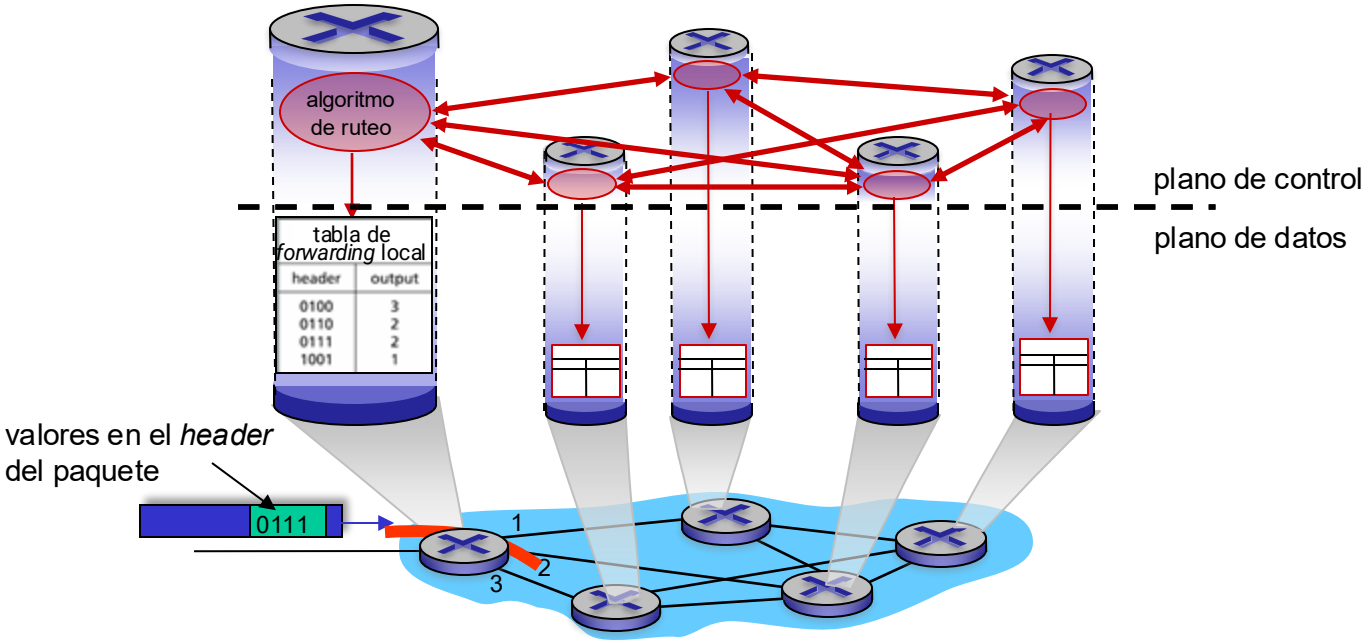
plano de control

Dos enfoques para estructurar el plano de control:

- Control en cada router (tradicional)
- Control centralizado (*Software Defined Networking*)

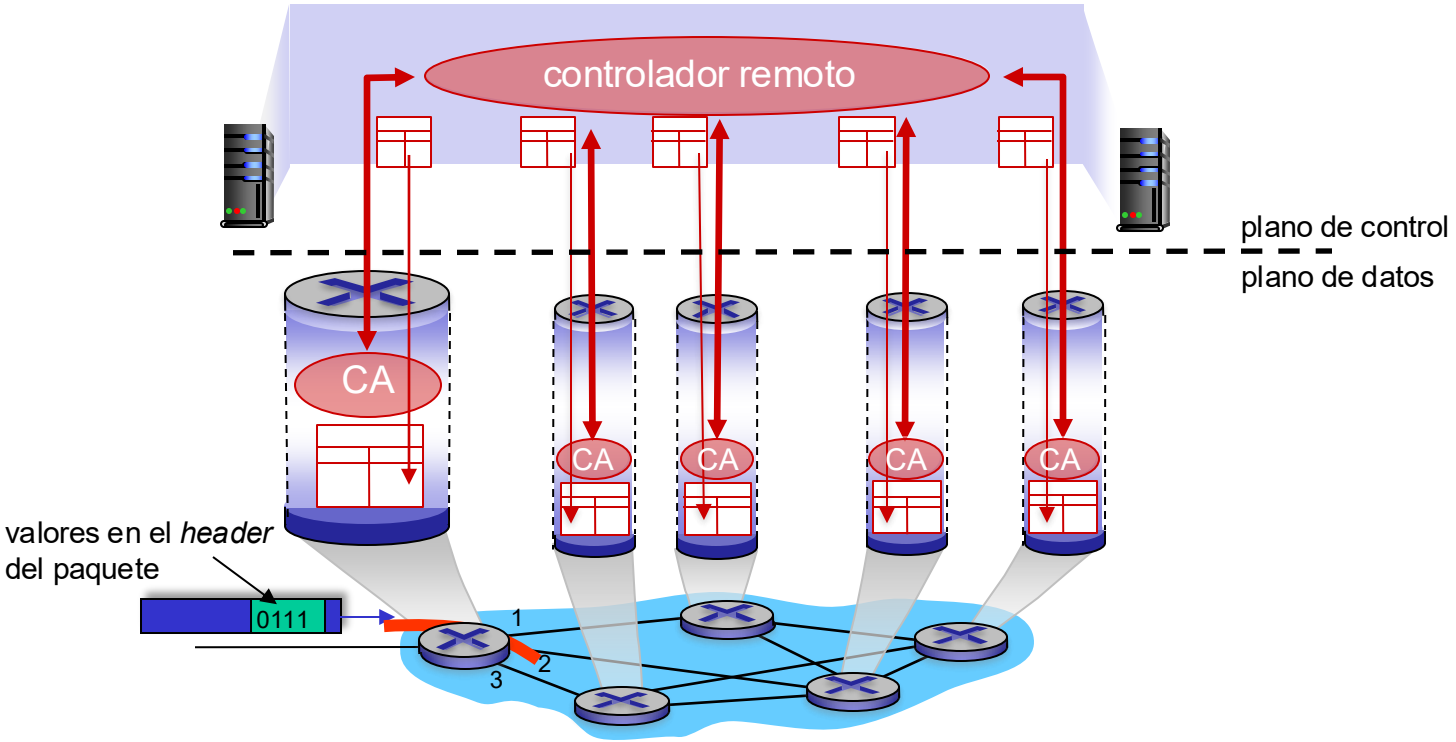
Plano de control en cada router

Algoritmos de ruteo distribuidos, implementados en todos los routers e interactuando entre sí



Plano de control vía SDN

Un controlador remoto computa e instala las tablas de *forwarding* en los routers

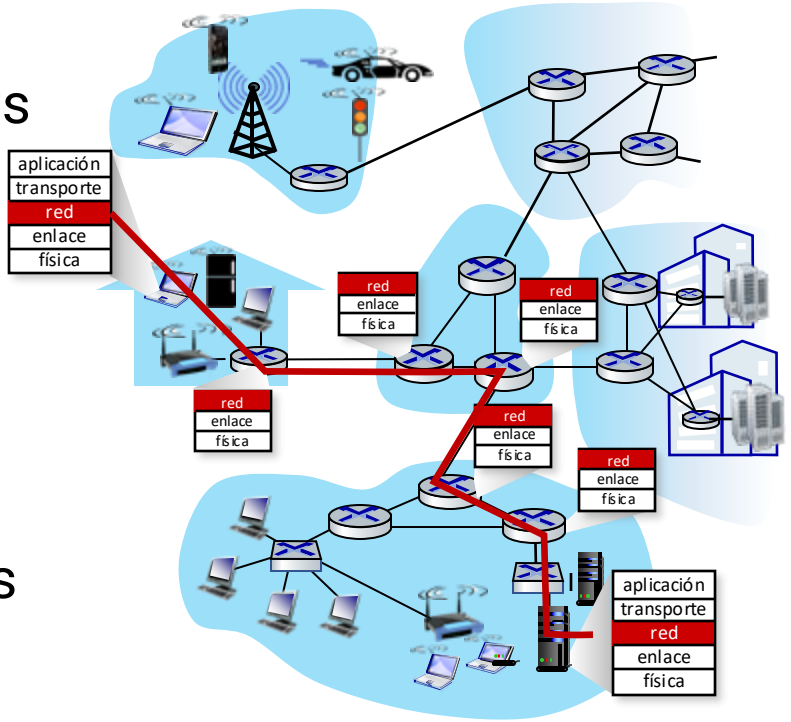


Protocolos de ruteo

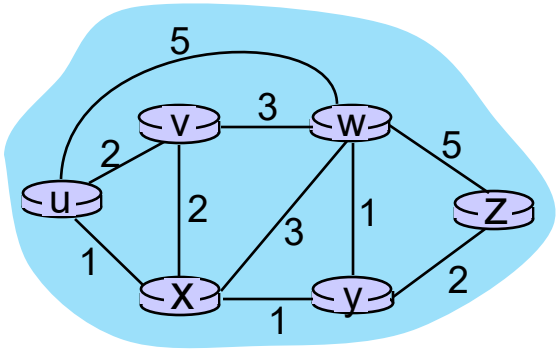
Protocolos de ruteo: introducción

Objetivo: determinar caminos “buenos” entre hosts a través de una red de routers

- *camino*: secuencia de routers que atraviesan los paquetes desde un host origen hasta un host destino
- *“bueno”*: el de menor “costo”, el “más rápido”, el “menos congestionado”...
- El ruteo es uno de los problemas más importantes en redes de computadoras



Abstracción: grafo con costos en los enlaces



$c_{a,b}$: *costo* del enlace que conecta ***a*** con ***b***

e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

el costo lo define el operador de la red: puede ser constante, inversamente proporcional al ancho de banda, directamente proporcional a la congestión, etc.

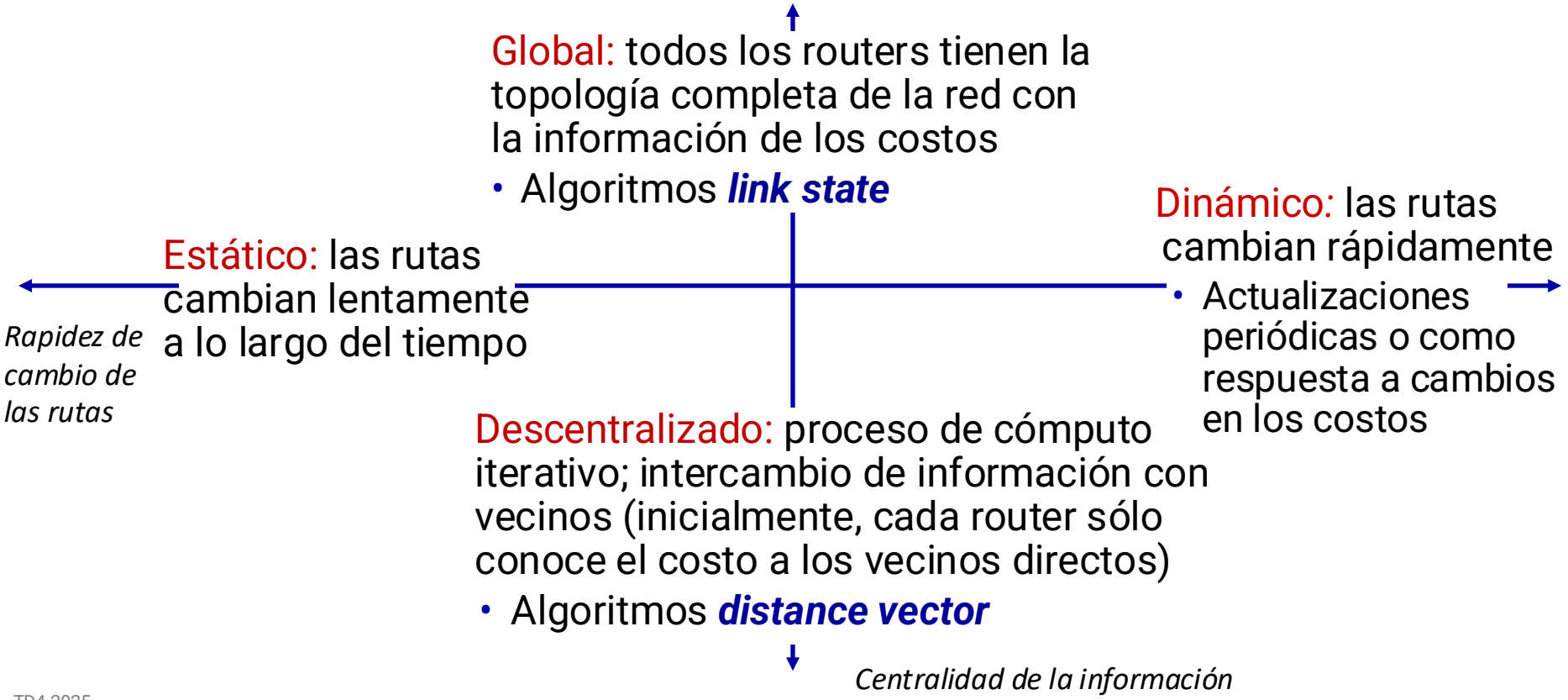
Grafo *G = (N,E)*

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)* }

Taxonomía de algoritmos de ruteo



Ruteo *link state*

Algoritmo de ruteo *link state*: Dijkstra

- Centralizado: todos los nodos conocen la topología de la red y los costos de los enlaces
  - Se logra vía *broadcasting* de estado de enlaces
  - Todos los nodos tienen la misma información
- Computa caminos de costo mínimo de un nodo origen hacia los nodos restantes
  - Genera la tabla de *forwarding* para dicho nodo
- Iterativo: luego de *k* iteraciones, se conoce el camino de costo mínimo hacia *k* destinos

notación

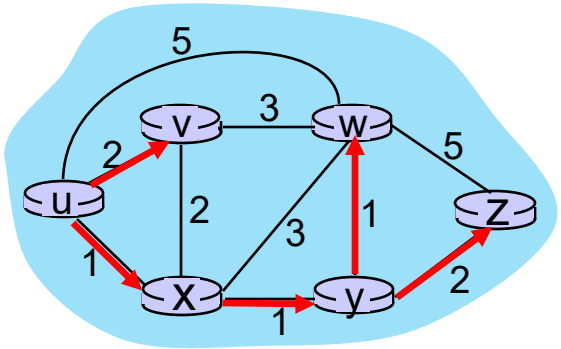
- $c_{x,y}$: costo del enlace de *x* a *y* (∞ si no son vecinos)
- *D(v)*: estimación actual del costo del camino mínimo desde el origen al destino *v*
- *p(v)*: nodo predecesor a lo largo del camino del origen a *v*
- *N'*: conjunto de nodos para los que ya se conoce el camino mínimo

Algoritmo de ruteo *link state*: Dijkstra

1 *Inicialización*
2 $N' = \{u\}$ `# calcular camino mínimo desde u hacia los otros nodos`
3 Para cada nodo v
4 Si v es vecino de u `# u sólo conoce el costo a sus vecinos directos al principio`
5 $D(v) = c_{u,v}$ `# puede no ser el camino de costo mínimo`
6 Si no, $D(v) = \infty$
7
8 *Repetir*
9 Encontrar w (que no esté en N') tal que $D(w)$ es mínimo
10 Agregar w a N'
11 Actualizar $D(v)$ para todo v vecino de w que no esté en N':
12 **$D(v) = \text{mín}\,(D(v),\ D(w) + c_{w,v})$**
13 `# el nuevo camino mínimo hacia v es, o bien el camino mínimo`
14 `anterior, o bien el camino mínimo conocido a w más el costo de ir de w a v`
15 *hasta que todos los nodos estén en N'*

Algoritmo de Dijkstra: 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					



Inicialización (paso 0): para todo *a*, si *a* es vecino, $D(a) = c_{u,a}$

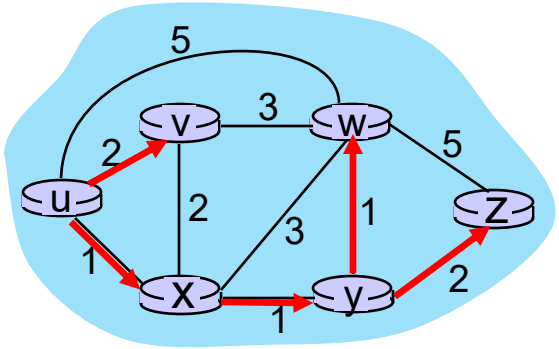
Encontrar *a* que no esté en N' tal que $D(a)$ es mínimo

Agregar *a* a N'

Actualizar $D(b)$ para todo *b* vecino de *a* no que no esté en N' :

$$D(b) = \text{mín}\ (\ D(b),\ D(a) + c_{a,b}\ )$$

Algoritmo de Dijkstra: ejemplo



Árbol de costo mínimo para *u*

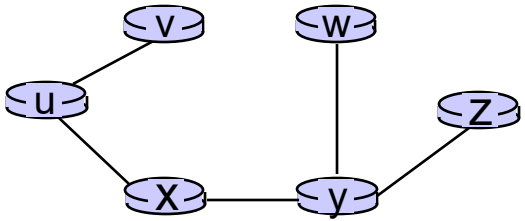


Tabla de *forwarding* para *u*

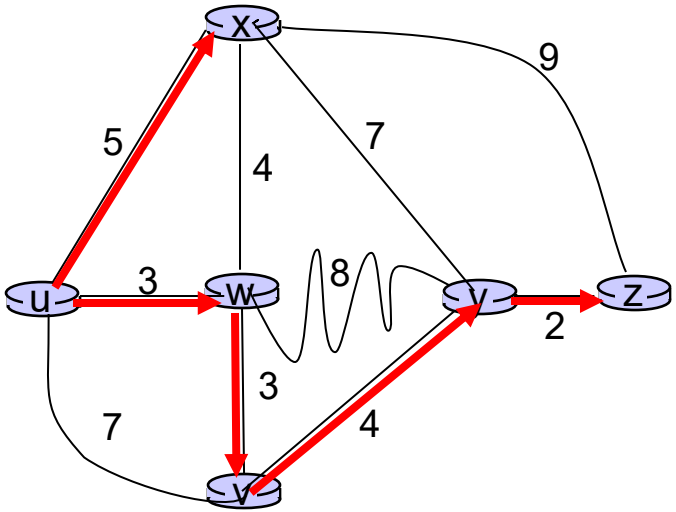
destino	enlace
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

ruta directa de *u* a *v*

ruta vía *x* hacia todos los otros destinos

Algoritmo de Dijkstra: otro ejemplo

Paso	N'	v D(v), p(v)	w D(w), p(w)	x D(x), p(x)	y D(y), p(y)	z 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	uwx v				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



Notas

- El árbol de costo mínimo se construye siguiendo los nodos predecesores
- Puede haber empates (con mecanismos de desempate arbitrarios)

Algoritmo de Dijkstra: análisis

Complejidad computacional: *n* nodos, *e* aristas

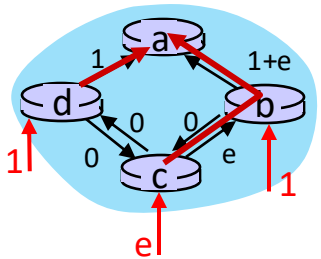
- Hay *n* iteraciones; en cada una se revisan los nodos *w* que no estén en *N'*
- $n(n+1)/2$ comparaciones: complejidad cuadrática, $\mathbf{O(n^2)}$
- Se puede implementar en tiempo $\mathbf{O(e + n \log n)}$ (con *Fibonacci heaps*)

Complejidad en mensajes

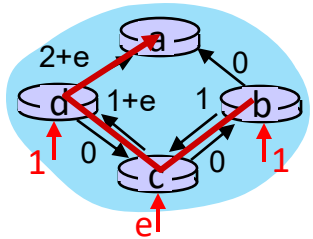
- Cada router debe hacer *broadcast* de su estado a los otros routers
- Algoritmos de *broadcasting* eficientes permiten diseminar un mensaje atravesando $O(n)$ enlaces
- Agregando para los *n* routers, tenemos $\mathbf{O(n^2)}$ mensajes en total

Algoritmo de Dijkstra: oscilaciones

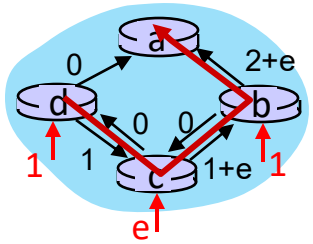
- Si los costos dependen del volumen de tráfico, pueden ocurrir **oscilaciones**
- Ejemplo:
  - Ruteando hacia *a*, tenemos tráfico entrante a *d*, *c*, *b* con tasas 1, *e* (<1), 1
  - Los costos de los enlaces son direccionales y dependientes del tráfico



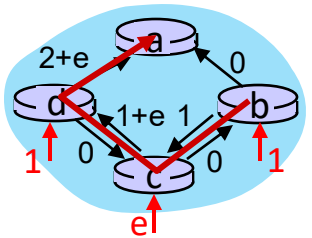
inicialmente



dados dichos costos, encontrar nuevas rutas (redundando en nuevos costos)



dados dichos costos, encontrar nuevas rutas (redundando en nuevos costos)



dados dichos costos, encontrar nuevas rutas (redundando en nuevos costos)...