



Módulo 04

La Capa de Red

(Pt. 3)



Redes de Computadoras
Depto. Cs. e Ing. de la Comp.
Universidad Nacional del Sur



Copyright

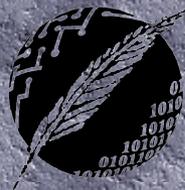
- Copyright © **2010-2024** A. G. Stankevicius
- Se asegura la libertad para copiar, distribuir y modificar este documento de acuerdo a los términos de la **GNU Free Documentation License**, versión 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera
- Una copia de esta licencia está siempre disponible en la página <http://www.gnu.org/copyleft/fdl.html>
- La versión transparente de este documento puede ser obtenida de la siguiente dirección:

<http://cs.uns.edu.ar/~ags/teaching>



Contenidos

- Modelos de servicios de la capa de red
- Estructura interna de un router
- El protocolo **IP**
- **IPv4 vs. IPv6**
- Protocolos de ruteo
- Ruteo jerárquico
- Ruteo en internet
- Multicast



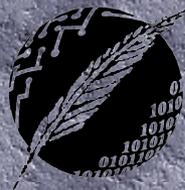
Protocolo ICMP

- El protocolo **ICMP** (Internet Control Message Protocol) es usado por las computadoras y los routers para intercambiar información de la capa de red
 - Se utiliza para reportar situaciones de error tales como señalar que una computadora o una red no puede ser alcanzada
 - Implementa la funcionalidad del programa **ping** (mediante los mensajes **echo request** y **echo reply**)
 - Los mensajes del protocolo **ICMP** se encapsulan dentro de datagramas **IP** convencionales



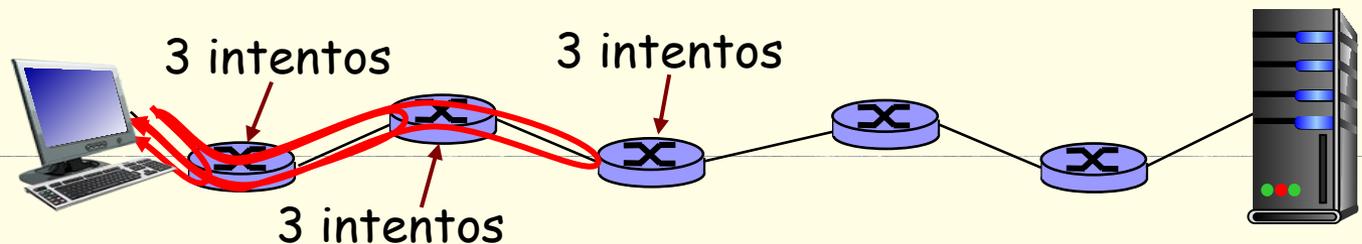
Protocolo ICMP

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 (control de cong.)
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

- El protocolo **ICMP** juega un rol crucial en la implementación del programa **traceroute**:
 - ➔ Se envían sucesivos segmentos **UDP** al destino, el primero con **TTL** de 1, en segundo de 2, etc.
 - ➔ Cuando el router descarta segmento **UDP** le avisa al origen mediante un mensaje **ICMP** (casualmente **tipo 11, código 0**)
 - ➔ Ese mensaje contiene el nombre y la dirección **IP** del router, justo el dato que se intentaba descubrir



Traceroute e ICMP

- Continúa:

- ➔ Cuando el mensaje **ICMP** retorna al origen, se toma nota del tiempo transcurrido (es decir, el **RTT**)
- ➔ Cuando el **TTL** sea suficiente como para atravesar la totalidad de la red, el destino contestará con un mensaje **ICMP** indicando que el puerto deseado no está ligado a proceso alguno (**tipo 3, código 3**)
- ➔ En este punto, la traza finaliza

- Existen variantes que usan otros protocolos, tales como **TCP** o directamente **ICMP**



Protocolo IPv6

- La motivación inicial para desarrollar una nueva versión del protocolo **IP** fue **el temor de que se agotaran las direcciones IPv4**
- Una vez tomada la decisión de crear una nueva versión, se aprovechó la oportunidad para corregir otros aspectos:
 - ➔ Adoptar un formato de encabezado que ayude a optimizar el proceso de forwarding
 - ➔ Agregar campos al encabezado para facilitar el **QoS**
 - ➔ Se crea un nuevo tipo de dirección: el anycast

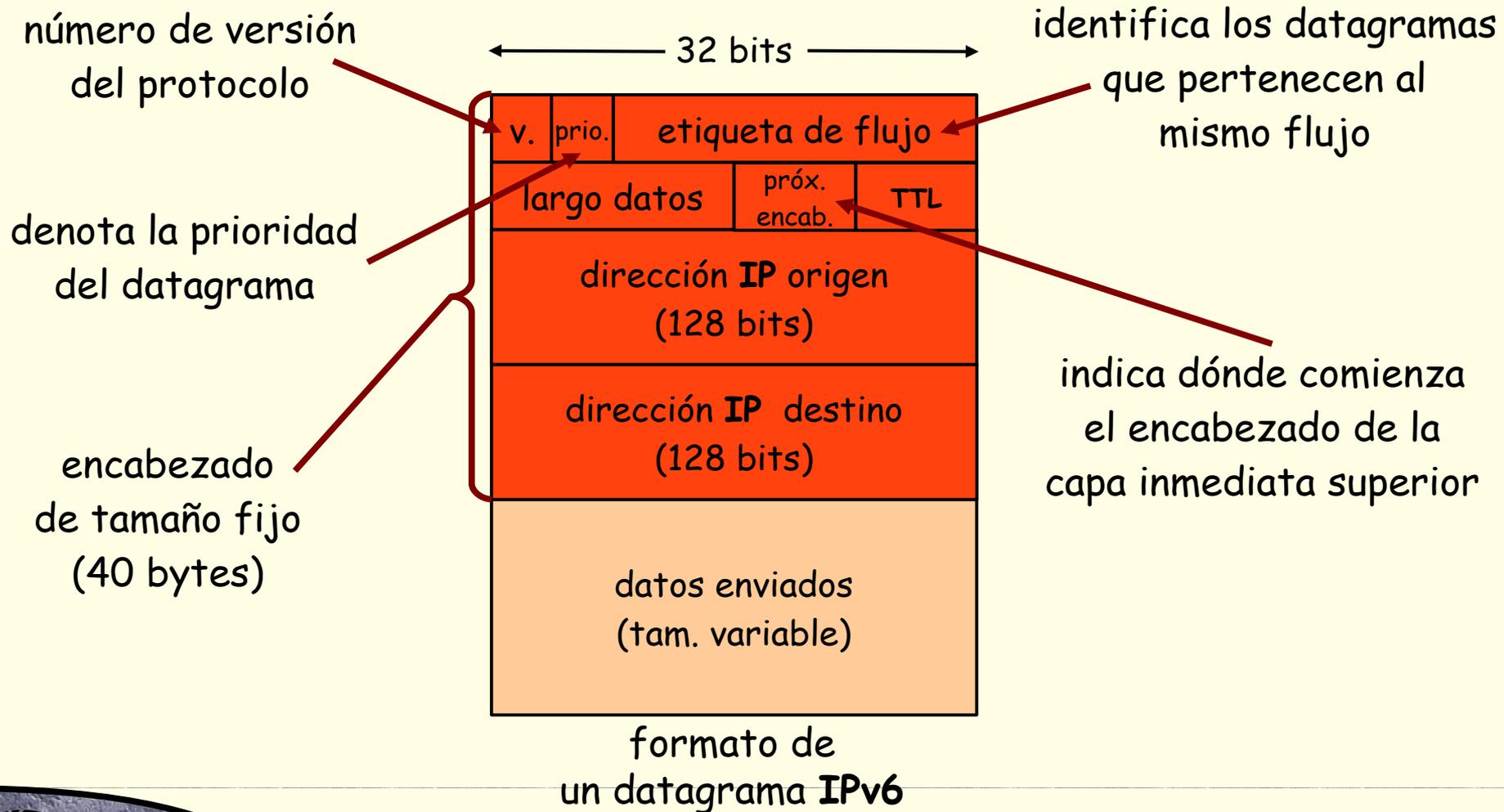


Protocolo IPv6

- La **versión 6** introduce un conjunto de cambios a nivel de los datagramas:
 - ➔ Se extiende la cantidad de bits en las direcciones **IP** de **32** a **128**
 - ➔ Se adopta un tamaño fijo de **40** bytes para el encabezado
 - ➔ Se deja de lado la posibilidad del fragmentado de datagramas
 - ➔ Se le da un lugar en el encabezado al concepto de “flujo”, si bien no se explicita del todo en qué consiste



Formato de los datagramas



IPv6 vs. IPv4

● Otros cambios introducidos en **IPv6**:

- Se eliminó el checksum del encabezado con el objeto de disminuir el tiempo de procesamiento en los routers visitados por los datagramas
- La parte opcional del encabezado no forma más parte del encabezado, pero se logra la misma funcionalidad usando el campo “próximo encabezado”
- Se actualizó el protocolo **ICMP** para adecuarse a todos estos cambios
- **ICMP** también incorpora funcionalidades para administrar grupos multicast

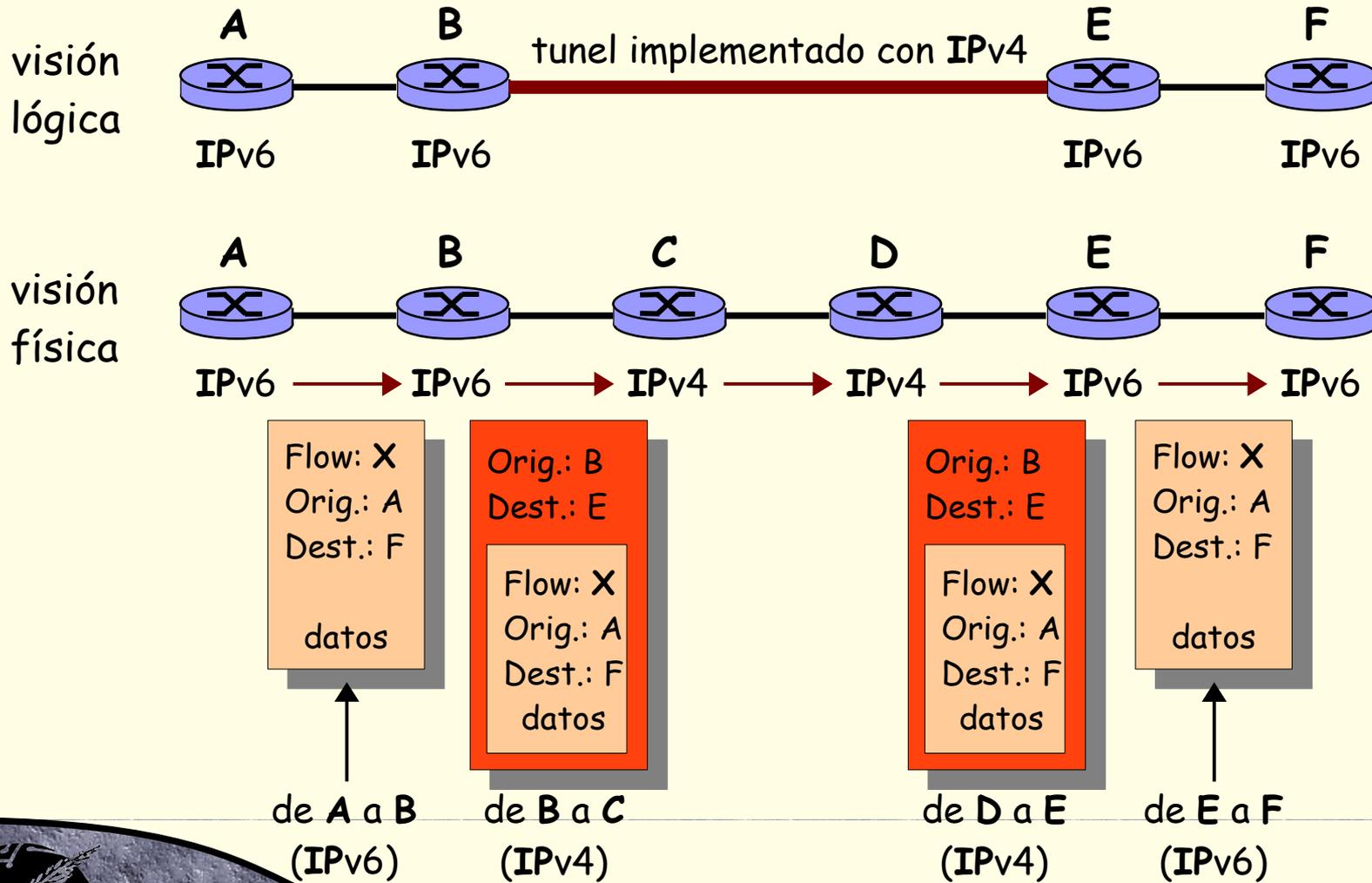


Migración IPv4 a IPv6

- Considerando el nivel actual de adopción de **IPv4** es imposible pensar en actualizar todos los routers al mismo tiempo
 - ➔ Se debe anticipar **un período de coexistencia** entre los protocolos **IPv4** e **IPv6**
- La técnica de **tunneling** es la herramienta ideal para hacer más llevadera esta coexistencia
 - ➔ La idea es encapsular los datagramas **IPv6** en la carga útil de datagramas **IPv4** cuando se deba atravesar porciones aun no compatibles del núcleo



Tunneling



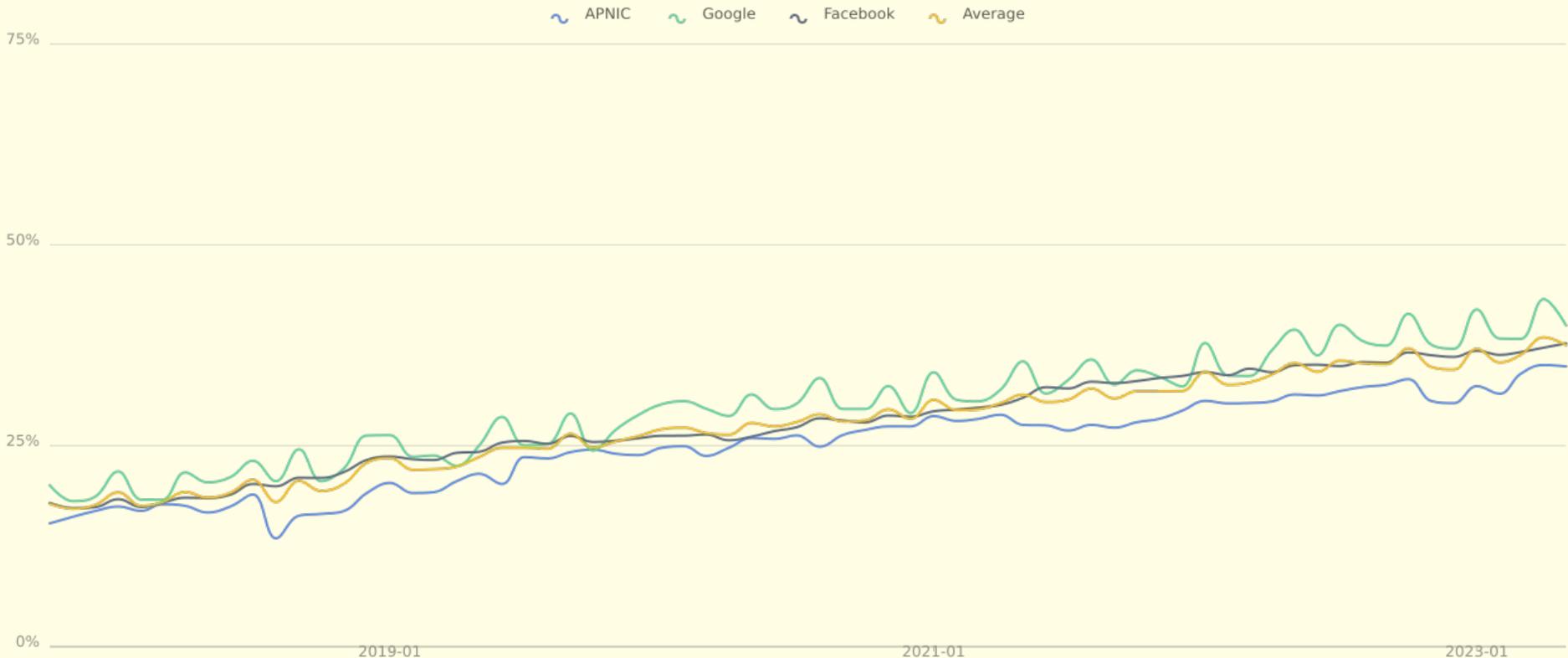
Grado de adopción

- Una encuesta reciente (año 2013) calculó que el nuevo estándar corre en apenas:
 - ➔ ~3% de los routers privados
 - ➔ ~11% de los routers gubernamentales
- Llevamos una eternidad en este proceso de transición...
 - ➔ Más de 25 años y contando (¡IPv6 surgió en 1998!)
 - ➔ ¿Cuántos avances hubo en estos más de 25 años?
 - ➔ ¿A qué se debe la demora? ¿Cómo se justifica?



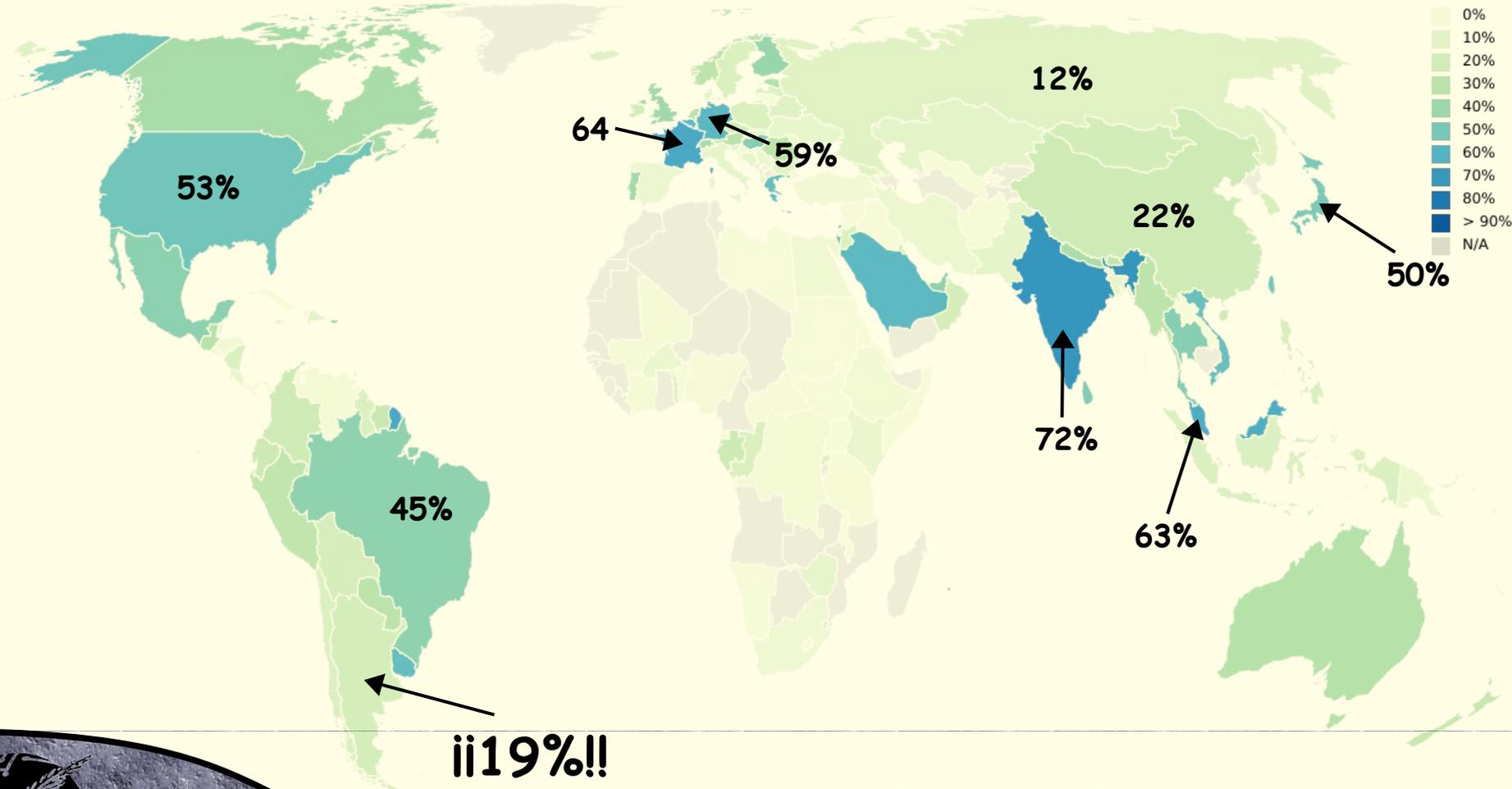
Grado de adopción

● No obstante...



Grado de adopción

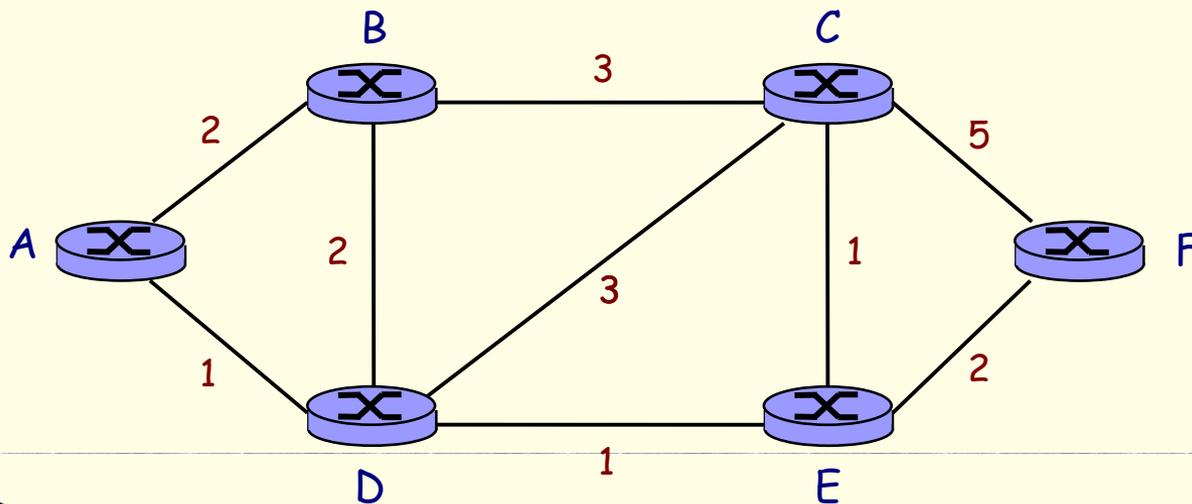
● Más aun...



<https://pulse.internetsociety.org/technologies>

Algoritmo de ruteo

- El **algoritmo de ruteo** tiene por objeto encontrar un buen camino a través de la red desde un cierto origen a un determinado destino
- En este contexto denominaremos **camino** a una secuencia de routers



Algoritmo de ruteo

- Generalmente se adopta un **grafo** como abstracción de la red en estos protocolos
 - Los **nodos** representan a los routers
 - Los **arcos** representan a los enlaces
- La noción de “buen camino” puede tener distintas interpretaciones:
 - Menor retardo
 - Más económico
 - Menos congestionado



Características deseables

- Un buen algoritmo de ruteo debe ser:
 - Sin lugar a duda **correcto**
 - **Simple**, de ser posible
 - **Robusto** con respecto a las fallas y los cambios en las condiciones de la red
 - **Estable** en lo que sus decisiones de ruteo respecta
 - **Equitativo** al asignar recursos
 - **Óptimo**
 - **Eficiente**



Global vs. descentralizado

● Algoritmo de ruteo **global**:

- Todos los routers tienen información completa acerca de los otros routers, los enlaces que los unen, el costo de cada uno de los mismos, etc.
- Se denominan algoritmos **estado de los enlaces**

● Algoritmo de ruteo **descentralizado**:

- Los routers sólo conocen a sus vecinos inmediatos y a los enlaces que los unen
- El algoritmo iterativamente intercambia información con sus vecinos para formar el **vector de distancias**



Estático vs. dinámico

- Algoritmo de ruteo **estático**:

- ➔ Las rutas encontradas no cambian o bien cambian muy poco a lo largo del tiempo

- Algoritmo de ruteo **dinámico**:

- ➔ Las rutas encontradas cambian con relativa frecuencia a lo largo del tiempo
- ➔ Requieren de actualizaciones periódicas en respuesta a las modificaciones en el estado de los enlaces



Algoritmo de Dijkstra

- El algoritmo de Dijkstra puede ser utilizado como algoritmo de ruteo tipo link state
 - Sólo puede aplicarse en los casos en que el costo asociado a los arcos sea no negativo
 - Los routers deben conocer la topología de la red y tener acceso a los costos de todos los enlaces
 - Este algoritmo produce como resultado un árbol cubriente minimal para un determinado router
 - A partir del árbol cubriente minimal se puede obtener la tabla de forwarding de ese router



Algoritmo de Dijkstra

- En primer lugar, un poco de notación:
 - $c(i, j)$: costo del arco del nodo i a j (se asume costo infinito para los pares de nodos no adyacentes)
 - $d(v)$: valor actual del mejor camino desde el nodo origen al nodo v
 - $p(v)$: nodo predecesor en el camino del origen al nodo v
 - N : conjunto de nodos para los cuales ya se conoce el camino de menor costo
 - Q : conjunto de nodos del grafo



Algoritmo de Dijkstra

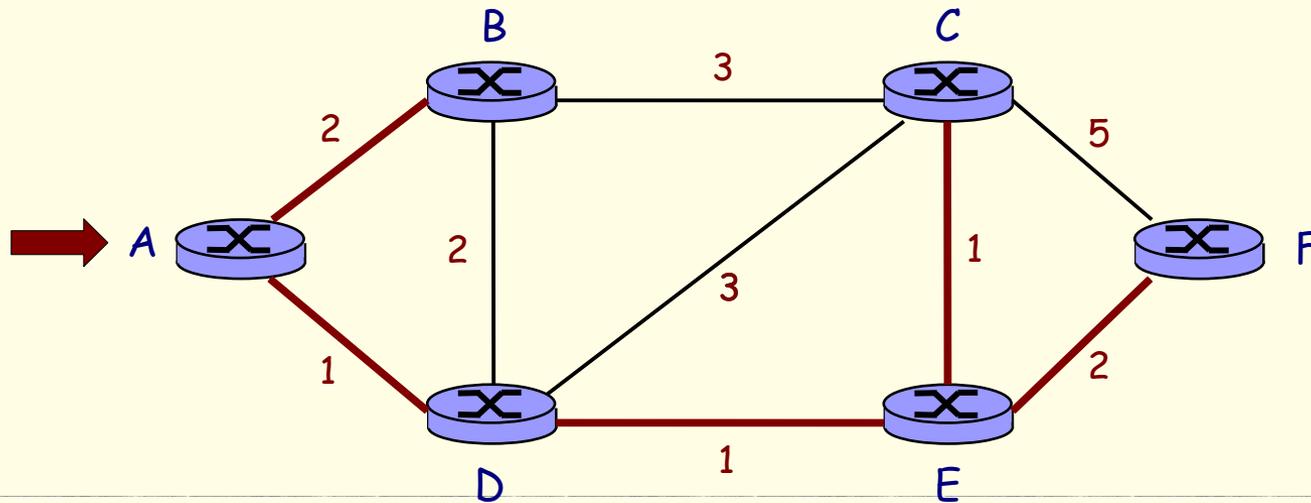
```
N = {A}; // A es el nodo inicial
para todos los nodos v hacer
    si v es adjacente a A entonces
        d(v) = c(A,v)
    si no
        d(v) = ∞;

repetir
    encontrar un w que no esté en N tal que d(w) sea mínimo;
    agregar w a N;
    actualizar d(v) para todos los v adyacentes a w que
        no estén en N haciendo d(v) = min ( d(v), d(w) + c(w,v) );
hasta que N == Q;
```



Traza

Iteración	N	d(B),p(B)	d(C),p(C)	d(D),p(D)	d(E),p(E)	d(F),p(F)
0	A	2,A	∞	1,A	∞	∞
1	AD	2,A	4,D		2,D	∞
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Google no se quiere quedar atrás...

A explicame por favor paso a paso el algoritmo estado de los enlaces de Dijkstra que se usa al rutear en internet

Show drafts ▾

✦ El algoritmo de estado de enlace de Dijkstra en el enrutamiento de Internet: Paso a Paso

El algoritmo de Dijkstra es un algoritmo fundamental en el enrutamiento de Internet, permitiendo a los routers encontrar la ruta más corta entre dos nodos en una red. Se basa en el concepto de estado de enlace, donde cada router comparte información sobre sus enlaces adyacentes con sus vecinos. A partir de esta información, cada router construye una tabla de enrutamiento que contiene la mejor ruta conocida para cada destino.

Pasos del algoritmo:

1. **Inicialización:** Cada router crea una tabla de nodos, donde cada nodo tiene una entrada para cada otro nodo en la red. Inicialmente, la distancia a cada nodo se establece como infinita, excepto para el nodo en sí mismo, que tiene una distancia de 0.
2. **Selección del nodo no visitado con menor distancia:** Se selecciona el nodo de la tabla con la menor distancia conocida, que aún no ha sido visitado. Si no hay nodos no visitados, el algoritmo termina.
3. **Actualización de las distancias:** Para el nodo seleccionado, se calcula la distancia a cada uno de sus nodos adyacentes. La nueva distancia se calcula como la suma de la distancia actual al nodo seleccionado más el costo del enlace al nodo adyacente.

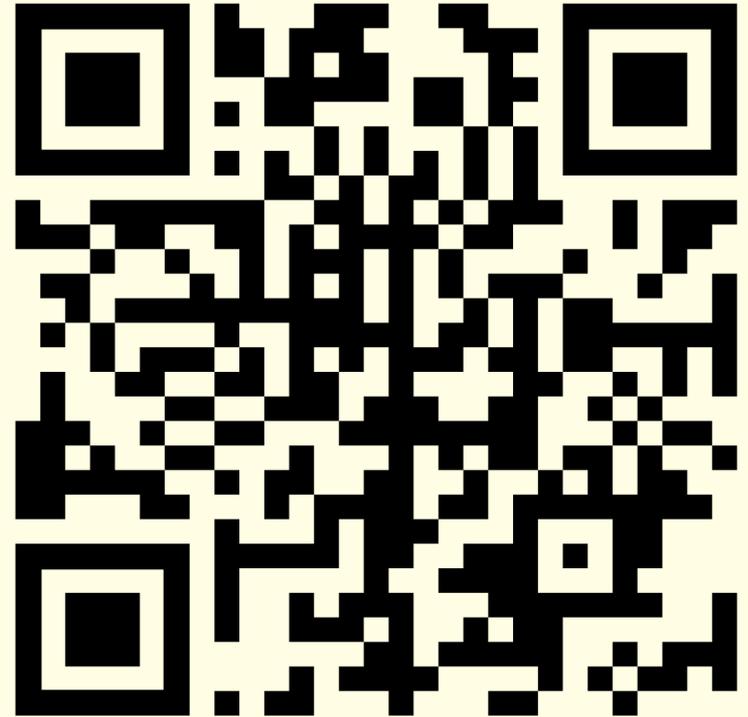
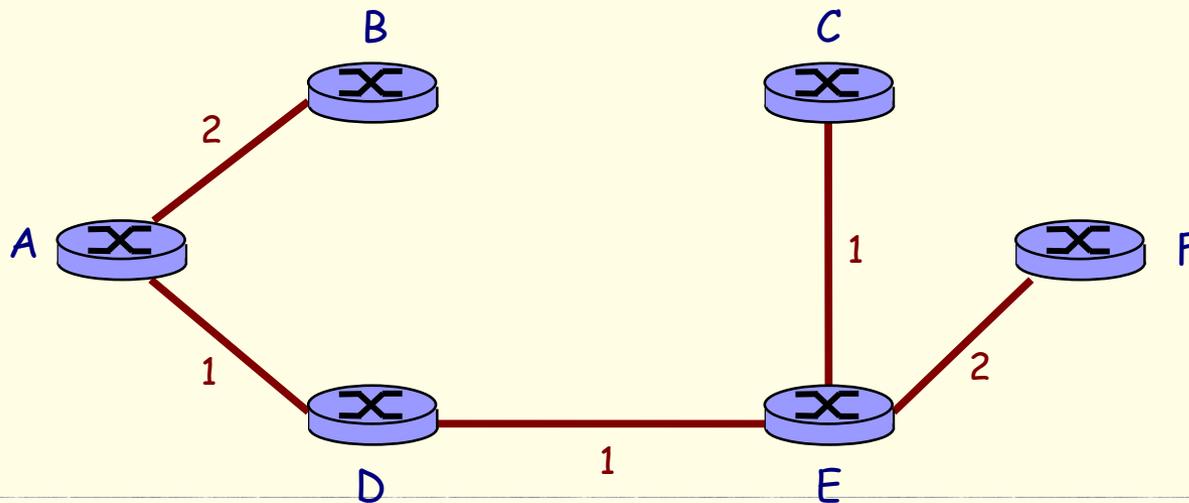


Tabla de forwarding

en base a la información contenida en el árbol cubriente minimal es posible construir la tabla de forwarding asociada

Destino	Enlace
A	-
B	(A, B)
C	(A, D)
D	(A, D)
E	(A, D)
F	(A, D)



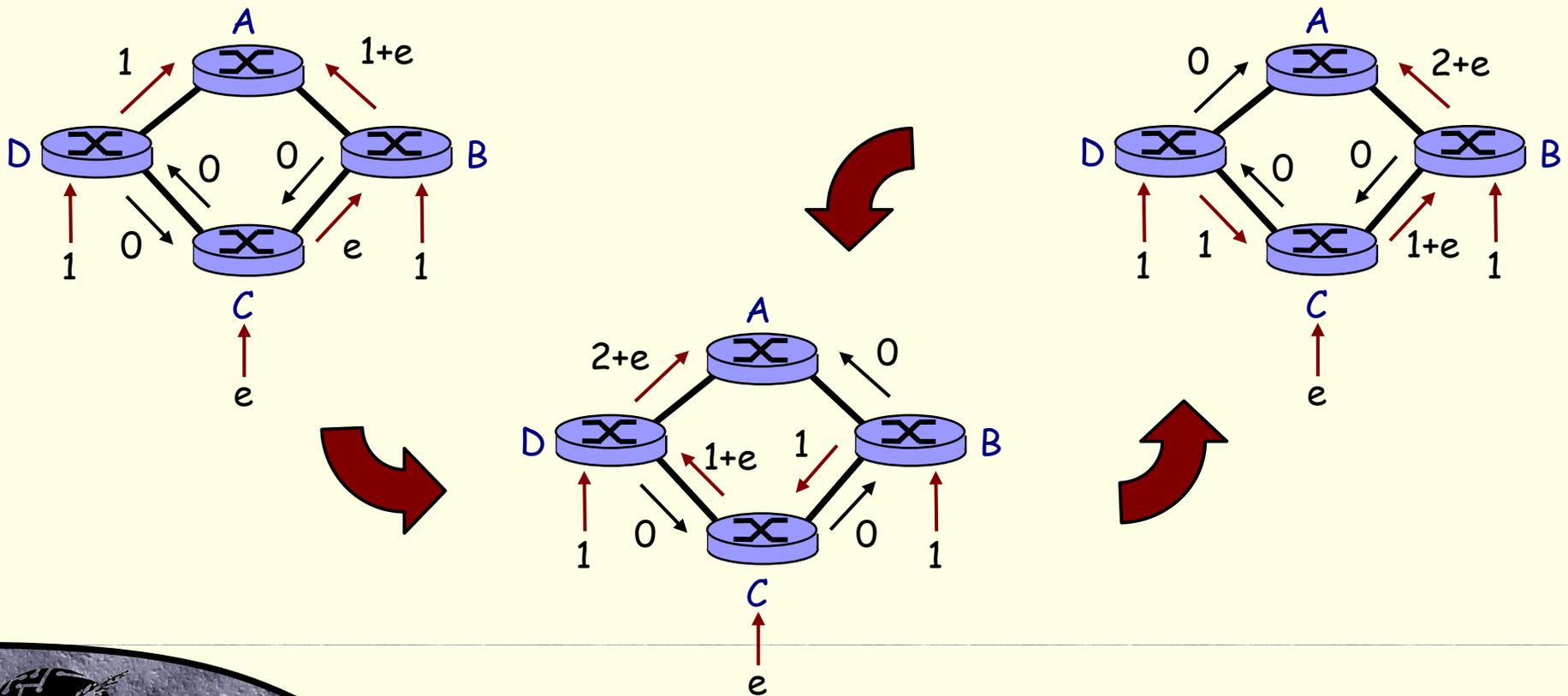
Análisis de la complejidad

- Consideremos la complejidad computacional de este algoritmo para n nodos:
 - El algoritmo debe iterar $n - 1$ veces
 - En cada iteración se comparan todos los nodos w que no están en N (en búsqueda del más próximo)
 - Esta implementación presenta una complejidad computacional de $O((n \times (n - 1)) / 2) = O(n^2)$
 - Se pueden encontrar mejores implementaciones, por caso, usando una pila con prioridad para N se alcanza una complejidad de $O(n \times \log(n))$



Oscilaciones

- El algoritmo de estado de los enlaces puede presentar un **comportamiento oscilatorio**



Vector de distancias

- El algoritmo de ruteo **vector de distancias** resuelve este problema bajo otro enfoque:
 - Es **iterativo**, continúa hasta que los nodos se quedan sin información nueva para propagar
 - Es **asincrónico**, los nodos no necesitan ponerse de acuerdo acerca de en qué momento iterar o cuándo intercambiar información
 - Es **distribuido**, cada nodo sólo se comunica con sus vecinos inmediatos



Algoritmo de Bellman-Ford

- El algoritmo de **Bellman-Ford** resuelve un problema análogo al resuelto por Dijkstra
 - La idea central consiste en aplicar la técnica de **programación dinámica** para mejorar el desempeño del algoritmo naive (esto es, probar uno por uno todos los posibles caminos)
 - Es más general que el de Dijkstra pues permite la existencia de **enlaces con costo negativo**
 - No obstante, resulta un tanto menos eficiente (por eso no se lo utiliza como algoritmo estado de los enlaces)



Vector de distancias

• Un poco de notación:

→ Sea $d_x(y)$ el costo del mejor camino de x a y

→ Sea $c(x, y)$ el costo del enlace entre x e y

• En este contexto, Bellman-Ford postula que el mejor camino x a y puede calcularse recursivamente de la siguiente forma:

→ $d_x(y) = \min(\{ c(x, z) + d_z(y) \})$, para todos los nodos z adyacentes inmediatos a x

→ El algoritmo **vector de distancias** se basa en esta idea

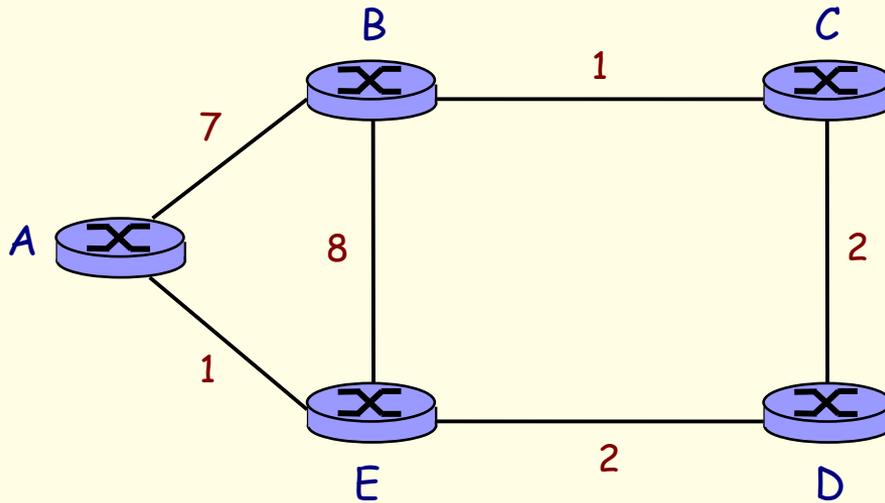


Tabla de distancias

- El algoritmo vector de distancias recopila en la **tabla de distancias** toda la información recogida con respecto a los caminos encontrados
 - Cada nodo cuenta con una tabla de distancias
 - La tabla tiene una fila y una columna para origen y cada destino posible, respectivamente
 - Cada nodo conoce el costo actualizado para cada uno de los enlaces que lo conectan de manera directa con otros nodos



Ejemplo



$$\begin{aligned}d_A(C) &= \text{mín}(\{ c(A,B) + d_B(C), \\ &\quad c(A,E) + d_E(C) \}) \\ &= \text{mín}(\{ 7 + 1 = 8, \\ &\quad 1 + 4 = 5 \}) \\ &= 5\end{aligned}$$

$$\begin{aligned}d_E(D) &= \text{mín}(\{ c(E,A) + d_A(D), \\ &\quad c(E,B) + d_B(D), \\ &\quad c(E,D) + d_D(D) \}) \\ &= \text{mín}(\{ 1 + 3 = 4, \\ &\quad 8 + 3 = 11, \\ &\quad 2 + 0 = 2 \}) \\ &= 2\end{aligned}$$



Actualización de vecinos

- En cada nodo de la red se computa la tabla de distancias y la tabla de forwarding asociada
 - Los nodos quedan a la espera de detectar algún cambio en las condiciones de la red, ya sea a nivel de costo de un enlace o a nivel del vector de distancias de unos de los vecinos adyacentes
 - Cuando se detecta un cambio, se debe **recomputar la tabla de distancias y la de forwarding**
 - En caso de descubrir una nueva mejor ruta (producto del cambio que se detectó), se procede a **propagar la noticia a todos los nodos vecinos inmediatos**



Algoritmo

```
para todos los nodos x e y, hacer  $d_x(y) = \infty$ ;  
para todos los nodos z adyacentes a x, hacer  $d_x(z) = c(x,z)$ ;  
para todos los nodos z adyacentes a x, hacer  
    enviar a z el vector de distancias  $d_x$ .
```

repetir

```
esperar hasta detectar un cambio en el costo  $c(x,y)$  de algún enlace  
o bien recibir de un vecino su vector de distancias actualizado;
```

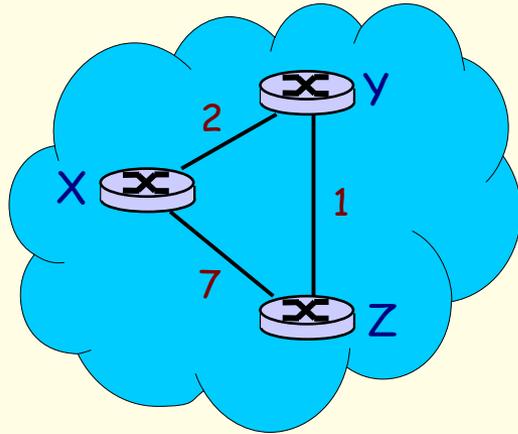
```
para todos los nodos y, hacer  
     $d_x(y) = \min(\{ c(x,z) + d_z(y) \})$ , para todo nodo z;
```

```
si el vector de distancias  $d_x$  cambió, entonces  
    para todos los nodos z adyacentes a x, hacer  
        enviar a z el vector de distancias  $d_x$ .
```

por siempre;



Traza paralela



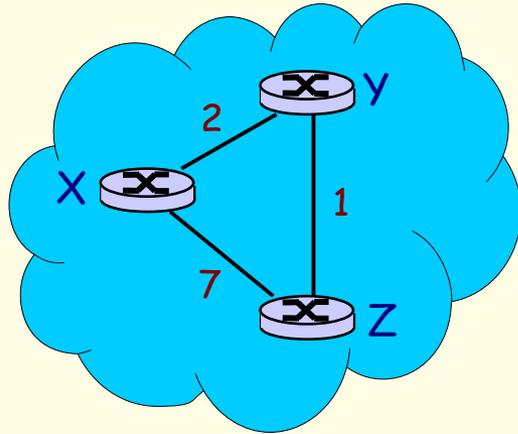
d_x	X	Y	Z
X	0	2	7
Y	∞	∞	∞
Z	∞	∞	∞

d_y	X	Y	Z
X	∞	∞	∞
Y	2	0	1
Z	∞	∞	∞

d_z	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	7	1	0



Traza paralela



d_x	X	Y	Z
X	0	2	7
Y	∞	∞	∞
Z	∞	∞	∞

d_y	X	Y	Z
X	∞	∞	∞
Y	2	0	1
Z	∞	∞	∞

d_z	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	7	1	0

d_x	X	Y	Z
X	0	2	3
Y	2	0	1
Z	7	1	0

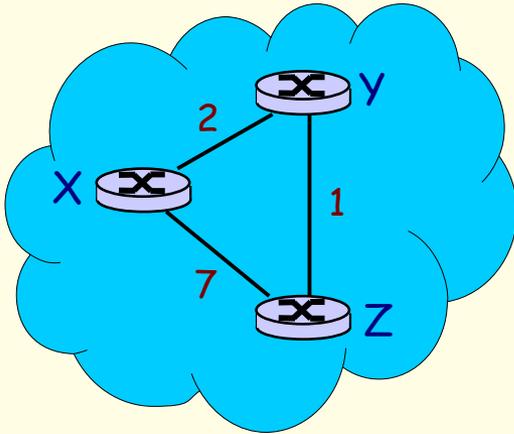
¡cambió!

$$\begin{aligned}
 d_x(Y) &= \min(\{ c(X,Y) + d_y(Y), \\
 &\quad c(X,Z) + d_z(Y) \}) \\
 &= \min(\{ 2 + 0 = 2, \\
 &\quad 7 + 1 = 8 \}) = 2
 \end{aligned}$$

$$\begin{aligned}
 d_x(Z) &= \min(\{ c(X,Y) + d_y(Z), \\
 &\quad c(X,Z) + d_z(Z) \}) \\
 &= \min(\{ 2 + 1 = 3, \\
 &\quad 7 + 0 = 7 \}) = 3
 \end{aligned}$$



Traza paralela



d_x	X	Y	Z
X	0	2	7
Y	∞	∞	∞
Z	∞	∞	∞

d_x	X	Y	Z
X	0	2	3
Y	2	0	1
Z	7	1	0

d_x	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0

d_y	X	Y	Z
X	∞	∞	∞
Y	2	0	1
Z	∞	∞	∞

d_y	X	Y	Z
X	0	2	7
Y	2	0	1
Z	7	1	0

d_y	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0

d_z	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	7	1	0

d_z	X	Y	Z
X	0	2	7
Y	2	0	1
Z	3	1	0

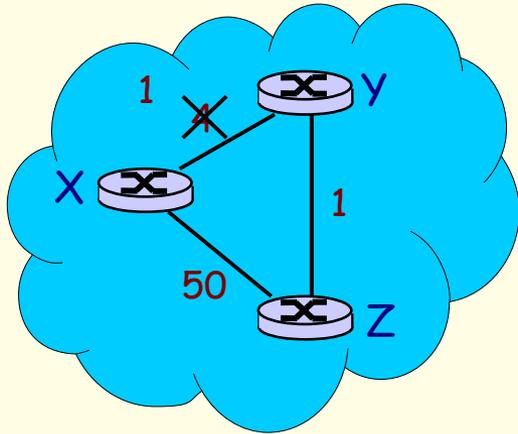
d_z	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0

Cambio en el costo del enlace

- A medida que pasa el tiempo la naturaleza dinámica de la red puede manifestarse
 - Por caso, el **costo de un cierto enlace puede cambiar** sin previo aviso
 - El nodo involucrado se da cuenta y **recalcula su tabla de distancias y de ruteo**
 - Si el cambio en el costo **modifica a alguno de los mejores caminos**, se procede como es usual a **notificar a los nodos adyacentes**



Propagación de novedades



Las buenas noticias se propagan rápido!

d_Y	X	Y	Z
Y	4	0	1
Z	5	1	0

d_Y	X	Y	Z
Y	1	0	1
Z	5	1	0

d_Y	X	Y	Z
Y	1	0	1
Z	5	1	0

d_Y	X	Y	Z
Y	1	0	1
Z	2	1	0

d_Z	X	Y	Z
Y	4	0	1
Z	5	1	0

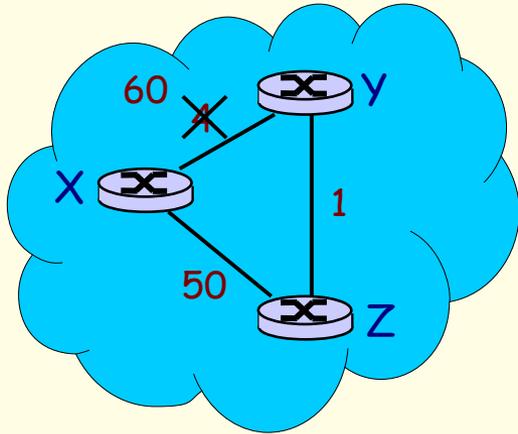
d_Z	X	Y	Z
Y	4	0	1
Z	5	1	0

d_Z	X	Y	Z
Y	1	0	1
Z	2	1	0

d_Z	X	Y	Z
Y	1	0	1
Z	2	1	0



Propagación de novedades



ilas malas noticias
no necesariamente!

d_y	X	Y	Z
Y	4	0	1
Z	5	1	0

d_y	X	Y	Z
Y	6	0	1
Z	5	1	0

d_y	X	Y	Z
Y	6	0	1
Z	5	1	0

d_y	X	Y	Z
Y	8	0	1
Z	7	1	0

d_y	X	Y	Z
Y	8	0	1
Z	7	1	0

d_z	X	Y	Z
Y	4	0	1
Z	5	1	0

d_z	X	Y	Z
Y	4	0	1
Z	5	1	0

d_z	X	Y	Z
Y	6	0	1
Z	7	1	0

d_z	X	Y	Z
Y	6	0	1
Z	7	1	0

d_z	X	Y	Z
Y	8	0	1
Z	9	1	0

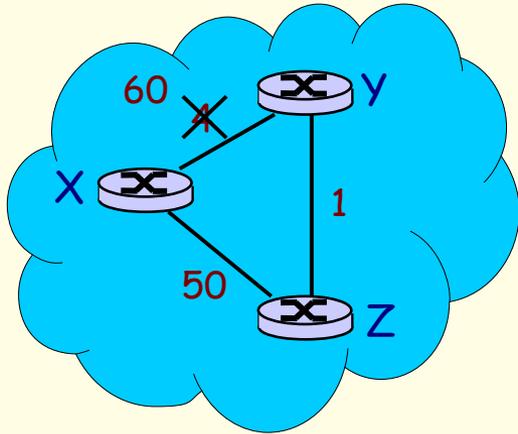


Cuenta al infinito

- Este problema es producto de las “demoras” en la actualización global de las rutas
 - Nótese que existe una ruta de **Y** a **X** a través de **Z**, la cual involucra ir hacia **Z** desde **Y** para luego volver a **Y** y de ahí recién ir a **X**
 - Ante una actualización del coste de un enlace, **Z** debería dejarle en claro a **Y** que como lo usa para ir a **X**, de hecho no constituye un camino alternativo
 - La clave radica en notar que **Z** debería informarle a **Y** que su mejor camino a **X** tiene costo infinito, pues involucra volver a **Y**



Envenenamiento de rutas



el algoritmo termina en tres iteraciones! no obstante, ¿cubre a todos los casos?

d_y	X	Y	Z
Y	4	0	1
Z	∞	∞	0

d_y	X	Y	Z
Y	60	0	1
Z	∞	∞	0

d_y	X	Y	Z
Y	60	0	1
Z	∞	∞	0

d_y	X	Y	Z
Y	51	0	1
Z	50	∞	0

d_y	X	Y	Z
Y	51	0	1
Z	50	∞	0

d_z	X	Y	Z
Y	4	0	∞
Z	5	1	0

d_z	X	Y	Z
Y	4	0	∞
Z	5	1	0

d_z	X	Y	Z
Y	60	0	∞
Z	50	1	0

d_z	X	Y	Z
Y	60	0	∞
Z	50	1	0

d_z	X	Y	Z
Y	∞	0	∞
Z	50	1	0



Link state vs. vector de dist.

- Resulta instructivo comparar los algoritmos link state y vector de distancias en base al siguiente conjunto de criterios
- En relación a la **cantidad de mensajes** que deben ser transmitidos:
 - ➔ Para **n** nodos y **e** enlaces, el algoritmo link state intercambia $O(n \times e)$ mensajes entre todos los nodos
 - ➔ Para el algoritmo vector de distancia sólo se intercambia mensajes entre los nodos adyacentes



Link state vs. vector de dist.

- En relación a la **velocidad de convergencia** hacia la solución:
 - Link state implementado en $O(n^2)$ hace uso de $O(n \times e)$ mensajes, los cuales una vez recibidos permiten al algoritmo encontrar la solución
 - Con vector de distancias la situación es diferente, si bien la propagación de buenas noticias es rápida, también puede caer en el problema de la **cuenta al infinito**, lo que ralentiza la convergencia



Link state vs. vector de dist.

- En relación a la **robustez de la infraestructura de red** al hacer uso de estos algoritmos:
 - ➔ En el caso de link state un router que falla puede **publicitar mal el costo de un enlace** (afectando a todos) o **calcular mal sólo su tabla**
 - ➔ En el caso de vector de distancias un router que falla puede **publicitar mal el costo de los caminos a través del mismo**, pero como esa información es usada por otros routers, **el error se termina propagando por toda la red**



Tarea para el hogar

- Llevar adelante una comparativa entre dos algoritmos que resuelven una misma tarea de dos maneras contrapuestas no es trivial
- Sin embargo, es el tipo de tarea en la que brillan los **LLMs** tipo **ChatGPT** o **Bard**
 - ➔ Tarea para el hogar: explorar qué tipo de consultas hay que formular para generar una comparativa que resulte instructiva



¿Preguntas?

