Enrutamiento

Introducción y Distance-Vector

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
Grado en Ingeniería en Tecnologías de Telecomunicación, 2º

Temario

1. Introducción
2. Arquitecturas de conmutación y protocolos
3. Introducción a las tecnologías de red
4. Control de acceso al medio
5. Conmutación de circuitos
6. Transporte fiable
7. Encaminamiento

Temario

1. Introducción
2. Arquitecturas de conmutación y protocolos
3. Introducción a las tecnologías de red
4. Control de acceso al medio
5. Conmutación de circuitos
6. Transporte fiable
7. **Encaminamiento**

Un algoritmo para encontrar caminos

- Algoritmo de Bellman-Ford
- Conocemos: Nodos i Nodo raiz r pesos w(i,j)
- Mantenemos: $d_h(i)$ mejor distancia de r al nodo i conocida hasta ahora

  $s_h(i)$ siguiente salto del nodo i hacia r (mejor conocido hasta ahora)

  h numero de saltos que hemos considerado

  $d_h(i)$ es la mejor distancia de i al nodo r dando como mucho h saltos

Algoritmo:

Inicializar

$d_0(i)$=infinito $d_0(r)$=0 [en 0 saltos solo desde r se puede llegar a r]

$s_0(i)$=desconocido $s_0(r)$= no hace falta siguiente salto yo soy la raiz

Para h=1,2,3,4 .... hasta cuando?

Para cada uno de los nodos i

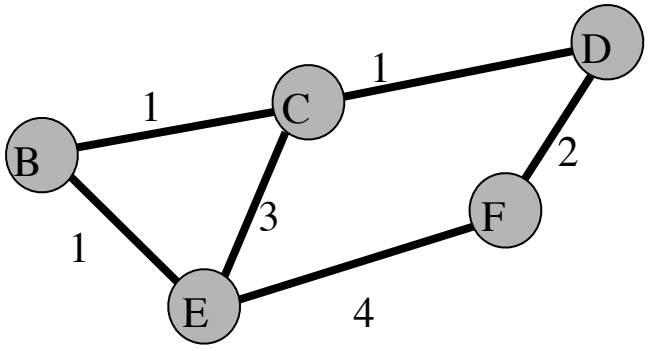
Para cada uno de los nodos k [vecinos de i]

$d_h(i)=\min\{ d_{h-1}(i)+w(i,k) \}$

$s_h(i)$= el k con que se obtiene el minimo [nada si son todos infinito]

Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc				
2					
3					
4					
5 ...					



salen todos infinitos

La minima distancia de B a D dando como mucho un salto es infinito no se puede llegar en un salto

- Para cada nodo por ejemplo para el B

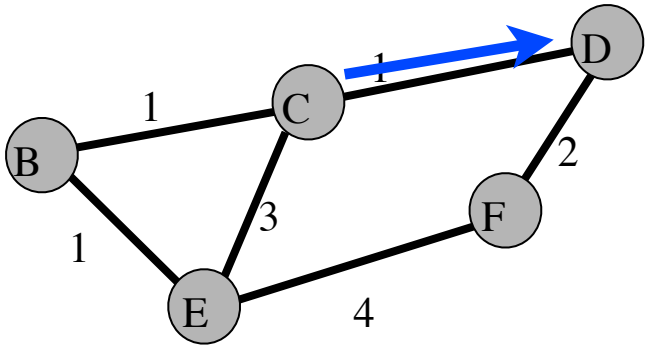
  $d_1(B)$ en la siguiente fila depende de la fila actual y las distancias

  $d_0(B)+w(B,B)$ $d_0(C)+w(B,C)$ $d_0(D)+w(B,D)$ $d_0(E)+w(B,E)$ $d_0(F)+w(B,F)$

  inf + 0 inf + 1 0 + inf inf + 1 inf + inf

Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D			
2					
3					
4					
5 ...					



El unico que no sale infinito es el D con distancia 1

La minima distancia de C a D dando como mucho un salto es 1

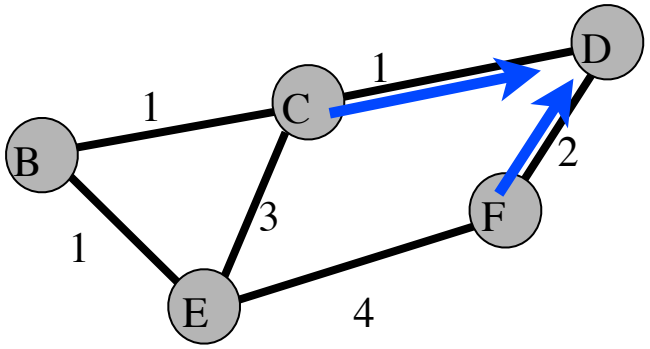
- Para el C

$d_1(C)$ en la siguiente fila depende de la fila actual

$d_0(B)+w(C,B)$	$d_0(C)+w(C,C)$	$d_0(D)+w(C,D)$	$d_0(E)+w(C,E)$	$d_0(F)+w(C,F)$
inf + 1	inf + 0	0 + 1	inf + 3	inf + inf

Bellman-Ford ejemplo

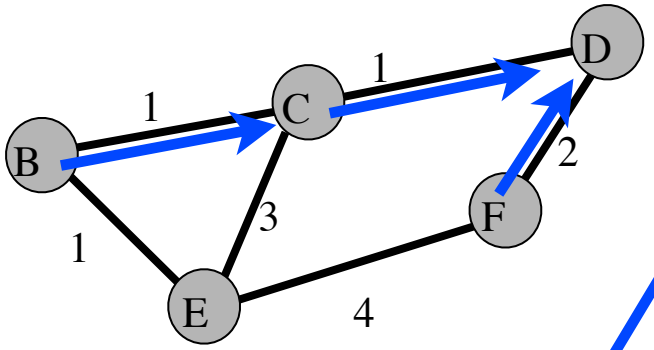
h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2					
3					
4					
5 ...					



- En una iteracion tenemos el arbol de los mejores caminos a D con un salto
- En este caso es ya el camino correcto pero no tiene por que ser (si la raiz hubiera sido C tendriamos que el mejor camino de E a C con un salto es de distancia 3)

Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C				
3					
4					
5 ...					



Ahora a traves de C tenemos distancia 2

En realidad no hace falta probar mas que con los vecinos
En los que no son vecinos de B siempre dara infinito

- Segunda iteración para el B

$d_2(B)$

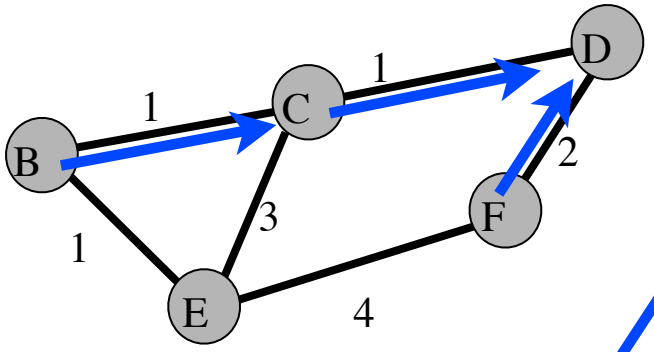
$d_1(B)+w(B,B)$ $d_1(C)+w(B,C)$ $d_1(D)+w(B,D)$ $d_1(E)+w(B,E)$ $d_1(F)+w(B,F)$

inf + 0 1 + 1 0 + inf inf + 1 2 + inf

Este es B tampoco es vecino

Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C	1 / D			
3					
4					
5 ...					



Este es C no es un candidato

Me vuelve a salir mejor el camino via D

- Para cada nodo por ejemplo para el C

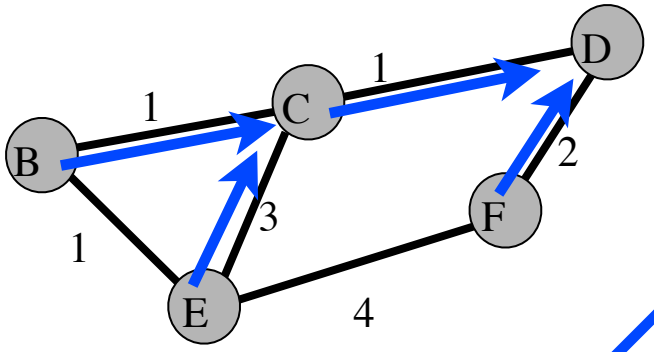
$d_2(C)$

$d_1(B)+w(C,B)$ ~~$d_1(C)+w(C,C)$~~ $d_1(D)+w(C,D)$ $d_1(E)+w(C,E)$ ~~$d_1(F)+w(C,F)$~~

inf + 1 ~~1 + 0~~ 0 + 1 inf + 3 ~~2 + inf~~

Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C	1 / D	0 / soy yo	4 / C	
3					
4					
5 ...					



E elige entre
ir a traves de C 3+d(C) = 4 <<<< este
ir a traves de F 4+d(F) = 6

- Para cada nodo por ejemplo para el E

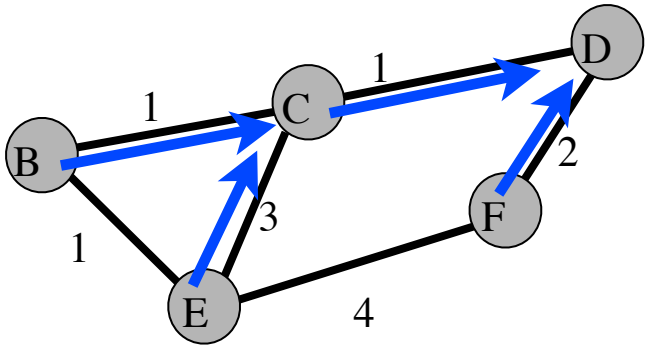
$d_1(B)+w(E,B)$ $d_1(C)+w(E,C)$ $d_1(D)+w(E,D)$ $d_1(E)+w(E,E)$ $d_1(F)+w(E,F)$

inf + 1 1 + 3 0 + inf inf + 0 2 + 4

B no es un candidato porque el no sabe como llegar en la iteracion 1

Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C	1 / D	0 / soy yo	4 / C	2 / D
3					
4					
5 ...					



- Al final de la segunda iteración tenemos el arbol de caminos mínimos a D usando como mucho 2 saltos
- Con este ya tenemos un camino desde cada nodo !!!
- Aunque esta claro que no son los mejores. Por ejemplo de E llegaríamos antes via B

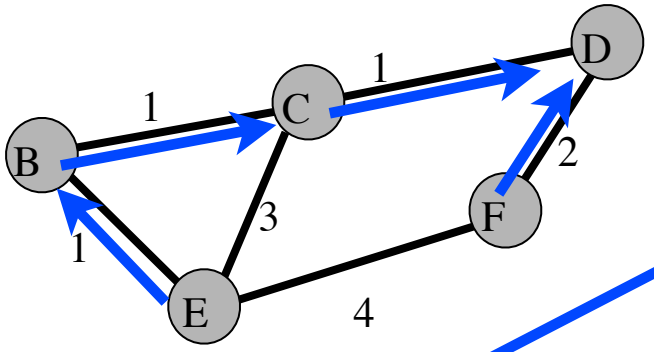
Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C	1 / D	0 / soy yo	4 / C	2 / D
3					
4					
5 ...					

- **Cuando acaba el algoritmo???**
- No vale decir cuando tengamos todos los caminos con coste mínimo. Como hacemos que un programa haga la comprobación?
- El objetivo del algoritmo es saber cuales son los caminos de coste minimo, asi que no podemos usar el conocimiento de cual es el coste minimo para resolverlo
- **Entonces**
- El algoritmo calcula cada fila d_h en funcion de la fila anterior dh-1
- Una vez que una fila se repita... va a seguir repitiendose eternamente...
- Esa es la condición que nos dice que el algoritmo ya no va a encontrar caminos mejores
- Bellman-Ford demostraron que cuando pasa eso ya ha encontrado los mejores

Bellman-Ford ejemplo

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C	1 / D	0 / soy yo	4 / C	2 / D
3	2 / C	1 / D	0 / soy yo	3 / B	
4					
5 ...					



Hay un mejor camino por B con distancia 3

- Iteración 3 para el E

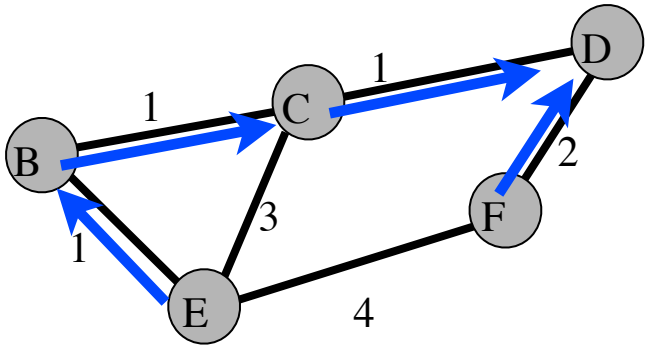
$d_3(B)+w(E,B)$	$d_3(C)+w(E,C)$	$d_3(D)+w(E,D)$	$d_3(E)+w(E,E)$	$d_3(F)+w(E,F)$
2 + 1	1 + 3	0 + inf	4 + 0	2 + 4

E elige entre ir por cualquiera de sus vecinos que ya tienen una distancia a D

Aunque en este caso es ya el mínimo no tendría por que ser

Bellman-Ford ejemplo

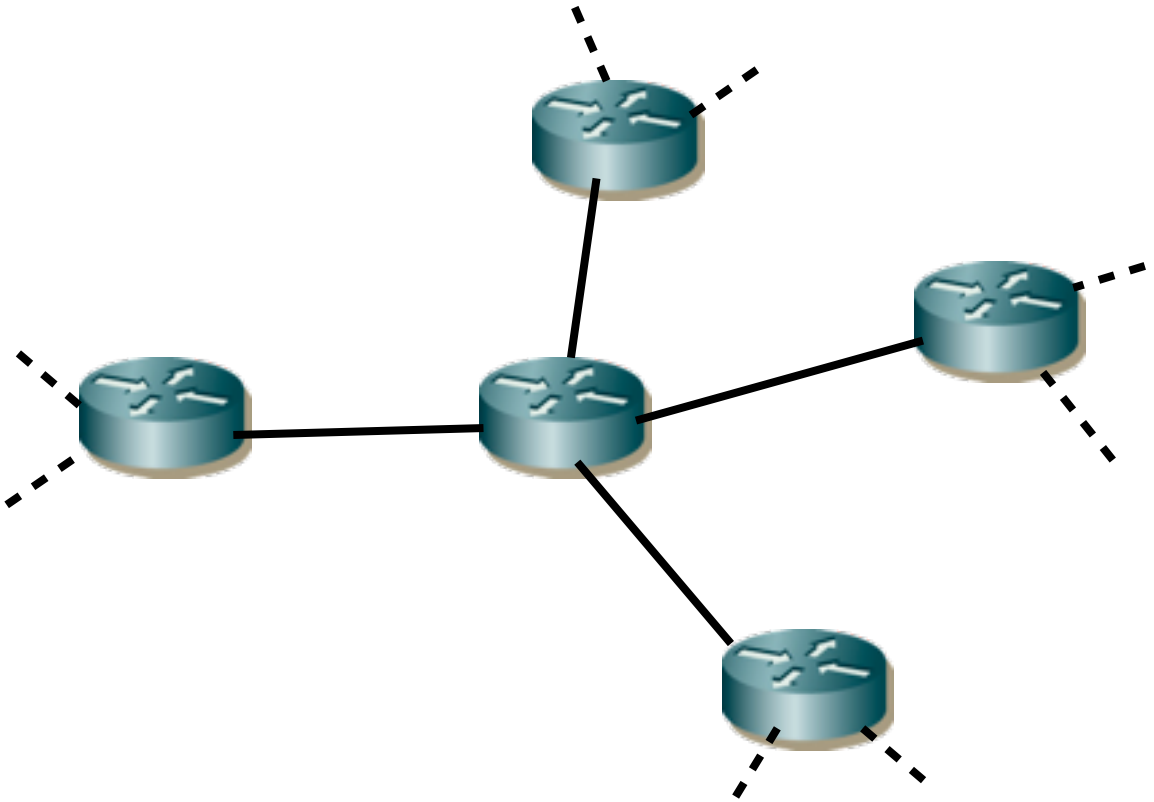
h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C	1 / D	0 / soy yo	4 / C	2 / D
3	2 / C	1 / D	0 / soy yo	3 / B	2 / D
4	2 / C	1 / D	0 / soy yo	3 / B	2 / D
5 ...					



- Tras la tercera iteración ya tenemos el arbol de caminos minimos con 3 saltos
- Que ya es el definitivo pero el algoritmo no lo sabe
- Al hacer la iteración 4 no hay ningún cambio
- Esa es la condición de que ya ha terminado

Como usar esto en la realidad

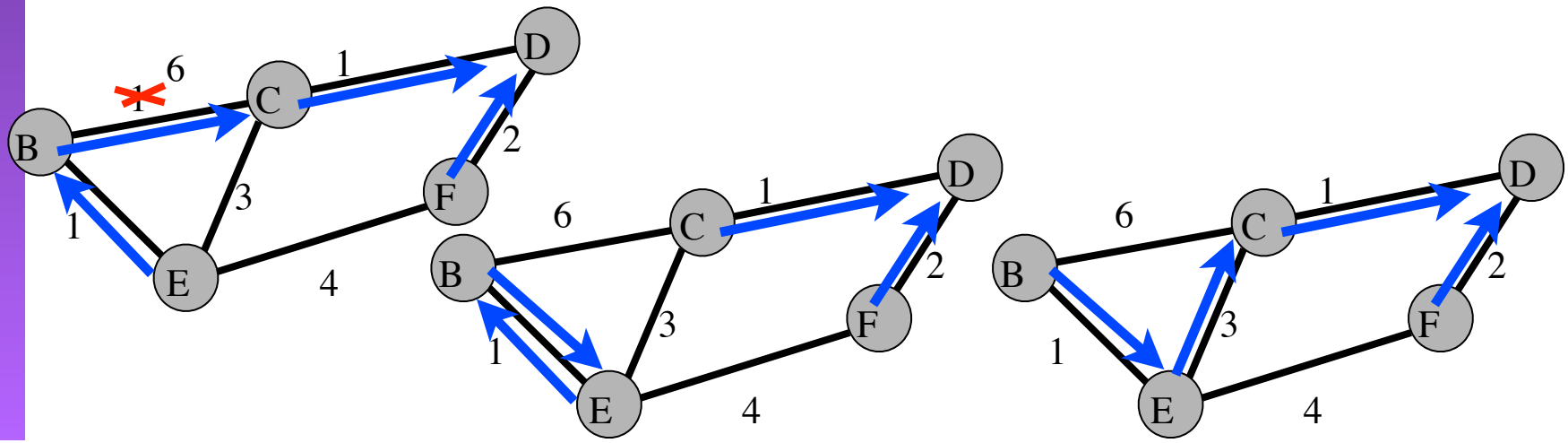
- El problema del cruce de caminos... En cada router
- Como calculo las rutas a todos los destinos?
- Solo con información mía o que pueda obtener de los vecinos
- Ni siquiera se cuales son todos los destinos !!!



Bellman-Ford continuo...

h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
12032	2 / C	1 / D	0 / soy yo	3 / B	2 / D
12033	2 / C	1 / D	0 / soy yo	3 / B	2 / D
12034	4 / E	1 / D	0 / soy yo	3 / B	2 / D
12035	4 / E	1 / D	0 / soy yo	4 / C	2 / D
12036	5 / E	1 / D	0 / soy yo	4 / C	2 / D
12037...	5 / E	1 / D	0 / soy yo	4 / C	2 / D

- Si hay un cambio? Enlace B-C cambia a peso 6
- parece que se adapta a el ... en unos pocos turnos

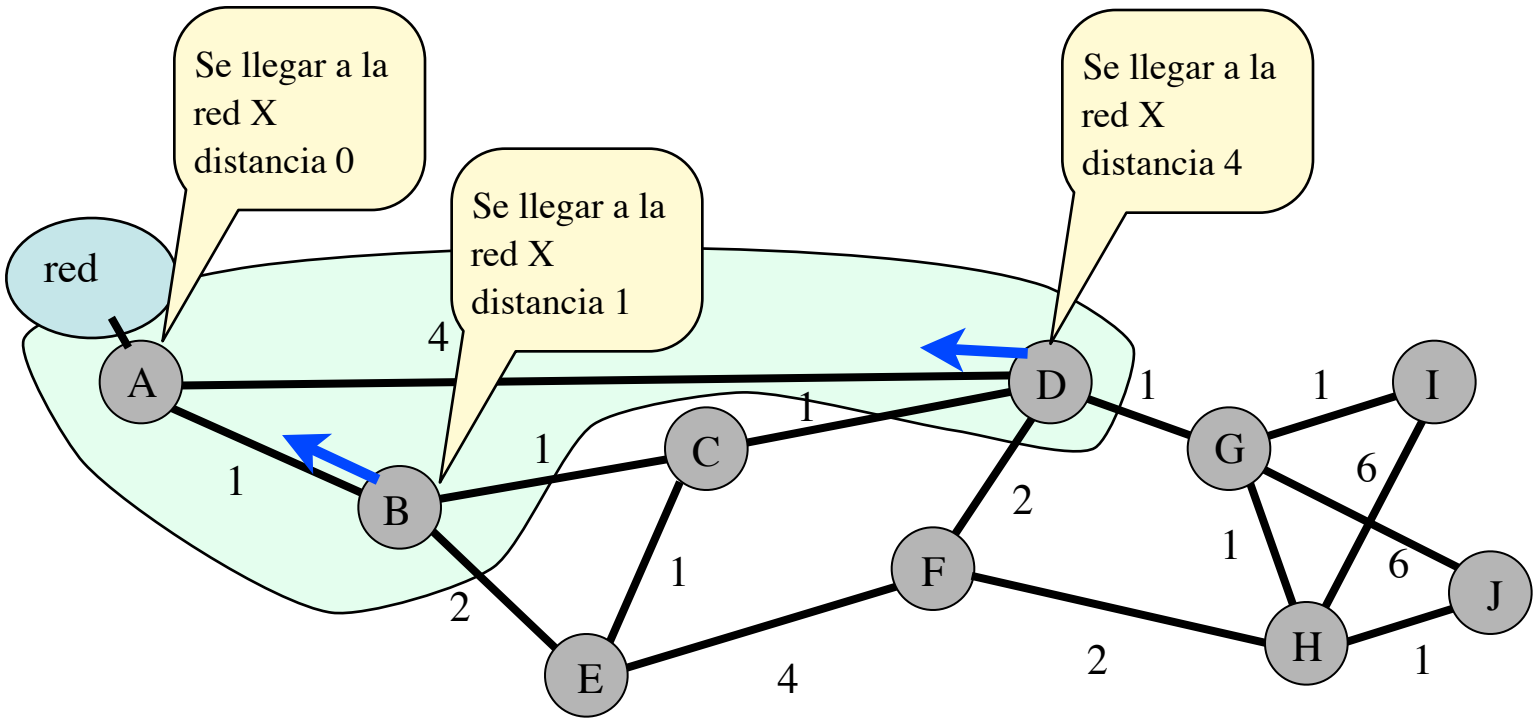


Distance-Vector

- La idea es la base para construir algoritmos de enrutamiento distribuidos
- Enrutamiento Distance-Vector
  - Cada router esta configurado con un identificador
  - Cada router esta configurado con los enlaces a sus vecinos y cada enlace tiene por configuración un numero para usar como coste del enlace
  - Cada router mantiene un vector de distancias a cada destino, se inicializa con 0 para si mismo y infinito para cualquier otro destino
  - Mantiene también otro vector con el siguiente salto a cada destino (tabla de rutas)
  - Cada router es capaz de comunicarse con sus vecinos y envía el vector de distancias a sus vecinos (cada vez que hay un cambio o periodicamente)
  - Cada router almacena el vector de distancias más reciente que ha recibido de cada vecino y utiliza la iteración de Bellman-Ford para decidir cual es el la mejor distancia y el mejor siguiente salto a ese destino.
  - Con eso actualiza su distancia y siguiente salto hacia ese destino
  - Cuando se recalcula el vector?
    - Cada vez que recibo de la red información nueva (nuevo vector)
    - Cada vez que cambia el peso de un enlace, o desaparece (cambio a infinito)
- Es una aproximación distribuida al algoritmo de Bellman-Ford
  - Salvo que no hay exactamente iteraciones o turnos...
  - De hecho (solo mensajes de algunos vecinos) parece razonable
  - Pero hay algunos problemas...

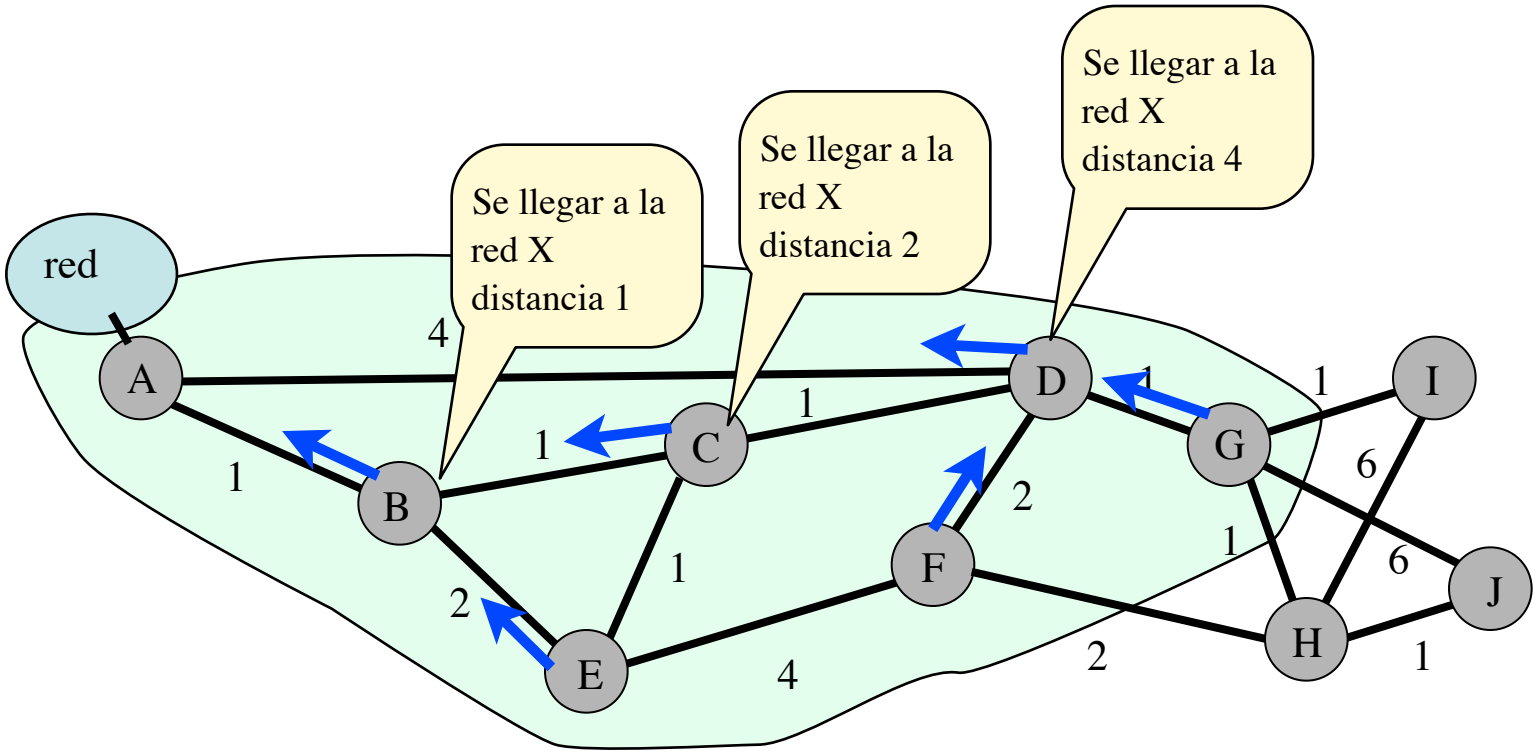
Y esto funciona?

- La información para cada destino se propaga desde los routers que la saben...
-



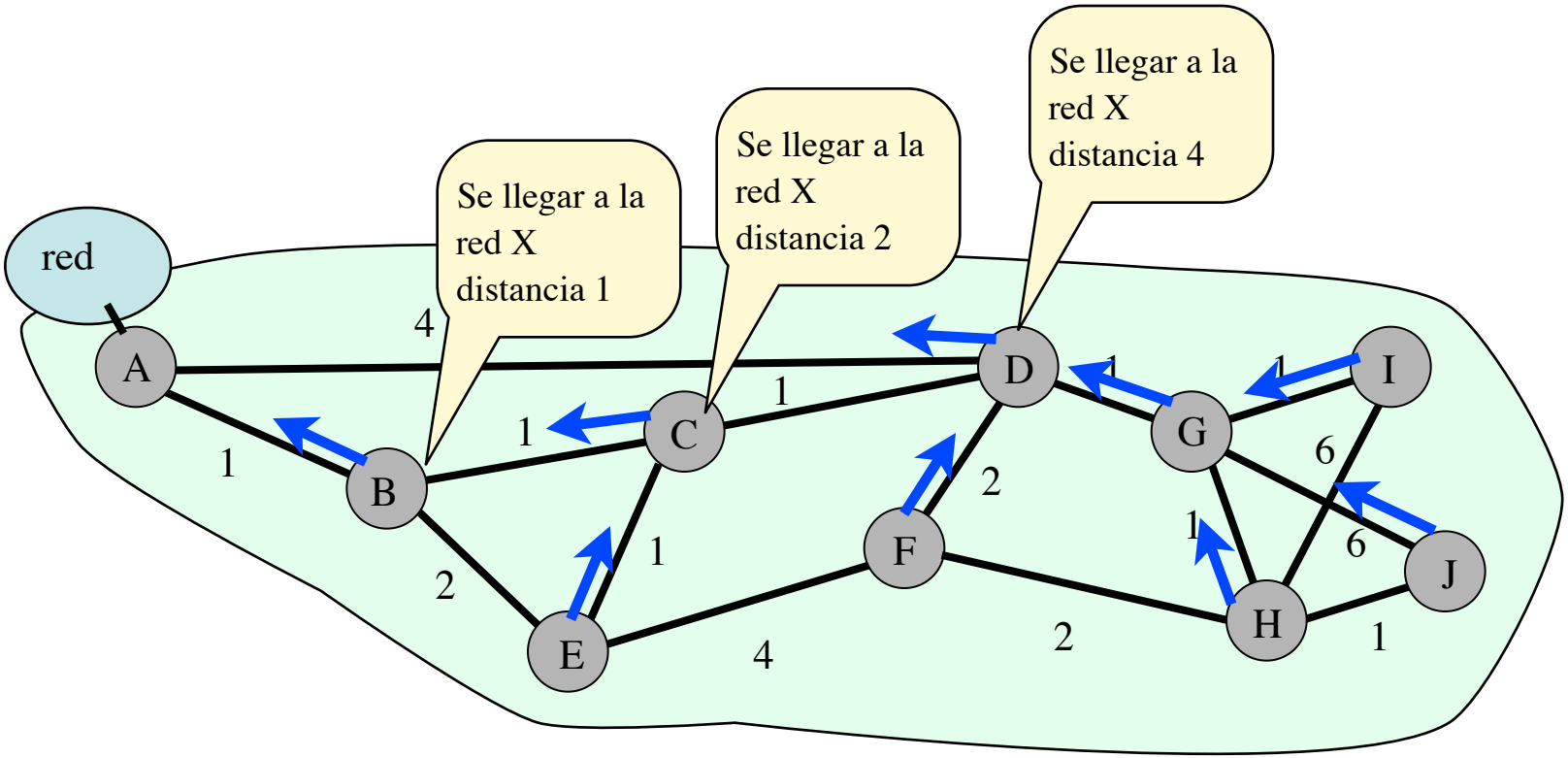
Y esto funciona?

- La información para cada destino se propaga desde los routers que la saben...
-



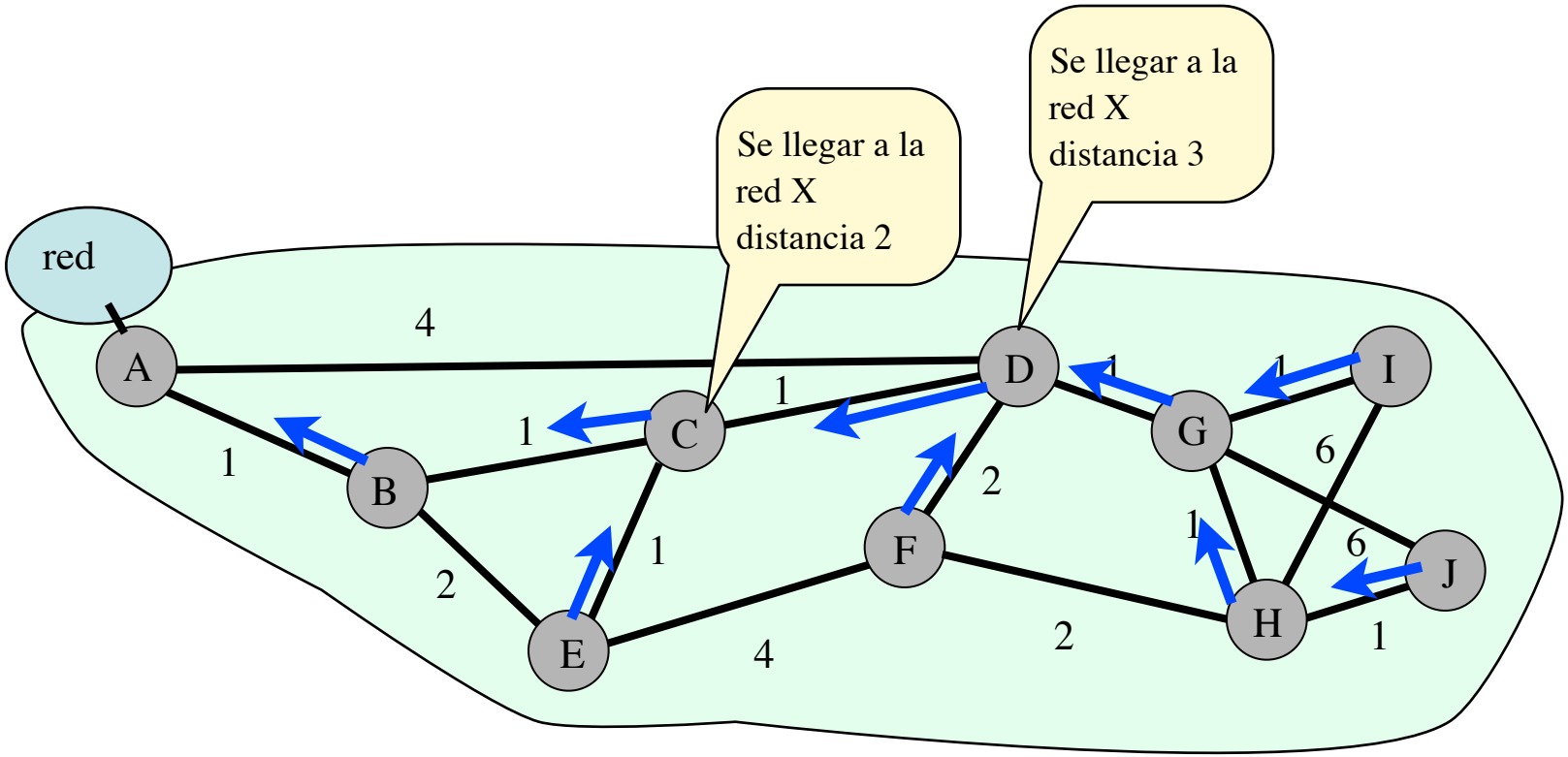
Y esto funciona?

- La información para cada destino se propaga desde los routers que la saben...
- El tiempo de propagación depende de cuantos routers hay hasta el destino
- El tiempo de propagación no es tanto si cada router envía la información a sus vecinos cada vez que hay cambios (triggered updates)



Y esto funciona?

- La información para cada destino se propaga desde los routers que la saben...
- No necesariamente la mejor ruta es la que oigo la primera vez por eso el algoritmo debe funcionar de forma continua
- Lo mismo pasa a la vez con todos los demas destinos



Y esto funciona?

- El algoritmo no sabe cuando ha terminado
- Pero no importa mucho que se ejecute de forma continua
- Si usamos envío de información periódica controlamos el trafico de enrutamiento que se envía... pero el tiempo en propagar rutas es mas largo
- Si enviamos en cuanto hay cambios (triggered updates) la propagación es rapida y se envía más tráfico cuando hay un cambio pero es self-stoping se autocontrola y deja de enviar cuando las rutas se estabilizan
- Normalmente se utiliza triggered updates con y envio periodico no demasiado frecuente
  - Envío rapido de cambios
  - El envio periodico ayuda si se pierden mensajes o para descubrir vecinos cuando un router se conecta a la red
- Parece razonable. Funciona, se propaga rapido y no crea mucho trafico...
- Y entonces por que es el sistama de **enrutamiento *antiguo* de Internet**
- Es lento en propagar algunos cambios, los cambios a peor dan problemas de estabilidad

Resumen hasta ahora

- Algoritmo distance-vector teorico
- Calculo de rutas distribuidas
- Adaptación a los cambios
- Convergencia rápida a los cambios y auto-parada en algunos casos
  - Normalmente los cambios a mejor se propagan rápido
- Problemas de convergencia/estabilidad y cuentas a infinito en otros casos
  - Normalmente los cambios a peor se propagan despacio
- Soluciones que alivian estos problemas pero no los resuelven totalmente

- Lo suficiente para que los algoritmos distance-vector sean útiles
- Qué protocolos reales distance-vector se utilizan? (RIP, IGRP, EIGRP...)
- Como conseguir algoritmos con menos problemas de convergencia?
  (La semana que viene)

Conclusiones

- Protocolos distance-vector permiten construir enrutamiento adaptativo distribuido
  - Basados en el algoritmo de Bellman-Ford
  - Son útiles para redes no demasiado complicadas
- Protocolos reales se verán en Redes de Ordenadores

- Próxima clase:

  Otro tipo de algoritmos de enrutamiento: link-state