Enrutamiento

Link-state...

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

Arquitectura de Redes, Sistemas y Servicios
3º Ingeniería de Telecomunicación

Temario

- Introducción
- Arquitecturas, protocolos y estándares
- Conmutación de paquetes
- Conmutación de circuitos
- Tecnologías
- Control de acceso al medio en redes de área local
- Servicios de Internet

Temario

- Introducción
- Arquitecturas, protocolos y estándares
- Conmutación de paquetes
  - Principios
  - Problemas básicos
    - Como funcionan los routers (Nivel de red)
    - **Encaminamiento (Nivel de red)**
    - Transporte fiable (Nivel de transporte en TCP/IP)
    - Control de flujo (Nivel de transporte en TCP/IP)
    - Control de congestión (Nivel de transoporte en TCP/IP)
- Conmutación de circuitos
- Tecnologías
- Control de acceso al medio en redes de área local
- Servicios de Internet

Algoritmos para encontrar caminos

- Algoritmo de **Dijkstra** para la distancia mínima
- Conocemos: Nodos i Nodo raiz r pesos w(i,j)
- Mantenemos: d(i) mejor distancia de r al nodo i conocida hasta ahora

  T conjunto de nodos a los que ya conocemos la distancia mas corta

  d(i) es la distancia menor de r hasta el nodo i pasando solo por nodos que están en T
- Algoritmo:

Inicializar

d(i)=infinito d(r)=0 si el nodo i es vecino de r [w(i,r)<infinito] d(i)=w(i,r)

T={r}

Mientras haya nodos que no pertenezcan a T

elegir el nodo i que no este en T con menor d(i)

añadir el nodo i a T

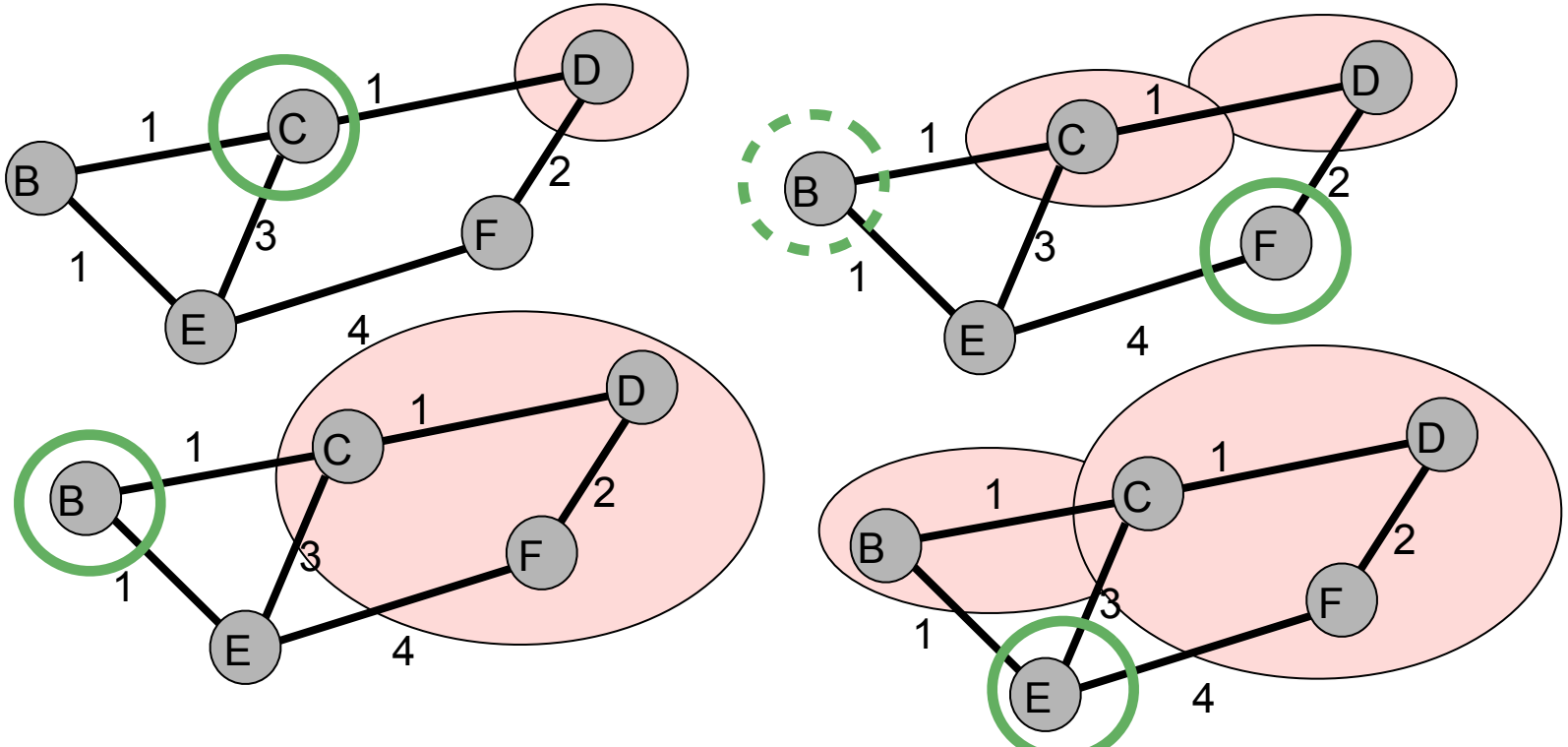
actualizar d(k) de los nodos vecinos al nodo i que no están en T.

Si d(i)+w(k,i) < d(k) entonces d(k)=d(i)+w(k,i)

[es menor la distancia pasando por i que la que ya tenia]

Dijkstra ejemplo

T	d(B)	d(C)	d(D)	d(E)	d(F)
{D}	infinito	1	0	infinito	2
{D,C}	**2**	1	0	**4**	2
{D,C,F}	2	1	0	4	2
{D,C,F,B}	2	1	0	**3**	2
{D,C,F,B,E}	2	1	0	3	2



Manteniendo el camino

- Usando el algoritmo de **Dijkstra** para el camino mínimo
- Conocemos: Nodos i Nodo raiz r pesos w(i,j)
- Mantenemos: d(i) mejor distancia de r al nodo i conocida hasta ahora
  s(i) siguiente nodo a i en el camino hacia r
  T conjunto de nodos a los que ya conocemos la distancia mas corta

Algoritmo:

Inicializar

d(i)=infinito d(r)=0 si el nodo i es vecino de r [w(i,r)<infinito] d(i)=w(i,r)

T={r}

s(i)=desconocido si i es vecino de r [w(i,r)<infinito] s(i)=r

Mientras haya nodos que no pertenezcan a T

elegir el nodo i que no este en T con menor d(i)

añadir el nodo i a T

actualizar d(k) de los nodos vecinos al nodo i que no están en T.

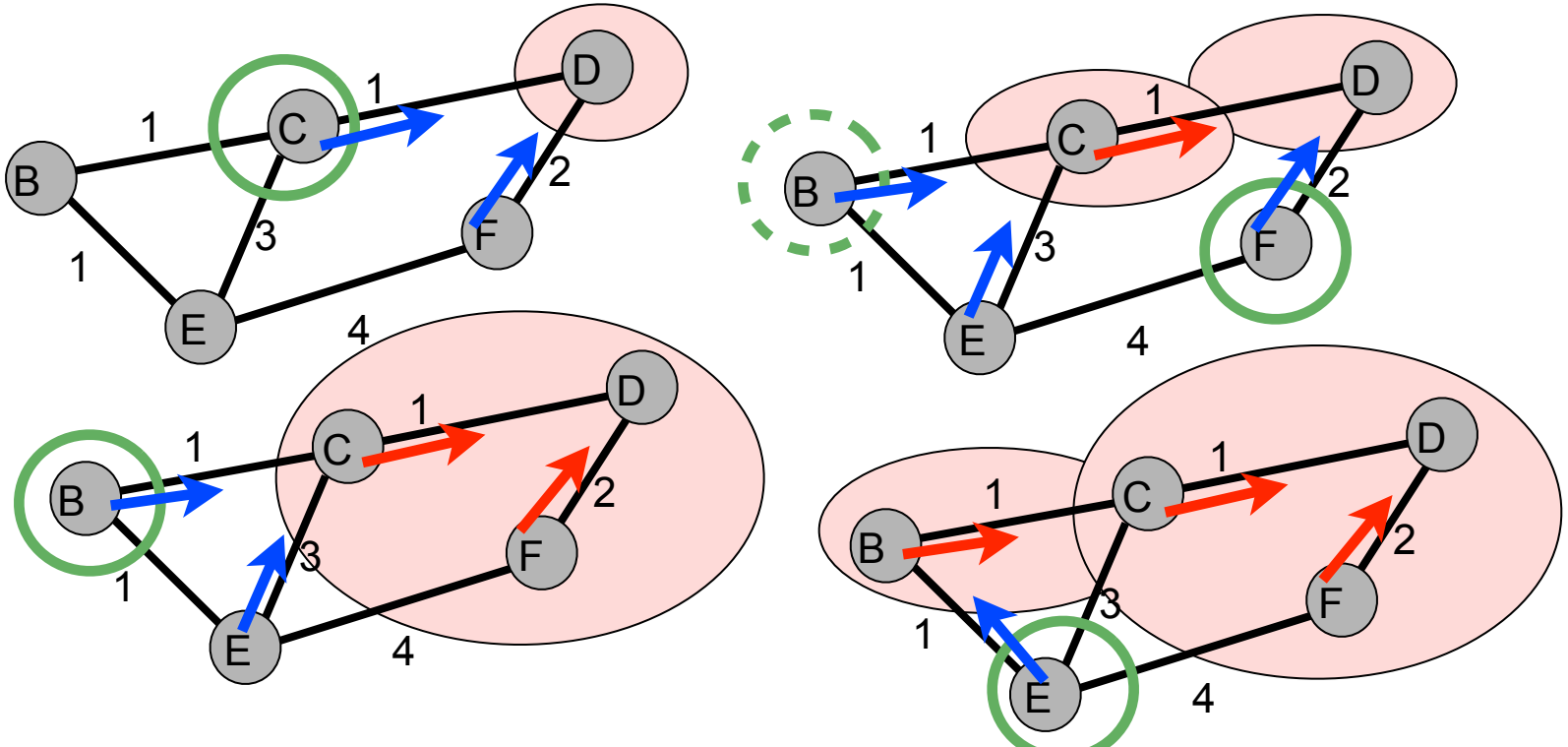
Si d(i)+w(k,i) < d(k) entonces d(k)=d(i)+w(k,i)

y tambien s(k)=i

[si el camino por i es mejor i es el nuevo siguiente salto de k]

Dijkstra ejemplo con camino

T	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
{D}	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
{D,C}	**2 / C**	1 / D	0 / soy yo	**4 / C**	2 / D
{D,C,F}	2 / C	1 / D	0 / soy yo	4 / C	2 / D
{D,C,F,B}	2 / C	1 / D	0 / soy yo	**3 / B**	2 / D
{D,C,F,B,E}	2 / C	1 / D	0 / soy yo	3 / B	2 / D

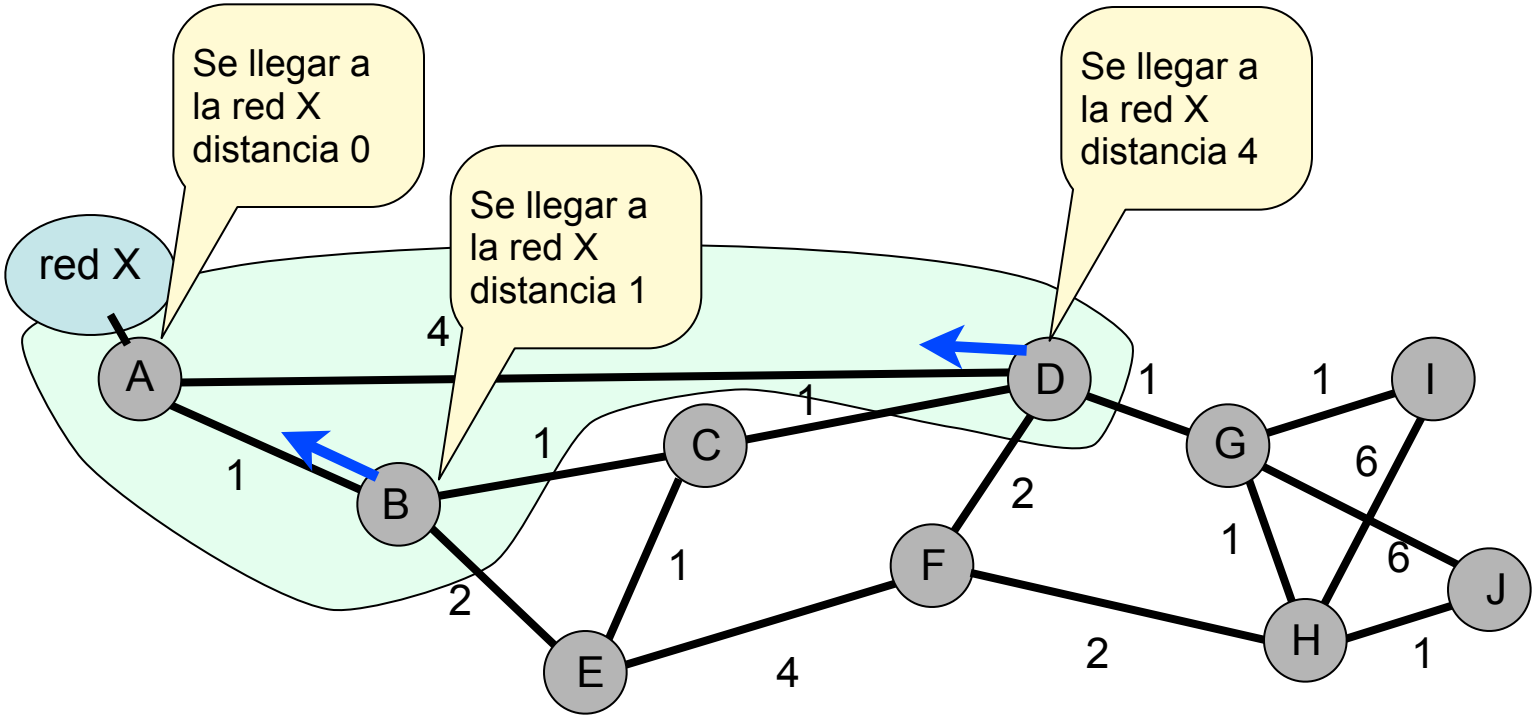


Algoritmos de caminos mínimos

- Dijkstra vs Bellman-Ford
  - Los dos calculan el árbol de expansión mínimo para una raiz dada
  - Los dos algoritmos dan el mismo resultado
  - El resultado no tiene por que ser único (probad a hacer los ejemplos anteriores cambiando el peso de C-E a 2)
- Cuál es más rápido?
  - Parece que Bellman-Ford hace menos iteraciones pero las iteraciónes de Dijkstra parecen más cortas y rápidas
- Cuál preferiríais programar?
- Normalmente se suele considerar mejor el algoritmo de Dijkstra para resolver el problema de los caminos en un grafo
- Bellman-Ford permite construir algoritmos distribuidos basados en el concepto distance-vector

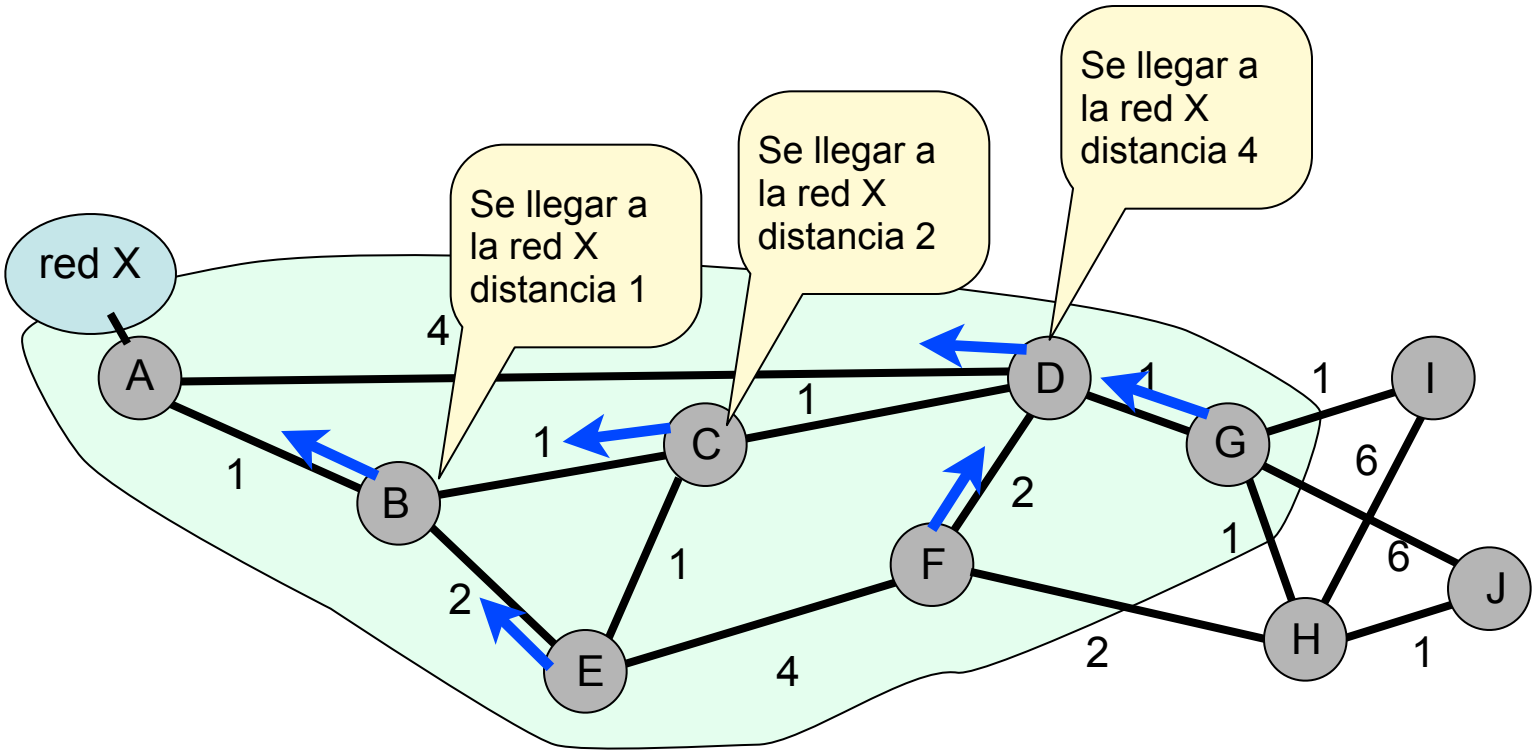
Funciona Distance-Vector?

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



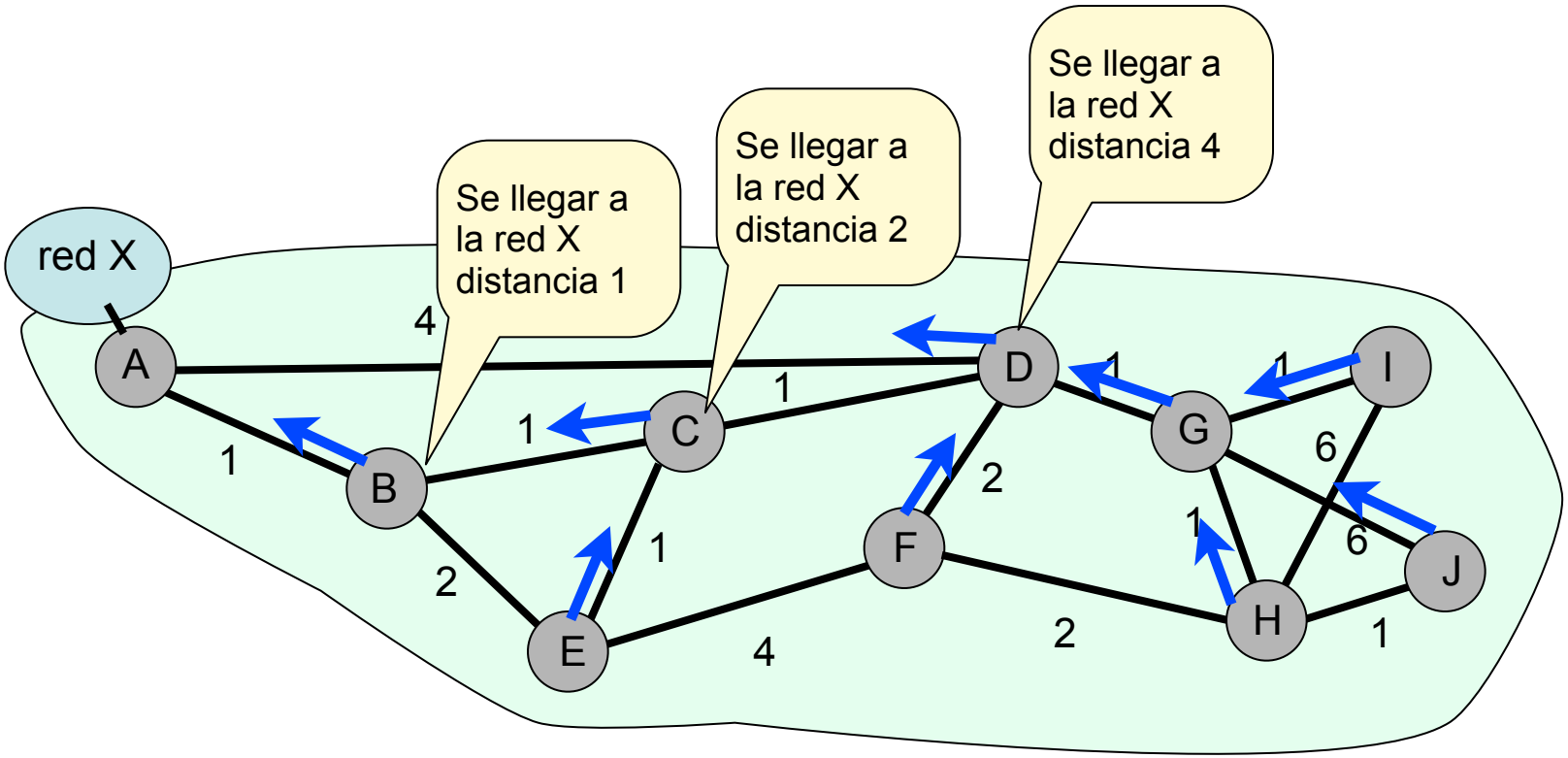
Funciona Distance-Vector?

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



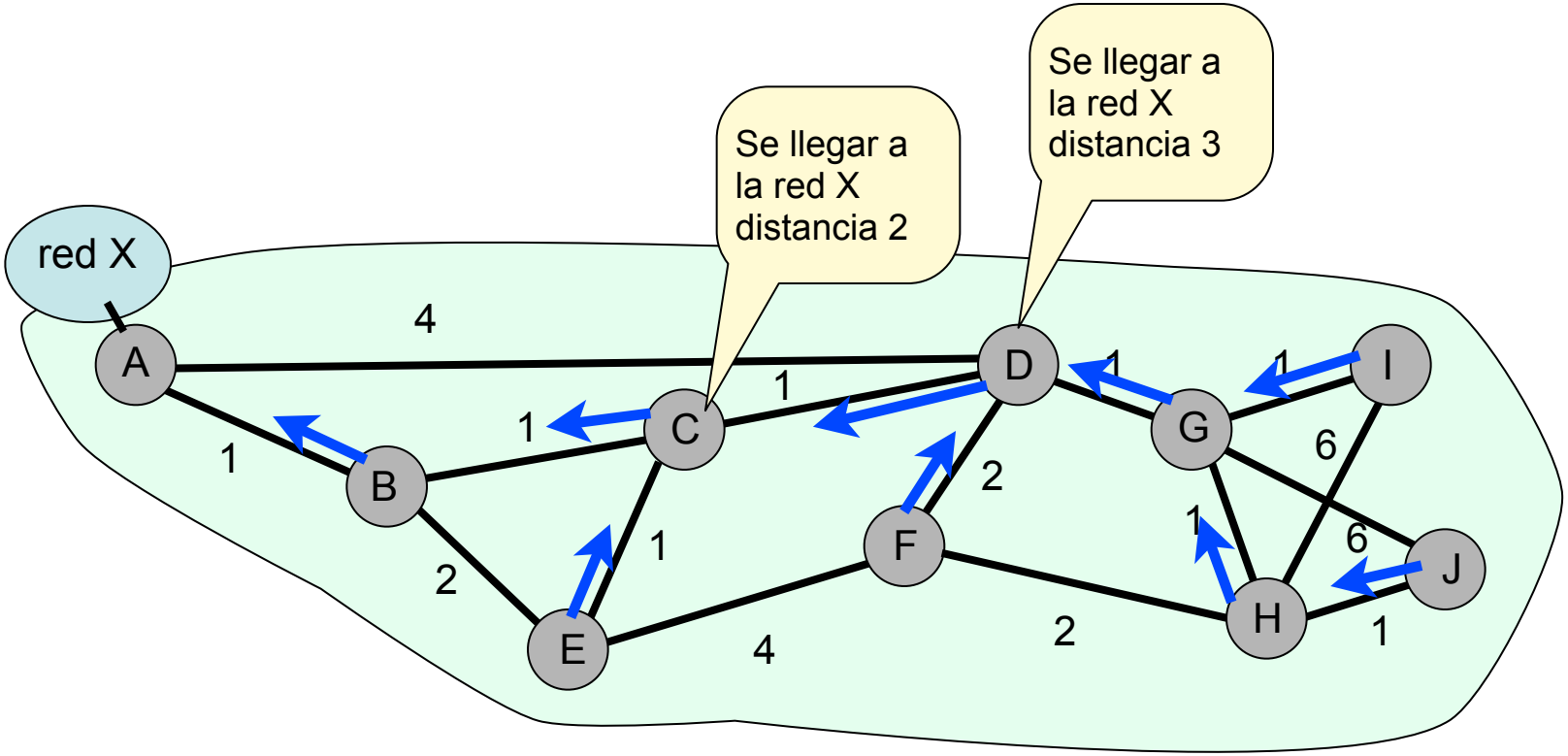
Funciona Distance-Vector?

- 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)



Funciona Distance-Vector?

- 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



Funciona Distance-Vector?

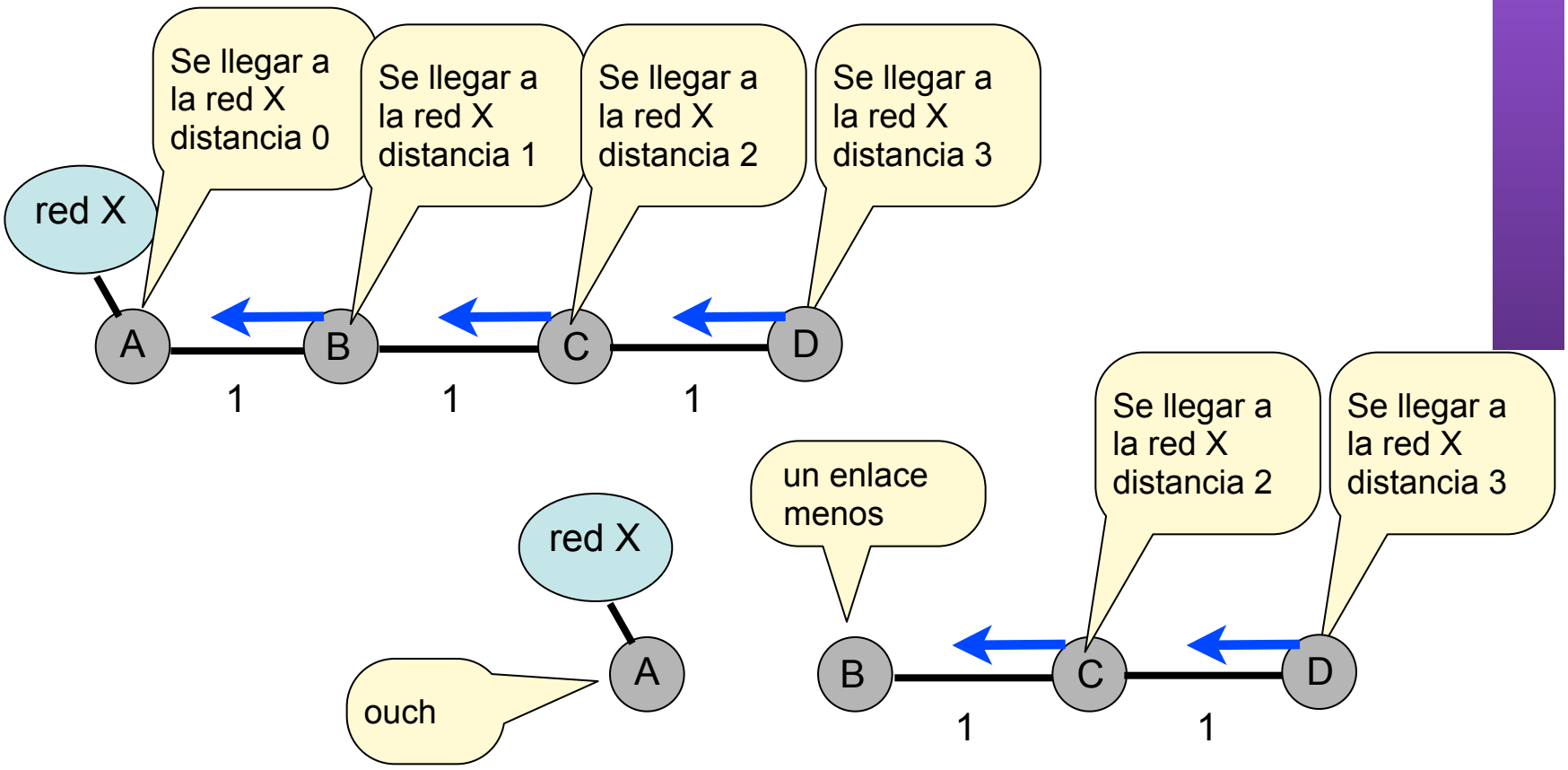
- El algoritmo no sabe cuando ha terminado
- Pero no importa mucho que se ejecute de forma continua
- Si usamos envío de información periódica controlamos el trafico de enrutamiento que se envía... pero el tiempo en propagar rutas es mas largo
- Si enviamos en cuanto hay cambios (triggered updates) la 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**

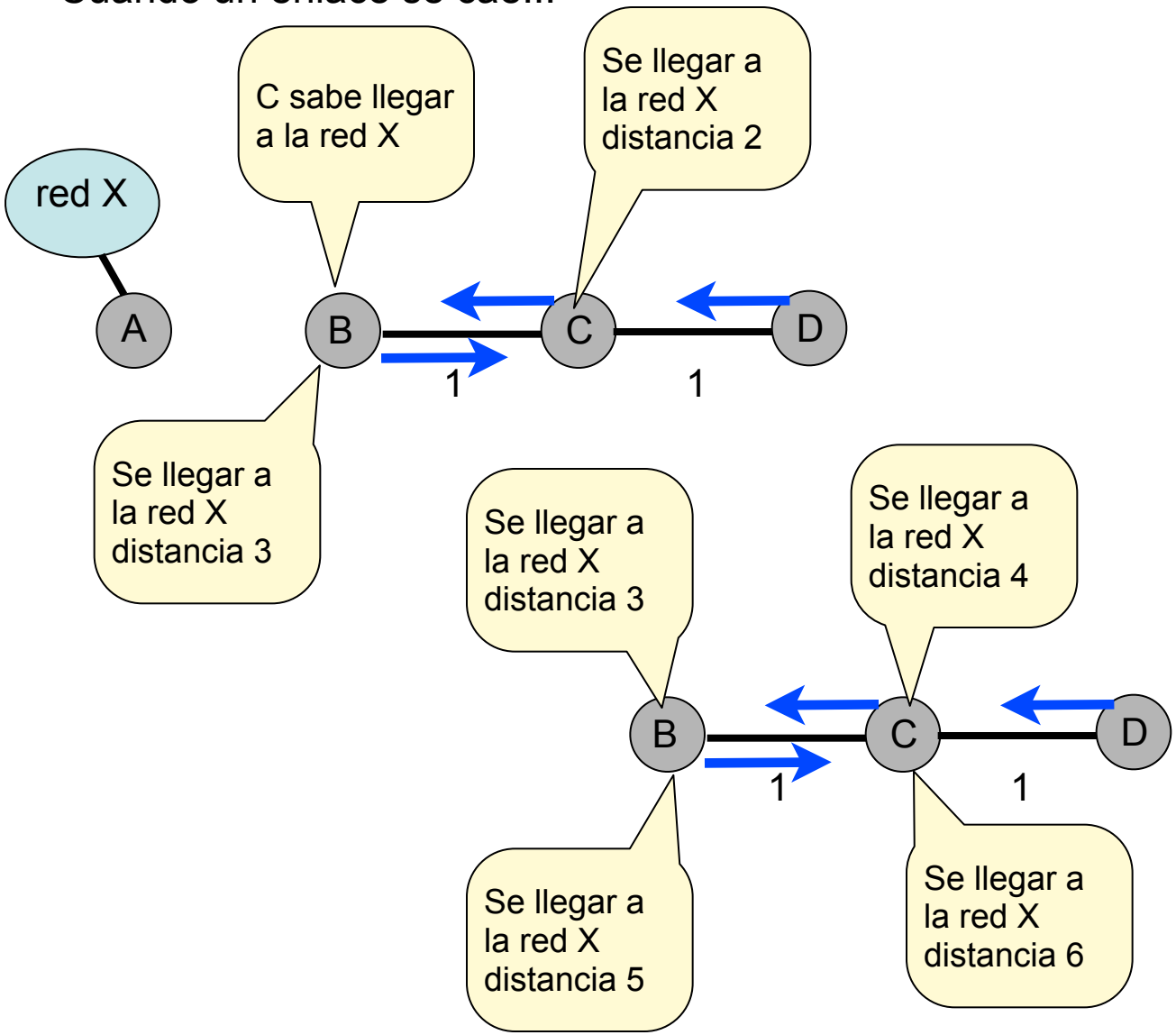
Problemas

- Es lento en reaccionar !!!
- No todos los cambios se propagan rápido hay situaciones anómalas
- Un caso muy simple y bien conocido
- Qué pasa si se cae el primer enlace?



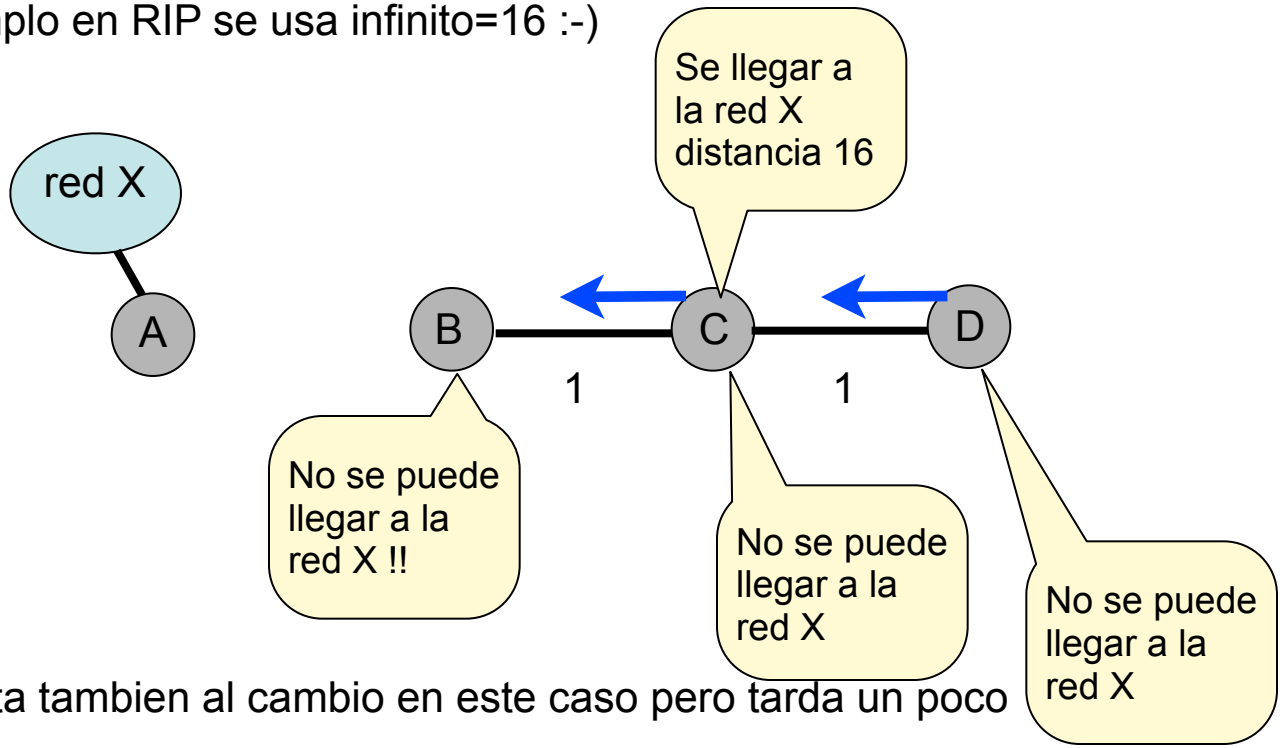
La cuenta a infinito

- Cuando un enlace se cae...



La cuenta a infinito

- Como acaba esto?
- Se hace que el campo para indicar la distancia tendrá bits limitados
  Cuando llegue al máximo se considera infinito y la ruta se descarta
  Establcer un valor de infinito pequeño hace que estos casos se detecten antes
  Pero entonces solo podremos calcular distancias menores a ese valor
  Por ejemplo en RIP se usa infinito=16 :-)



- Se adapta tambien al cambio en este caso pero tarda un poco y genera unos cuantos mensajes de actualización en el proceso

Problemas distance-vector

- Las cuentas a infinito hacen que los protocolos distance-vector puedan tardar en bastante en converger a la solución
- Mientras convergen las rutas puede no ser buenas

  (ciclos de enrutamiento y perdidas)

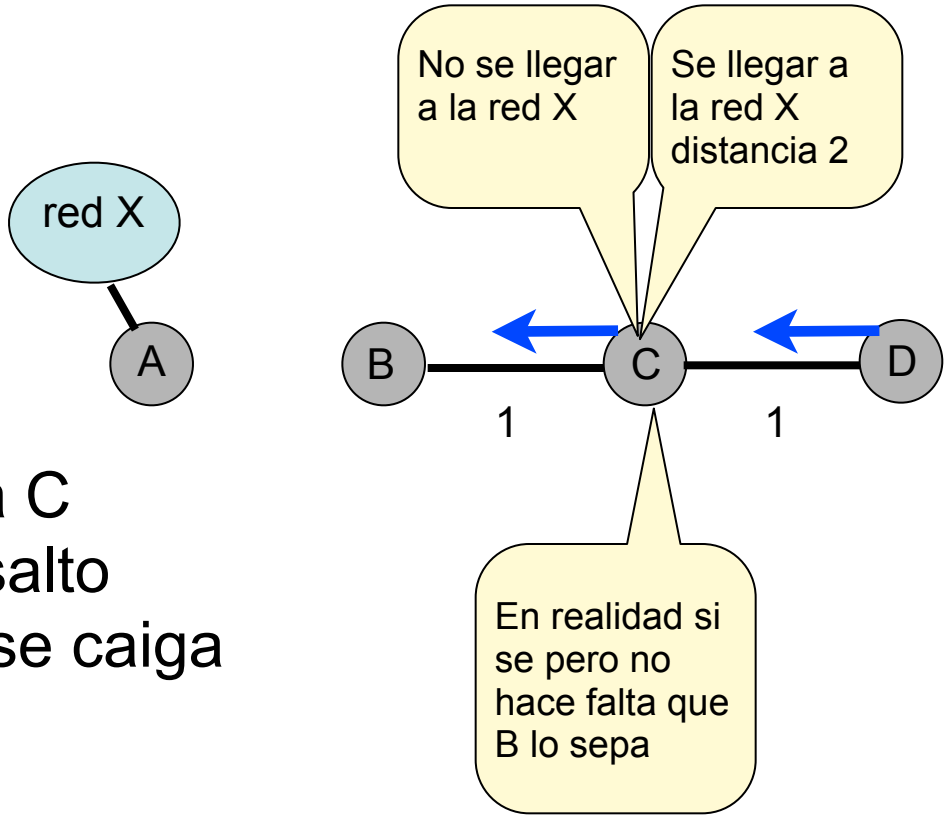
- Se puede resolver el problema de las cuentas a infinito?

  Hay algunas optimizaciónes que parecen obvias...
  - Mejor no anunciar una ruta a un nodo si la ruta pasa por el (split-horizon)
  - Cuando una ruta se vuelve es descartada no aceptar nuevas (hold-down)
  - ...

Soluciones: split horizon

- No anuncio las rutas a un destino a mi siguiente salto para ese destino

  El deberia saber como llegar porque yo le estoy mandando a el los paquetes que van a ese destino
- Esto resuelve el escenario anterior

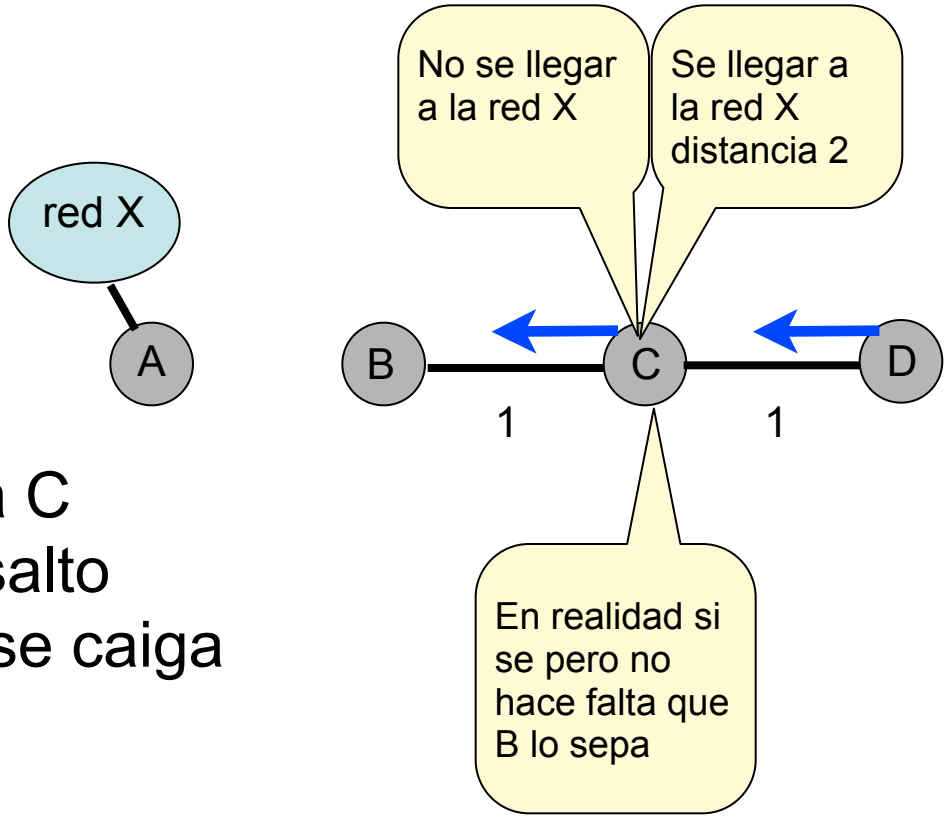


B ya no elegira a C como siguiente salto hacia X cuando se caiga el enlace

Soluciones: split horizon

- No anuncio las rutas a un destino a mi siguiente salto para ese destino

  El deberia saber como llegar porque yo le estoy mandando a el los paquetes que van a ese destino
- Esto resuelve el escenario anterior



B ya no elegira a C como siguiente salto hacia X cuando se caiga el enlace

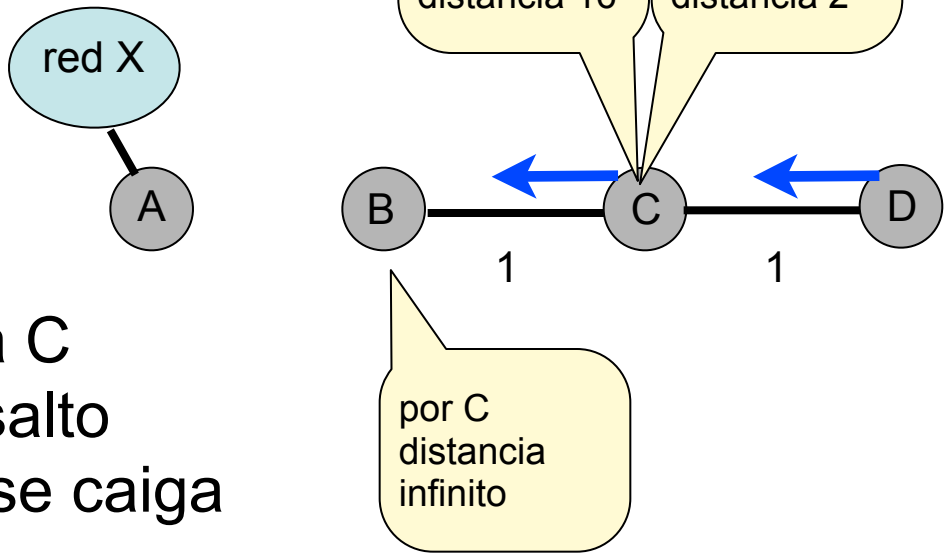
Soluciones: poison reverse

- Al siguiente salto para un destino le miento y le digo que la distancia es infinito

  Parecido a Split-horizon

  Un poco mejor porque si por alguna otra cosa habia un ciclo de enrutamiento lo rompe

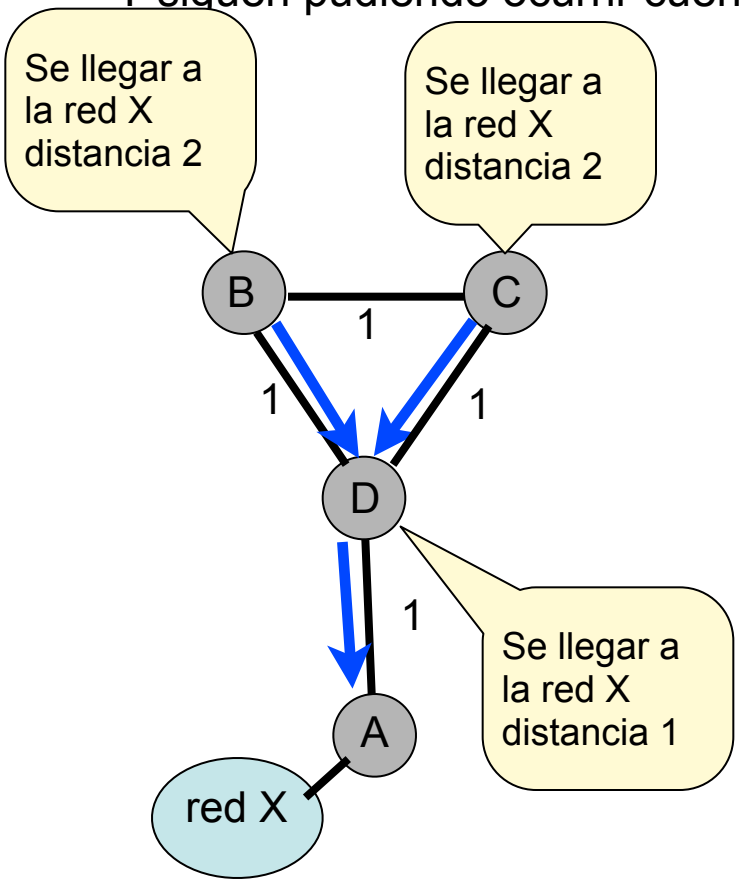
- Tambien resuelve el escenario anterior

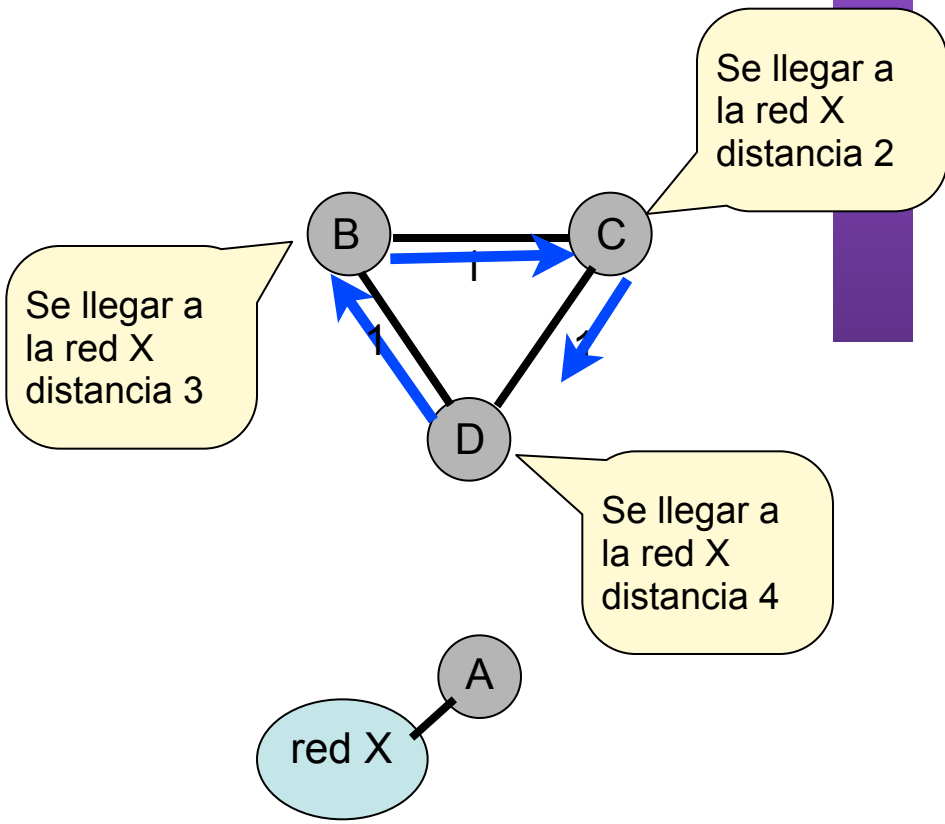


B ya no elegira a C como siguiente salto hacia X cuando se caiga el enlace

Split-horizon/poison-reverse

- Fáciles de implementar y resuelven el escenario anterior
- Tienen problemas en redes de area local cuando envio anuncios a broadcast como enveneno las rutas a mi siguiente salto pero no a los demas
- Y siguen pudiendo ocurrir cuentas a infinito de otros tipos





Más soluciones

- Hold-down: si una ruta se hace invalida no aceptar nuevos caminos a ese destino en un tiempo de hold-down

  Confia en que en ese tiempo la no alcanzabilidad se extienda a toda la red. si no da tiempo y algún nodo tiene aun ruta al destino puede crearse una cuenta a infinito
- Anunciar la distancia y el siguiente salto ~parecido a poison reverse
- Anunciar la distancia y el camino entero

  Este funciona bien pero requiere que los routers almacenen todo el camino para cada destino. No escala bien
- Usar dos métricas y mantener dos distancias

  Una para comparar caminos y la otra con el numero de saltos para las cuentas a infinito
- DUAL (Diffusing Update Algorithm)

  razonamientos con las distancias para decidir si es posible o no que el camino anunciado pase por ti

  i.e. si la distancia que anuncia C es menor que la que yo tenia al destino antes de hacerse invalida la ruta es seguro cambiar a C

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?

Enrutamiento Link-State

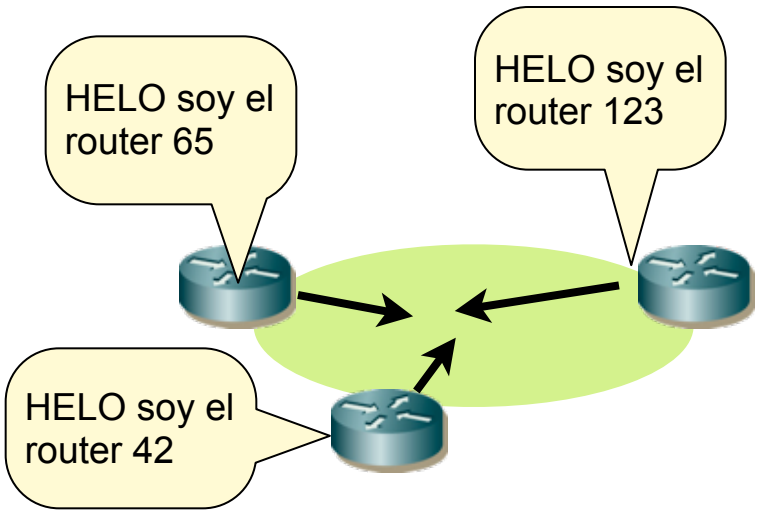
- Idea de funcionamiento
  - Cada router es responsable de reconocer a sus vecinos y aprenderse su nombre (identidad)
  - Cada router construye un paquete llamado LSP (link state packet) conteniendo una lista de todos sus vecinos y el coste asociado
  - El LSP se envía a todos los demás routers de la red. Cada router recuerda el ultimo LSP recibido de cada otro router
  - Cada router con el mapa completo de la topologia calcula las rutas a cada destino posible y actualiza su tabla de rutas

- Como de dificil es cada una de estas fases?

Link-State: 1 descubrir vecinos

- Descubrimiento de vecinos

  Mensaje HELO
  - Contiene una identidad del router
  - Periódico a la red de área local o al enlace punto a punto
  - Recordar los últimos HELOs vistos (lista de identidades de los vecinos)
  - Tiempo entre mensajes
  - Tiempo para dar al router por desaparecido



Vecinos	Visto hace
65	6s
42	14s

Link-State: construir LSPs

- Construir LSPs
  - Periódicamente
  - Cada vez que veo un nuevo vecino
  - Cada vez que cambia el coste a un vecino
  - Cada vez que dejo de ver a un vecino (ha pasado el tiempo definido sin escuchar HELOs de ese vecino
- El LSP contiene

  El router de identidad X tiene esta lista de vecinos

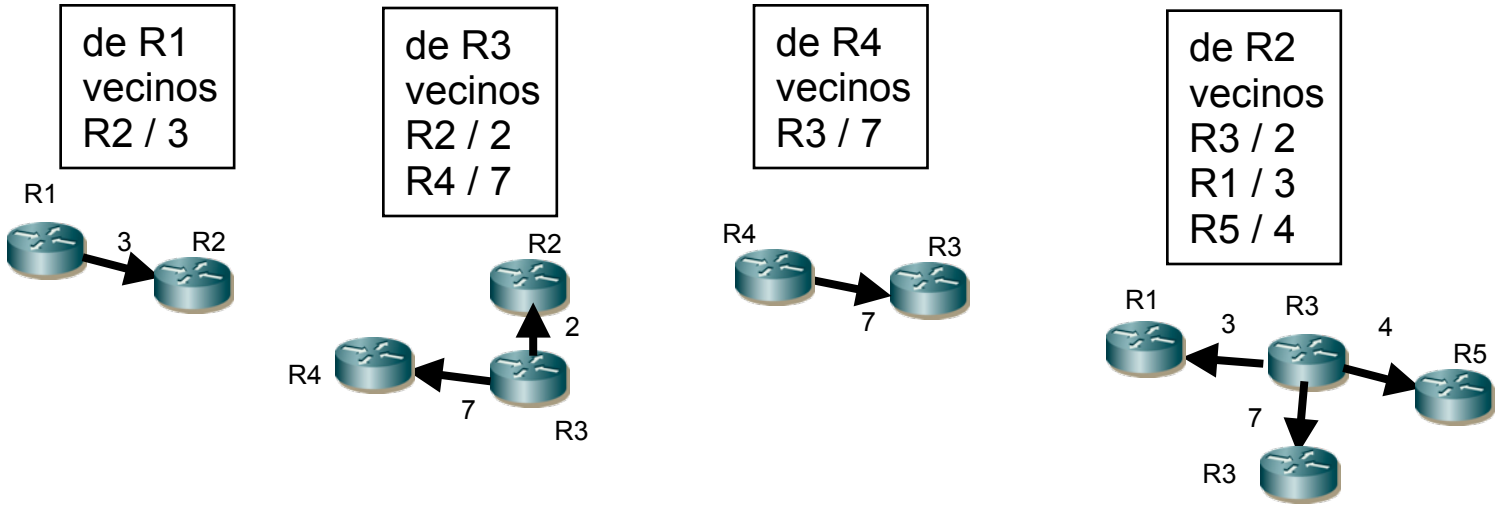
  Vecino 1: Y con coste c1

  Vecino 2: Z con coste c2

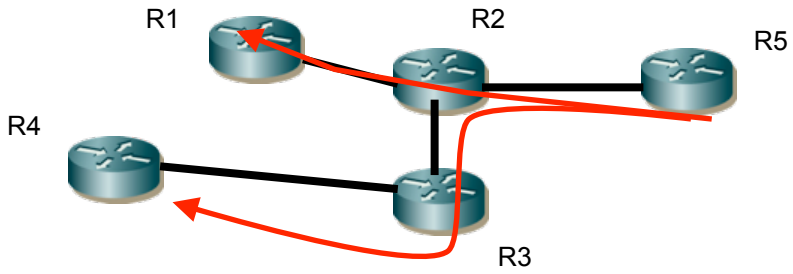
  Vecino 3 ..
- En que se diferencia de los distance-vector??

Link-State: 3 distribuir LSPs

- Cada router necesita los LSPs de todos los demás, no solo el de sus vecinos
- Recibo estos



- Base de datos de nodos y enlaces en la red
- Usando el algoritmo de Dijkstra calculo caminos



Link-State: 3 distribuir LSPs

- Si hay un cambio en la red los routers que lo ven envían el cambio a todos
- El resto de los enlaces están en la base de datos
- Con el cambio de ese enlace se recalculan los caminos en todos
- No hay cuentas a infinito
- Reaccion muy rapida

Link-State: 3 distribuir LSPs

- El destino de los LSPs

  Son todos los routers de la red

  Es facil enviar a los vecinos pero no a cualquier router de la red... (no se las direcciones de todos, no puedo basarme en la tabla de rutas para llegar a ellos)

- Es necesario un sistema de envío a cualquier destino que no requiera que funcionen las tablas de rutas.

  Inundación?

- La distribución de LSPs es el punto critico de los algoritmos de tipo link-state
  - Si no llega la información a todos, las rutas son incoherentes y puede haber ciclos de enrutamiento (recordar cuentas a infinito con un solo enlace en desacuerdo)
  - Si no tenemos cuidado la inundación puede causar mucho trafico extra, los routers pueden saturarse y dedicar gran parte de su cpu simplemente a recalcular rutas tras recibir

Link-State: 3 distribuir LSPs

- Inundando LSPs
- Cada router escucha LSPs de sus vecinos y reenvía
- Como limitar el tráfico
  - TTLs?
  - Con LSPs ya estamos guardando estado (cada router recuerda el ultimo LSP de cada destino) usar esto para limitar la inundacion (ignorar un LSP si ya lo he visto)

Link-State: 4 calcular rutas

- Los LSPs se guardan en una link-state database
- Se utiliza el algoritmo de Dijkstra para calcular caminos
- Se colocan las rutas de este nodo teniendo en cuenta esos caminos

- Si la red es muy grande todos los routers necesitan mas recursos que en el caso de distance-vector
  - Deben almacenar todos los enlaces de la red

    (mucha mas memoria que el vector de distancias)
  - Deben calcular el algoritmo de Dijkstra cada vez que ven un cambio (mucho mas complicado que la iteracion del Bellman-Ford)

- La convergencia es rapida porque los LSPs llegan a todos y no hay ambiguedades de caminos ni cuentas a infinito.

Distance-vector vs Link-state

- En uso de memoria: menor distance-vector
- En tráfico de red: controvertido distance-vector no hace inundacion pero la inestabilidad dura mas
- En carga de CPU: menor distance-vector
- En robustez: no esta claro. link-state es mas robusto frente a fallos no maliciosos
- En funcionalidad: link-state tiene muchas ventajas, ya que todos los routers conocen la topologia entera, es mas facil de depurar las situaciones de fallo, permite source-routing, caminos paralelos...
- Velocidad de convergencia: calramente mejor link-state

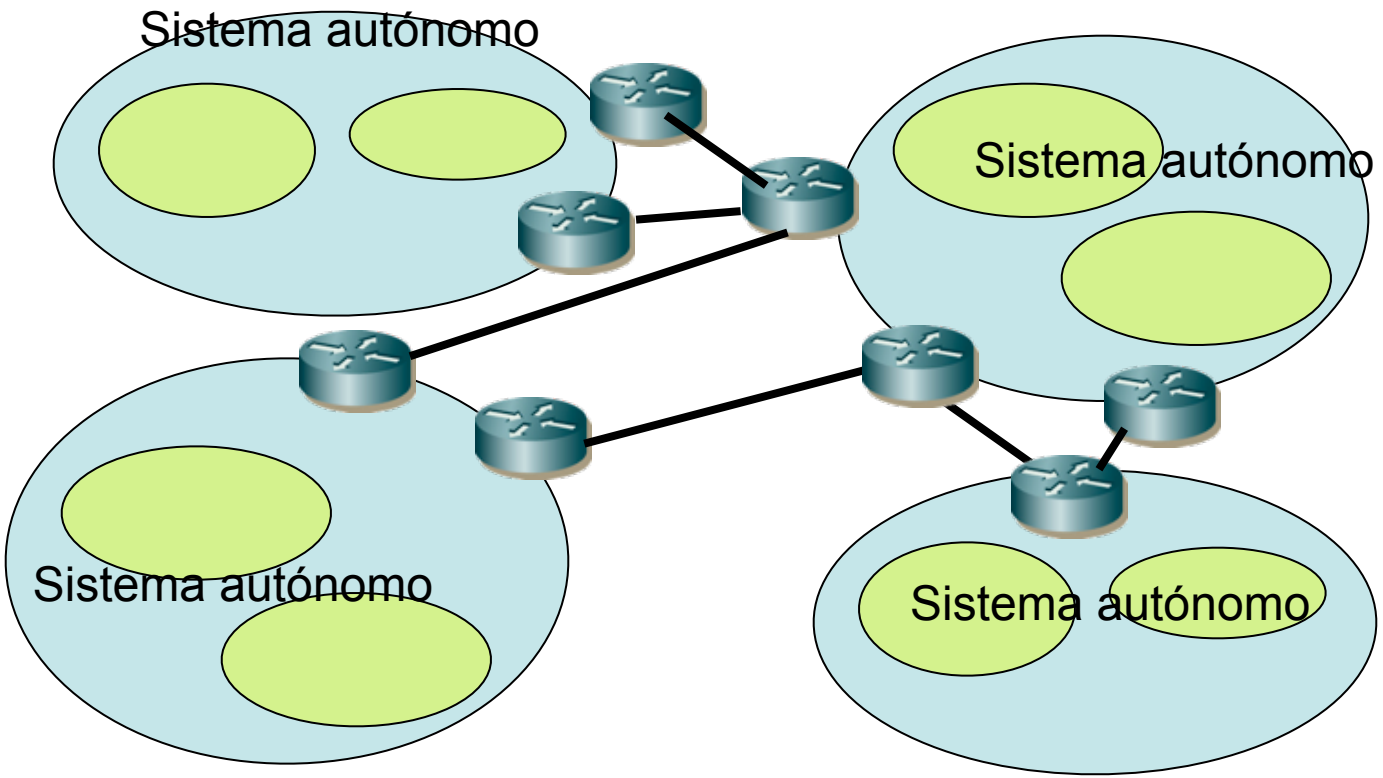
Link-state en Internet

- OSPF (Open Shortest Path First) (RFC-2328)
  - Envío del estado de los enlaces entre routers
  - Mapa de la topología en cada router
  - Base de datos de estado de enlaces
  - Cálculo de las rutas con el algoritmo de Dijkstra en cada router

- Protocolo para enviar información sobre el estado de los enlaces
  - Link-state advertisements en la terminologia de OSPF se llaman LSAs
  - Inundación (flooding) a todos los routers (del sistema usando OSPF)
  - Mensajes OSPF directamente sobre IP (no TCP ni UDP) protocol=89
  - Enrutamiento jerárquico para simplificar (areas)

Intradomain vs Interdomain

- Internet es un conjunto de sistemas autonomos con enrutamiento independiente
- Cada uno puede utilizar los protocolos de enrutamiento que elija internamente **intra-dominio**
- Hay protocolos para compartir las rutas entre dominios (**inter-dominio**)
  - Un protocolo común para hablar con cualquier otro sistema autonomo
  - Actualmente se usa BGP4



Interdomain routing

- Solo puede haber uno

  un protocolo unico BGP4
- No puede basarse en rutas por defecto y exterior

  No hay exterior. Se le llama tambien default-free-zone
- Protocolo de intercambio de rutas entre sistemas autonomos vecinos
  - Los AS son los nodos del grafo
  - El protocolo debe soportar cuestiones politicas aparte de pesos. (i.e. el trafico que va de un destino a otro puede ir por este AS y no por este otro porque uno es cliente mio y no puedo usarlo como transporte)
  - BGP4 es de tipo PATH-Vector (Bellman-Ford pero anuncio el camino completo no solo el peso)

Conclusiones

- Protocolos de enrutamiento link-state
  - Mas complejos
  - pero en general mejores que distance-vector
- Protocolos link-state reales: OSPF (+IS-IS +otros)
- Usan enrutamiento jerarquico
- Enrutamiento intradominio e interdominio