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

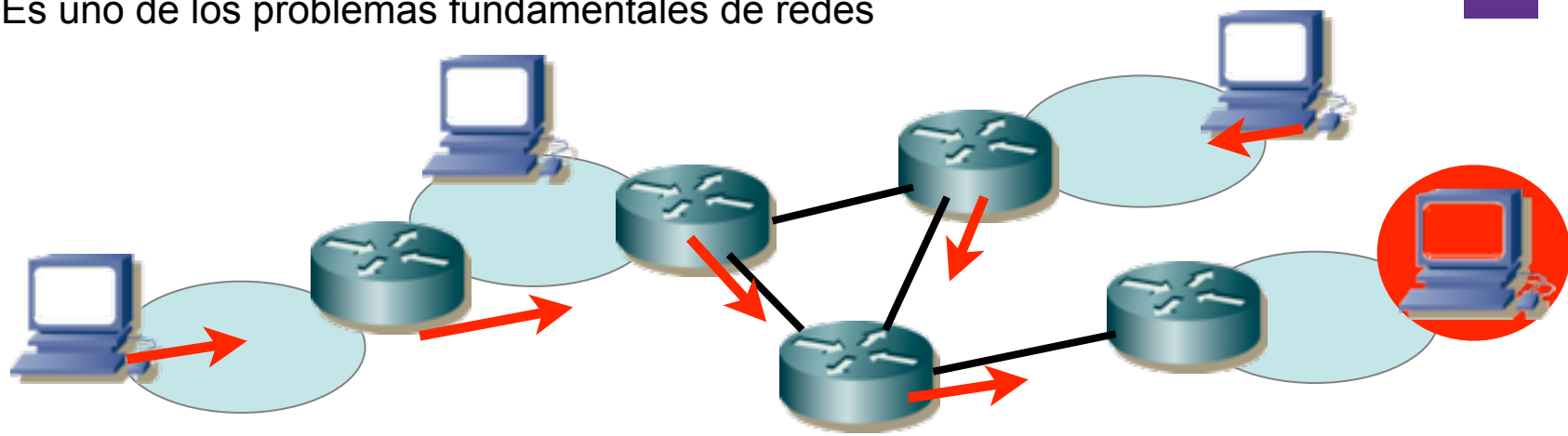
Temario

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**

El problema del encaminamiento

- El nivel de red reenvía paquetes hacia su destino
- Usando el nivel de enlace que permite enviar a los vecinos
  - Al otro lado de un enlace
  - Vecinos en la misma red de área local
- Nivel de red : reglas de reenvío basadas en el destino del datagrama
  - Si el destino es vecino enviar usando el nivel de enlace
  - Si el destino no es vecino decidir el mejor vecino al que delegar y enviarle el datagrama para que el se encargue
- **Encaminamiento / enrutamiento / routing**

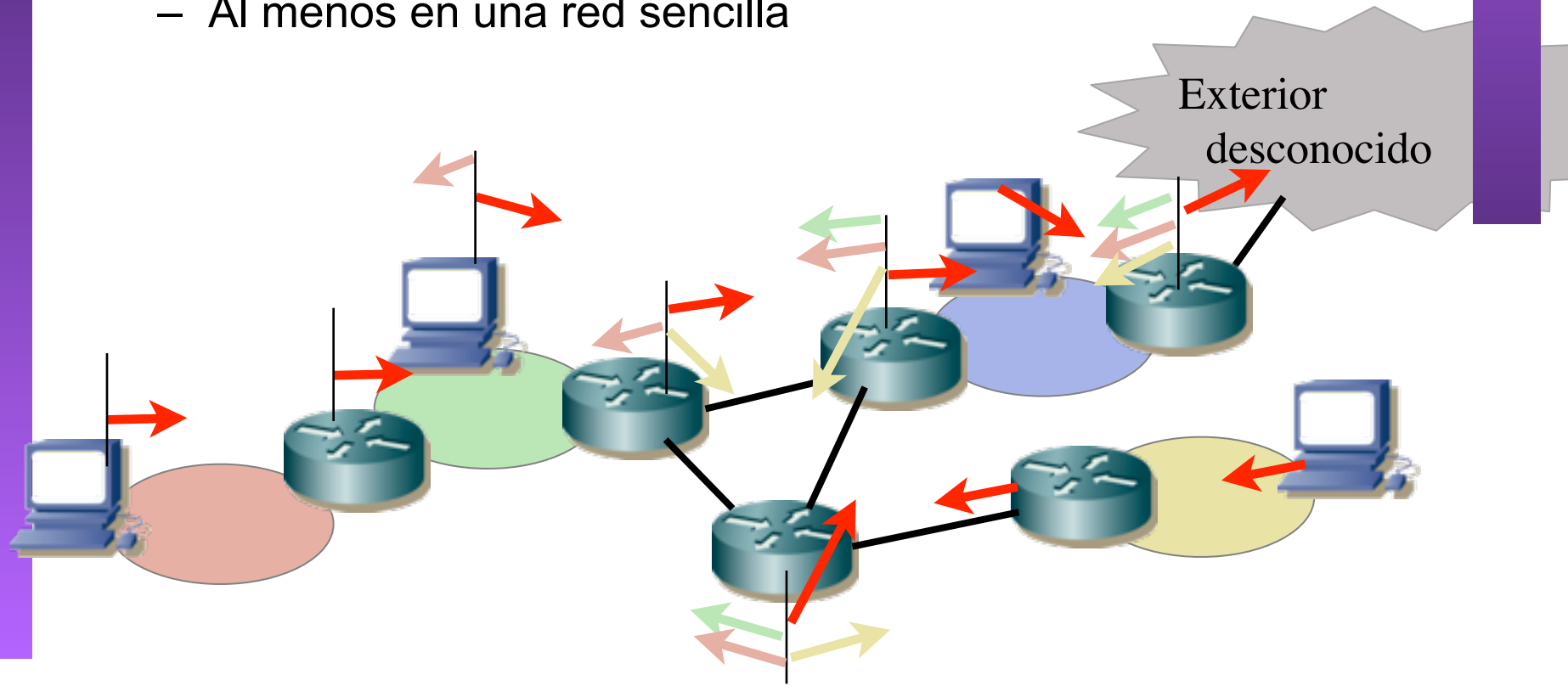
  Cómo decidir que caminos poner en las tablas de reenvio (tabla de rutas)
- Es uno de los problemas fundamentales de redes



Enrutamiento estático

- Solución más fácil: let the human do it

  ¿En esta red cual deberían ser las tablas de rutas?
- Parece fácil de pensar con un poco de paciencia
  - Se pueden resumir redes
  - Se pueden usar rutas por defecto
  - Al menos en una red sencilla



Enrutamiento estático: problemas

- No es tan fácil cuando no hay exterior
- No es tan fácil cuando el numero de nodos crece
- No es tan fácil cuando hay caminos alternativos y ciclos
- Los administradores se equivocan
- Los administradores pertenecen a diferentes empresas que no confían entre si
- ¿Qué hacemos para poner un nuevo router, una nueva red, añadir enlaces?
- ¿Se puede hacer que el encaminamiento funcione de forma automática?

- **Encaminamiento dinámico**

Enrutamiento dinámico

- En redes de datagramas
  - Cada paquete es reenviado según la información de enrutamiento que hay en cada nodo en cada momento.
  - Si cambian las condiciones de la red entre un paquete y otro los paquetes pueden seguir distintos caminos
  - Robusto, se adapta rápido a los cambios
- En redes de circuitos/circuitos virtuales
  - Al establecer el circuito/camino se utiliza la información de enrutamiento disponible en ese momento
  - Una vez establecido el circuito se reenvía más rápido. Pero si cambian las condiciones de la red el camino ya no es el mejor
  - Las decisiones de enrutamiento son menos frecuentes (=podemos la misma capacidad en usar algoritmos mas complicados para decidir el camino)

¿Cuál es el destino?

- **Unicast**: un solo destino
- **Broadcast**: el destino son todos los nodos
- **Multicast**: el destino es un conjunto de nodos, un mensaje tiene que llegar a todo el conjunto
- **Anycast**: el destino es un conjunto de nodos, un mensaje tiene que llegar a uno cualquiera del conjunto

- Nos centraremos en unicast
- Las herramientas/algoritmos son también la base del enrutamiento broad-, multi-, any- cast

Enrutamiento dinámico

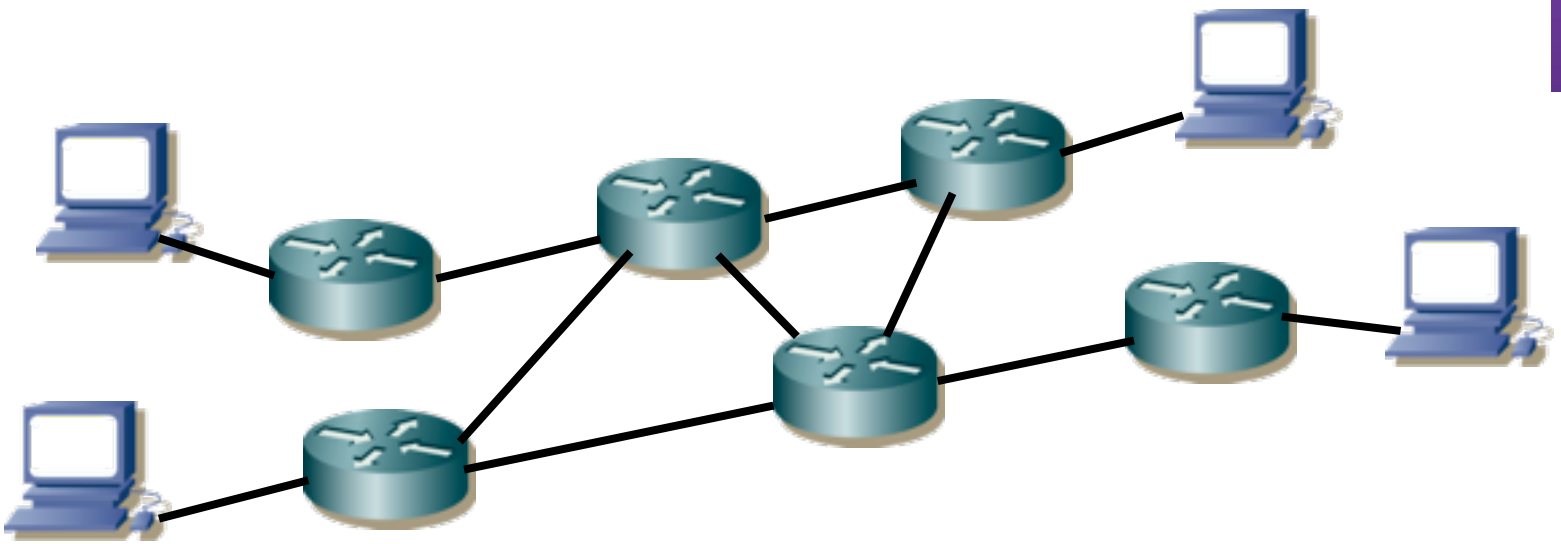
- ¿Qué información necesitamos de la red?
  - Un mapa de la red entera?
  - Se puede hacer algo con información parcial?
  - Se puede hacer algo sin información?

- Para programar y mantener lo mejor es lo mas simple...

Empezando desde cero

- ¿Hay algún algoritmo simple que permita que los paquetes lleguen a su destino, sin mucha complicación?
- En realidad si
- Y con información mínima en cada router

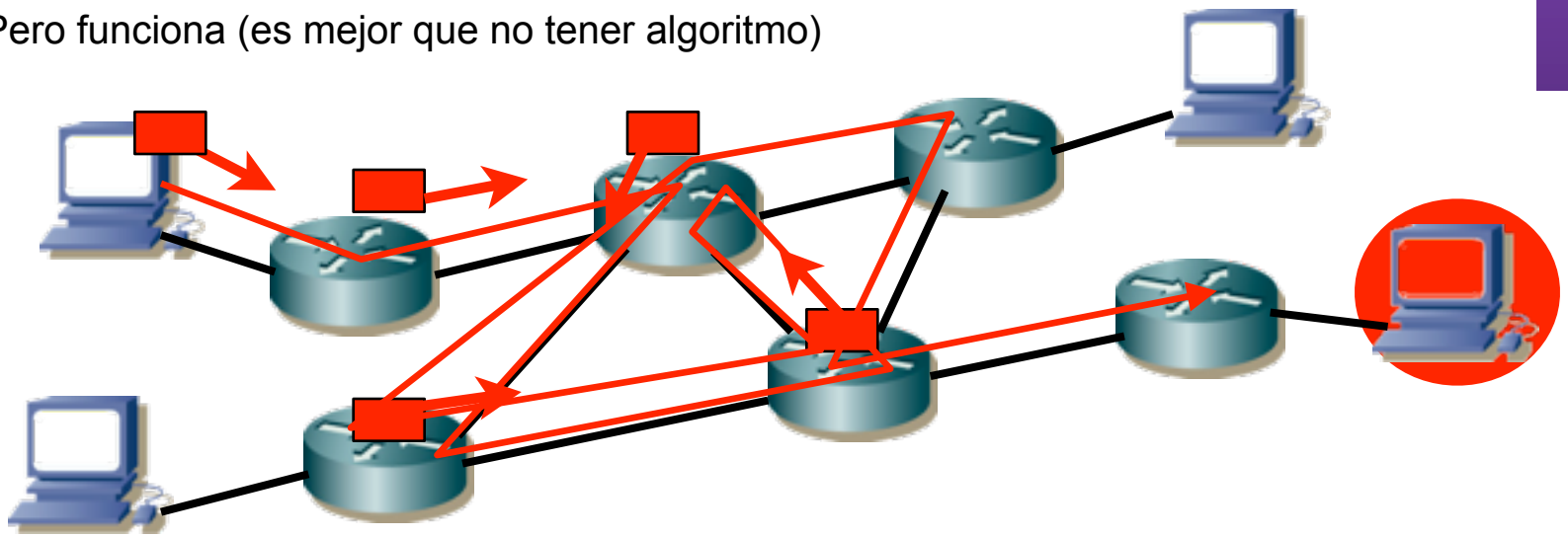
  (sin tabla de rutas)



Enrutamiento aleatorio !!

- Cuando tengo que reenviar un paquete que no es para mi (podria extenderse a ni para mis redes conectadas)...

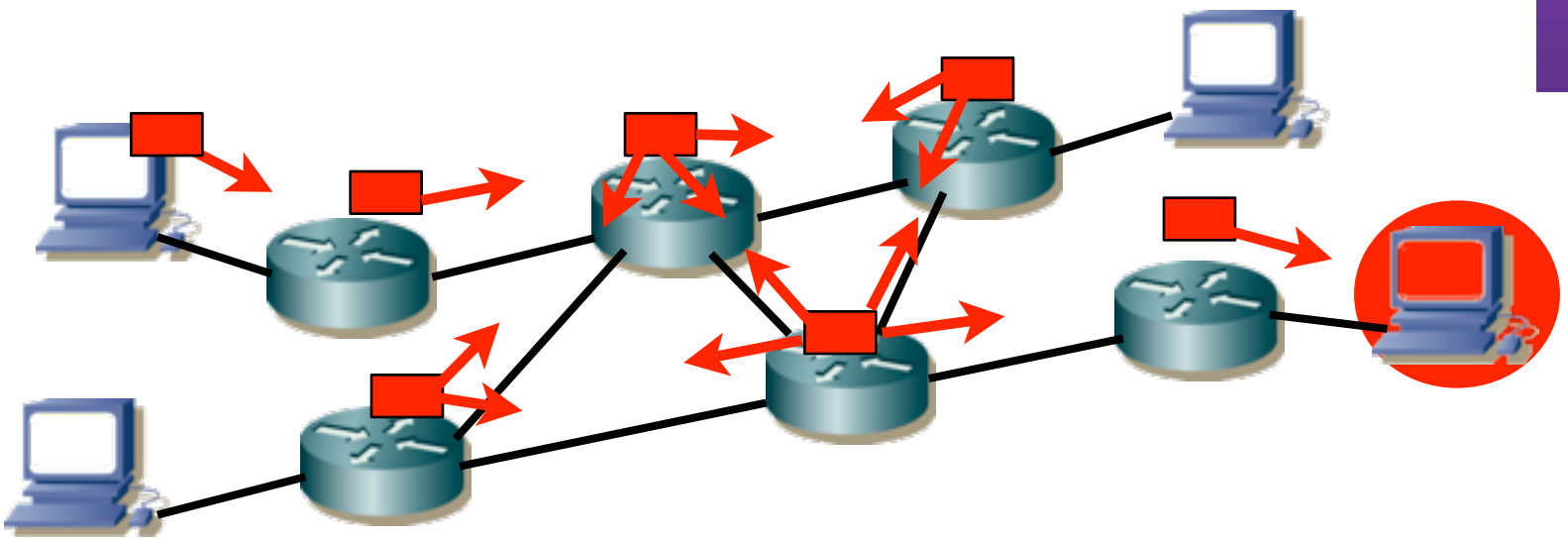
  elige un **siguiente salto aleatorio** entre todos los posibles y envíaselo a el
- No es muy eficiente
- El paquete puede tardar mucho en llegar pero al final llega (con TTL infinito) o bien podemos jugar con TTLs y probabilidades de entrega
- Puede pasar mas de una vez por los nodos
- Normalmente no llega por el camino mas corto
- Ni todos los paquetes que llegan van por el mismo camino =no mantiene el orden de paquetes
- Pero funciona (es mejor que no tener algoritmo)



Enrutamiento por inundación

- Cuando tengo que reenviar un paquete que no es para mi (podria extenderse a ni para mis redes conectadas)...

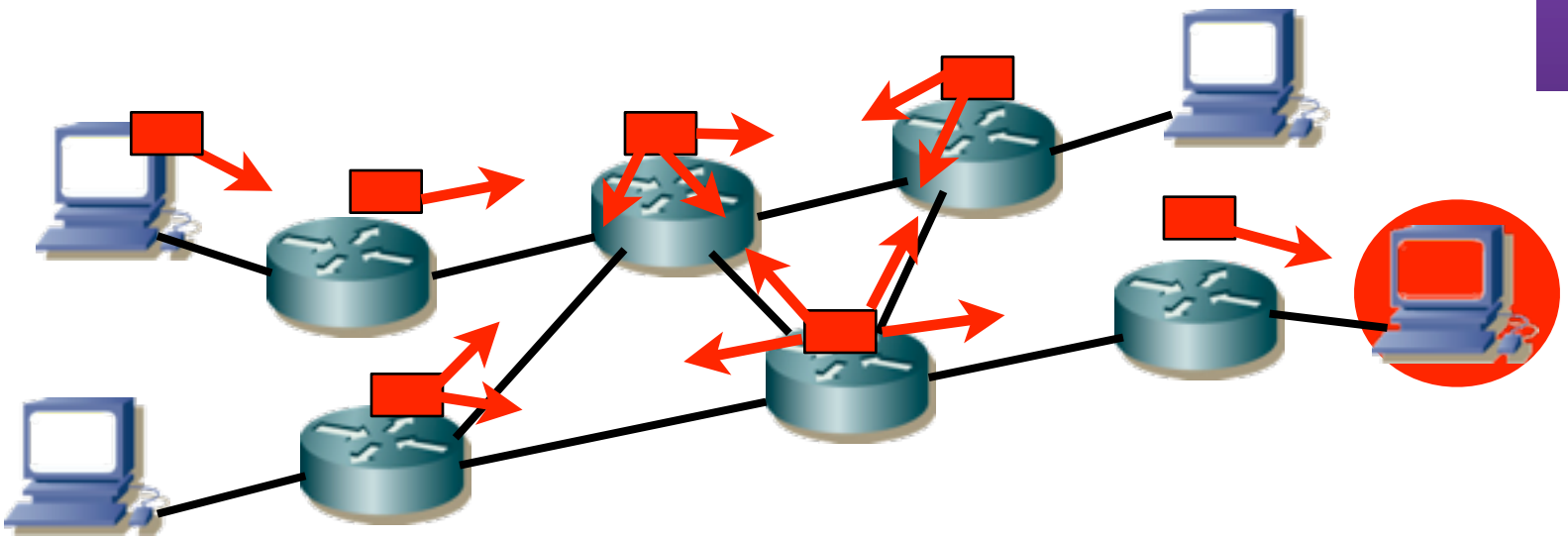
  envíalo a todos los siguientes saltos menos por el que llego
- Genera tráfico extra. Cuanto?
- El paquete llega seguro (salvo que se usen TTLs)
- El paquete llega por el camino mas corto
- El paquete llega probablemente mas de una vez. Hay que usar técnicas para garantizar que solo se entrega una vez (identidad del paquete)



Enrutamiento por inundación

- **¿Podemos resolver todos los problemas?**
- El paquete llega probablemente mas de una vez. No queremos entregar duplicados
  - Campo de Identidad en el paquete. Hay que garantizar que no se use de nuevo la misma identidad en el marco de tiempo en que podría confundirse con una copia
- Si hay ciclos en el grafo la inundación genera tráfico infinito
  - TTL en los paquetes (facil, pero ya no podemos garantizar que llegue seguro)
  - No reenviar paquetes recientes

    Parece facil... pero... que es un paquete reciente? Implica guardar los paquetes que hemos reenviado en los últimos T segundos. Estado en el router



Enrutamiento sin información

- **Enrutamiento aleatorio y por inundación**
- Presentan problemas de poca eficiencia/mucho tráfico extra
- Pero son simples y funcionan
- Hay situaciones en las que tiene sentido usarlos
- Incluso hoy en dia
- A veces combinados con otros sistemas de enrutamiento
  - Por ejemplo la inundación se puede controlar más facil si estamos en un entorno en que hay otro protocolo de enrutamiento funcionando

    Esto se usa para conseguir broadcast y multicast
  - Por ejemplo la inundación puede ser aceptable si solo se usa al establecer un circuito porque garantiza encontrar el camino mas corto.

- Pero en general en una red tan compleja y de la extensión de Internet los protocolos de este tipo no escalan.
  - Mejor mantener los paquetes a que vayan solo por un camino hacia el destino

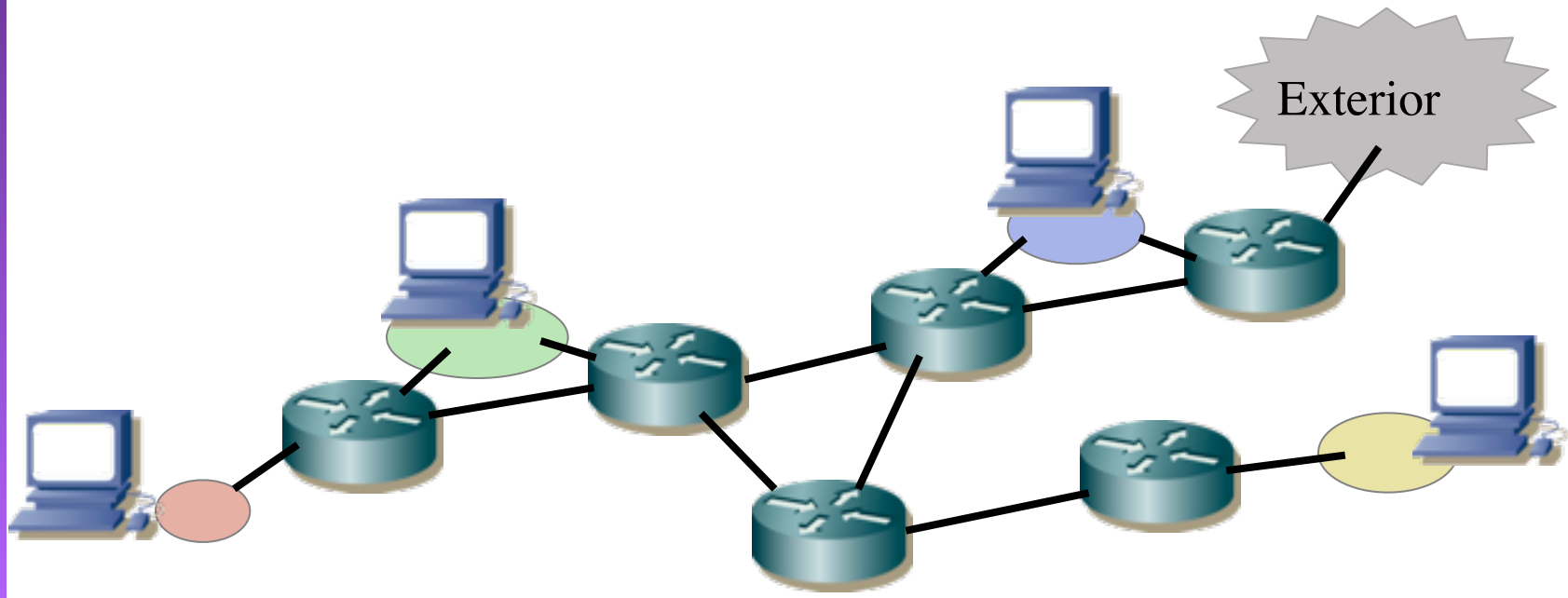
    Mejor si es el mas corto
  - Para eso hace falta utilizar información en el enrutamiento

Enrutamiento con información de la red

- Simplificaremos el problema pensando que las redes son una serie de nodos interconectados entre si

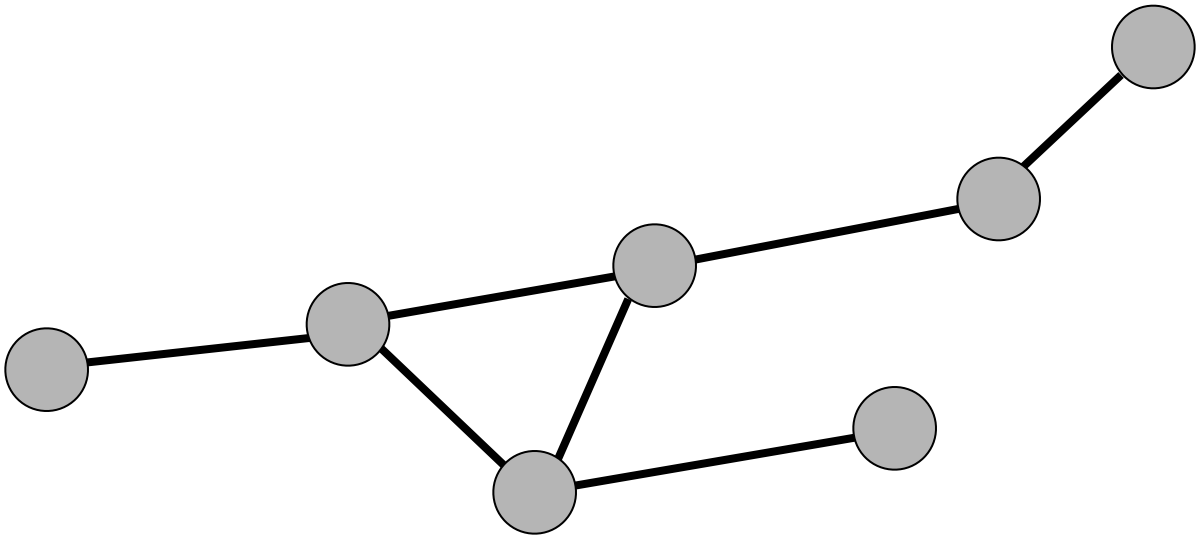
  formando un grafo. Las redes de area local se pueden pensar simplement como destinos

  Los detalles de como se distinguen redes y routers quedan para los protocolos de enrutamiento concretos en proximas clases



Enrutamiento con información de la red

- Tenemos un grado en el que los elementos son reenviadores de paquetes: nodos
- Los enlaces son conexiones entre estos (punto a punto, LAN...)
- Los nodos son posibles origenes y destinos
- ...



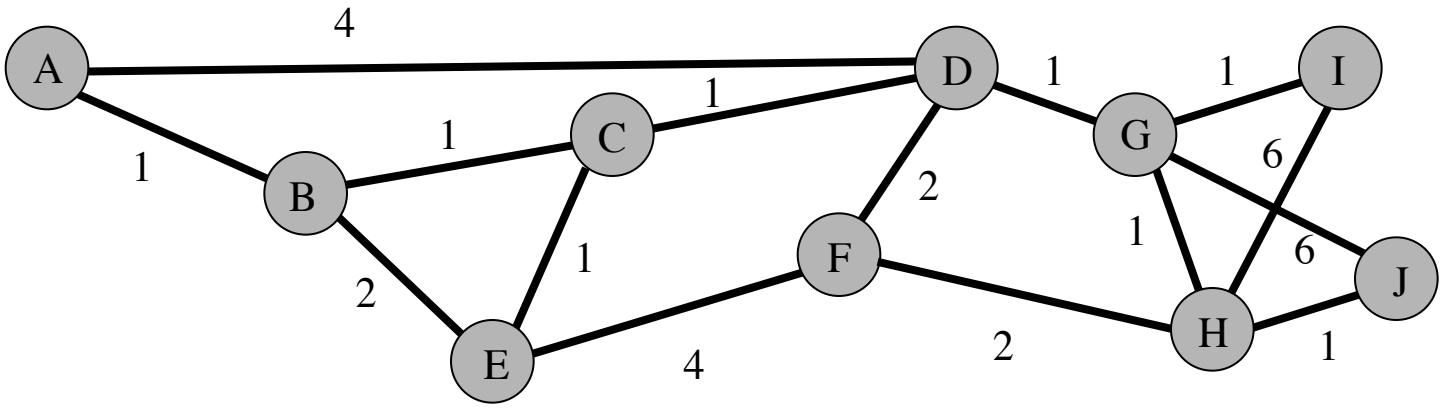
Enrutamiento con información de la red

- Los enlaces pueden ser iguales o no
- Normalmente asignamos un peso a cada enlace que mide cuanto nos cuesta o queremos evitar ese enlace comparado con otro

  Se supone que el peso es aditivo

  Preferimos A-B-C-D (1+1+1) que A-D (4)
- Si asignamos pesos 1 lo que nos interesa es el numero de saltos del camino total
- Por simplicidad suponemos un grafo no dirigido

  (= los enlaces son bidireccionales)



Enrutamiento con información de la red

- La red puede cambiar
  - Los enlaces pueden crearse y destruirse
  - Los pesos de los enlaces pueden cambiar (pueden depender de la carga que este atravesando el enlace por ejemplo)
- El enrutamiento dinámico debe adaptarse a estos cambios
- Los protocolos reales se han centrado más en adaptarse a los cambios en la topología que a los cambios de la carga.
  - La carga puede cambiar muy rápido
  - Es malo que el enrutamiento cambie demasiado rápido

    Compromiso robusto vs estable
    - Cambiar rápido: robusto, ante una caída reacciona rapido encontrando nuevos caminos
    - En los periodos de cambio pueden generarse situaciones anómalas, bucles de enrutamiento
    - Resistencia a cambiar: sistema más estable y condiciones mas predecibles

Clasificación

- Distribuido vs Centralizado
- Centralizado
  - Un ente recopila toda la información de la red y decide las rutas
  - Este ente puede tomar decisiones conociendo toda la red
- Distribuido
  - Cada nodo se comunica con el resto de nodos y utiliza la información que obtiene para calcular su tabla de rutas
  - Los nodos pueden calcular la tabla de rutas con información parcial o conseguir toda la información

Clasificación

- Proactivo vs reactivo
- Enrutamiento **proactivo**
  - Los protocolos de enrutamiento funcionan de forma continua rellenando las tablas de rutas para todos los caminos y cambiandolas cada vez que hay un cambio en la topologia
  - Al enviar un paquete a un destino ya existe la entrada en la tabla
  - Este es el caso normal en Internet
- Enrutamiento **reactivo**
  - Las tablas de rutas se calculan solo para los destinos que se están usando
  - El calculo se dispara cuando hace falta enviar a un destino
  - Asociado a redes de circuitos

    O bien a redes donde se intenta evitar enviar información (i.e. inalambricas)
  - Por ejemplo:

    red de circuitos virtuales sin tablas de rutas.

    cuando se va a establecer circuito a un destino que no es vecino

    enviamos inundación preguntando por el destino

    cada nodo que reenvía la información añade su direccion

    por lo que el paquete que recibe el destino tiene el camino

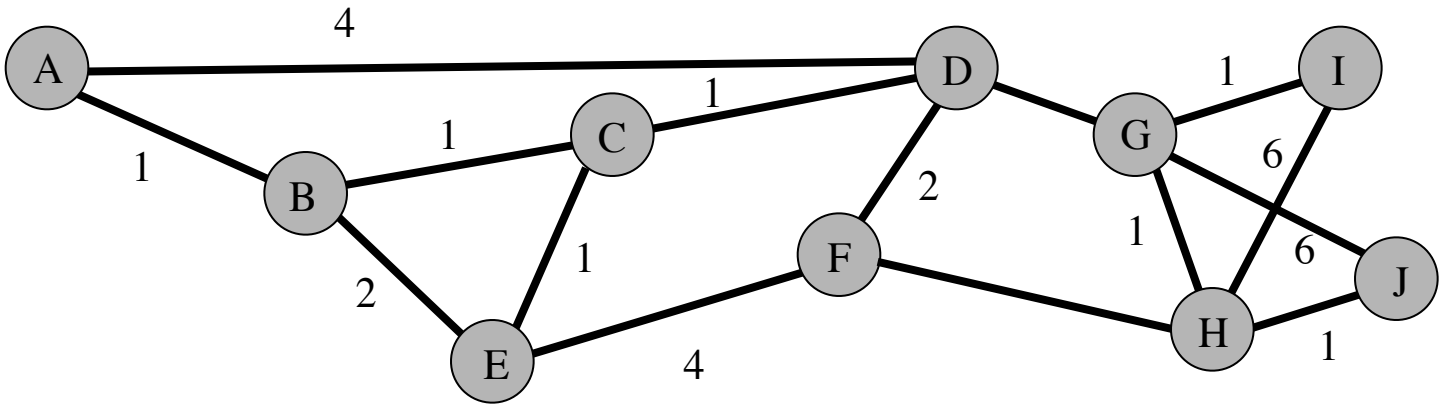
Características avanzadas

- Confianza/seguridad
- Broadcast y multicast
- Un solo camino o caminos paralelos
- Con calidad de servicio
- Usando información geográfica

El modelo hasta ahora

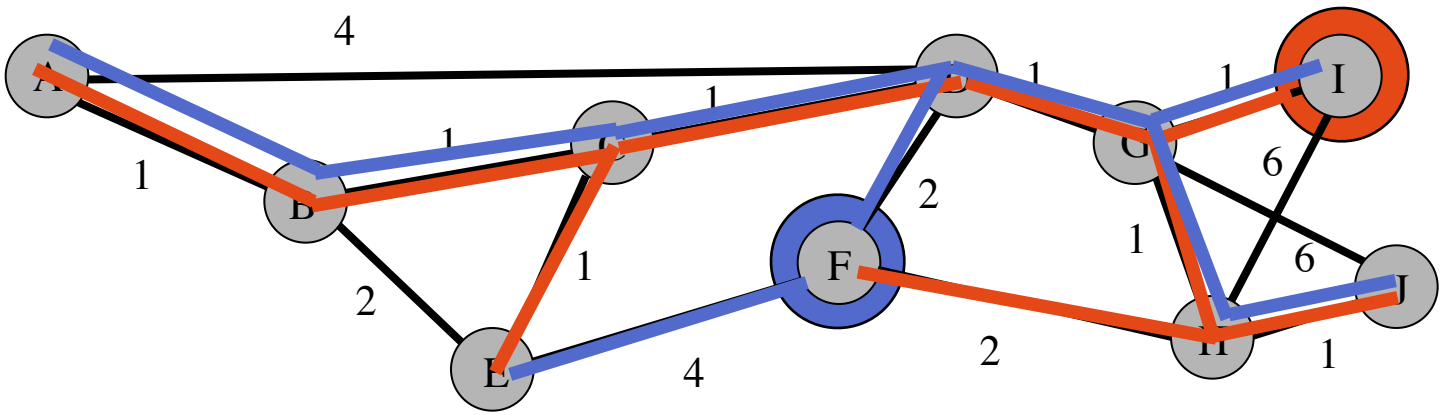
- Tenemos un grafo que representa nuestra red
- Queremos un sistema de enrutamiento...

  proactivo, adaptativo que debe buscar caminos unicast desde todos los nodos a todos los nodos para que estén calculadas todas las tablas de rutas. Se asume que la topologia no va a variar en un tiempo prudencial y los pesos de los enlaces tampoco...



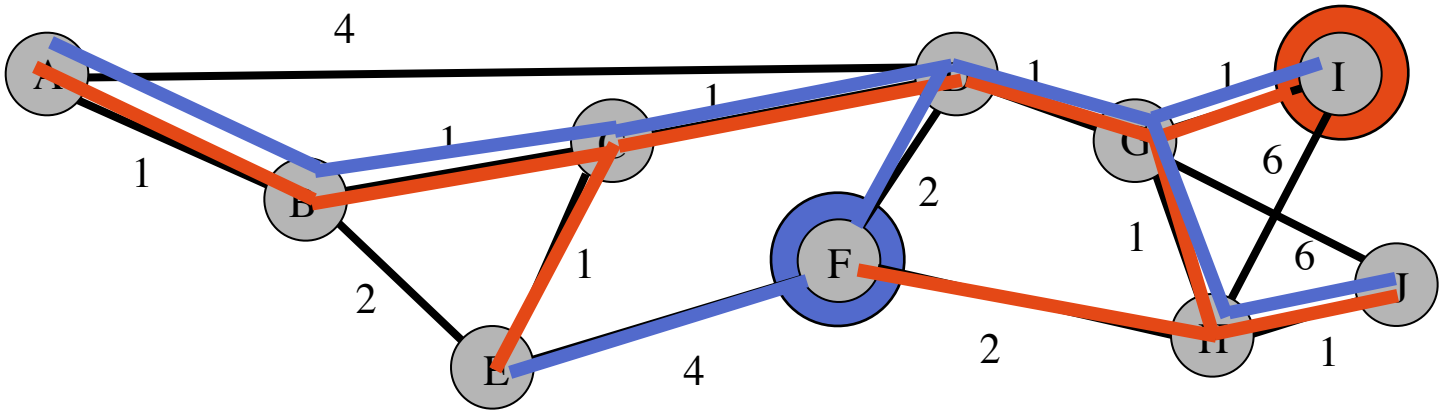
El resultado...

- Todos los caminos a un destino forman un arbol de expansión de coste mínimo del grafo
- Los caminos a cada destino son un árbol pero no tienen por que coincidir sus enlaces
- Con esto rellenaríamos la tabla de rutas de cada nodo



Adaptativo

- Como hacemos para que el enrutamiento centralizado se adapte a cambios en la red?
  - Cambio en los pesos
  - Cambio en la existencia de nodos o enlaces
- Si un ente central tiene toda la información: trivial
  - Volver a calcular los arboles con el nuevo grafo cuando se detectan cambios
  - Periodicamente obtener toda la información de la red y calcular las tablas de rutas



Distribuido

- Como hacemos para hacer el calculo de forma distribuida
  - Se puede hacer un algoritmo para que cada nodo calcule la parte del arbol que le interesa (quien es el siguiente salto) sin generar el resto del arbol
  - SI, pero no con el algoritmo de Dijkstra
  - Ademas se adapta naturalmente a los cambios de la red
  - Pero tiene algunos problemas
  - O bien podemos generar todo el arbol en cada nodo
- En cualquier caso cada nodo necesita recibir información de otros nodos... al menos de sus vecinos
- En esto se basa el enrutamiento de Internet y los protocolos clásicos
  - Algoritmos de calculo de rutas con más o menos información
  - Protocolos para comunicar información de enrutamiento a otros nodos
  - Reciben el nombre genérico de **protocolos de enrutamiento**

    i.e. RIP, OSPF, IS-IS, EGP, BGP ...

Temas avanzados

- Como hacemos para hacer el calculo de los arboles...
  - Para que puedan usarse también para multicast, broadcast, anycast...
  - De forma que sea resistente a que algunos nodos funcionen mal y envíen información incorrecta... o bien que algunos nodos quieran atacar la red y funcionen deliberadamente mal
  - En un entorno de routing reactivo (normalmente inalambrico o peer-to-peer)
  - En un entorno en el que los parametros de red/topologia cambien muy rapidamente
  - Que calculen mas de un camino al mismo destino, para garantizar redundandica, calidad de servicio...
  - ...

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]

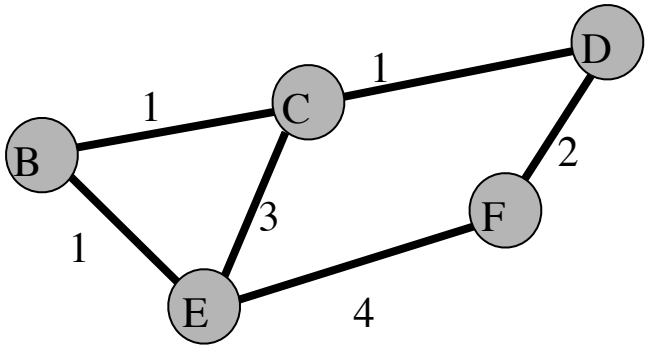
$d_h(i)=\min\{ d_{h-1}(i)+w(i,k) \}$

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

[igual que en el anterior solo que ahora consideramos que puede haber mejorado el camino de cualquiera de nuestros vecinos]

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

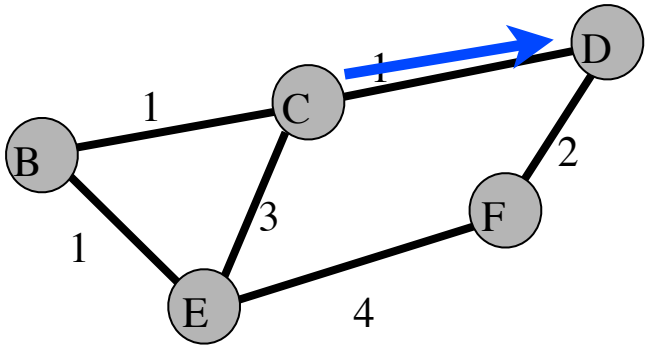
  $d_1(B)$ en la siguiente fila depende de la fila actual y las distancias

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

  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

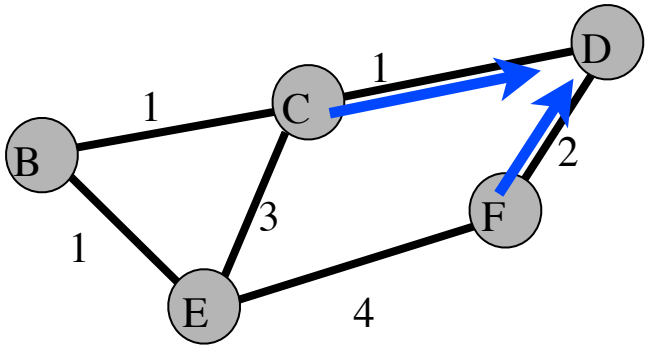
- Para el C

$d_1(C)$ 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

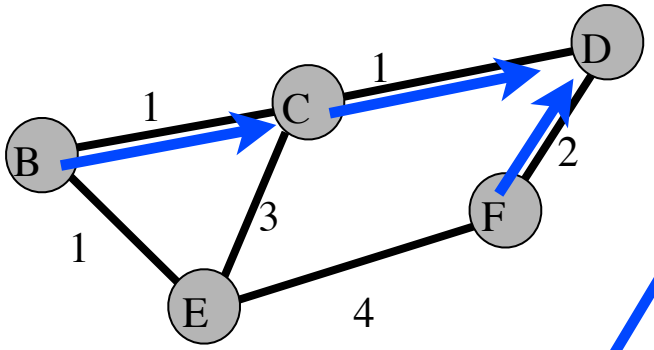
h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2					
3					
4					
5 ...					



- 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

$d_2(B)$

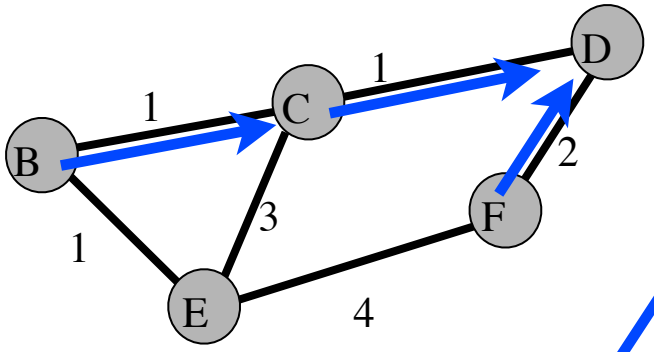
$d_1(B)+w(B,B)$ $d_1(C)+w(B,C)$ $d_1(D)+w(B,D)$ $d_1(E)+w(B,E)$ $d_1(F)+w(B,F)$

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

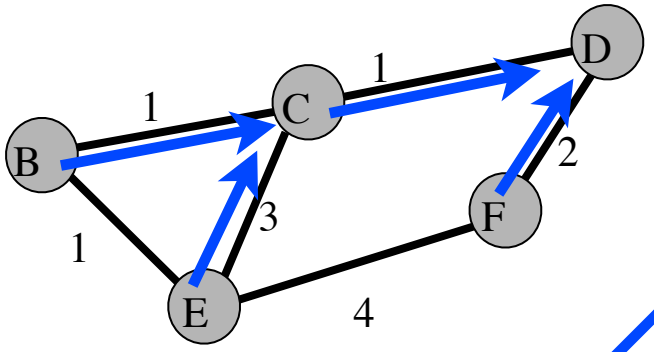
$d_2(C)$

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

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

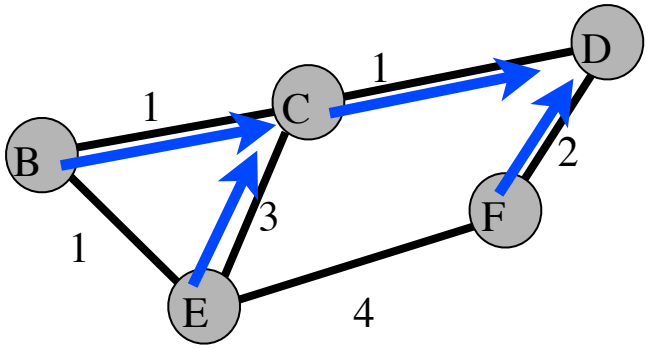
$d_1(B)+w(E,B)$ $d_1(C)+w(E,C)$ $d_1(D)+w(E,D)$ $d_1(E)+w(E,E)$ $d_1(F)+w(E,F)$

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

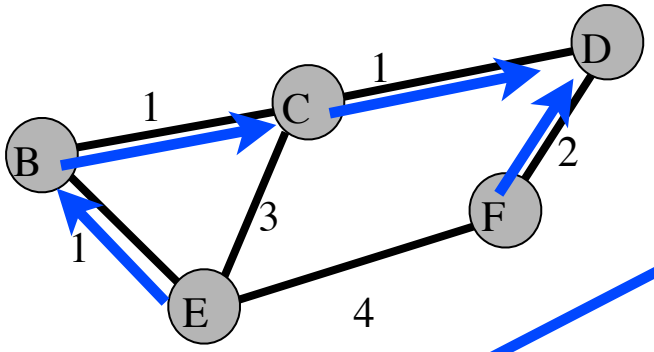
Bellman-Ford ejemplo

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

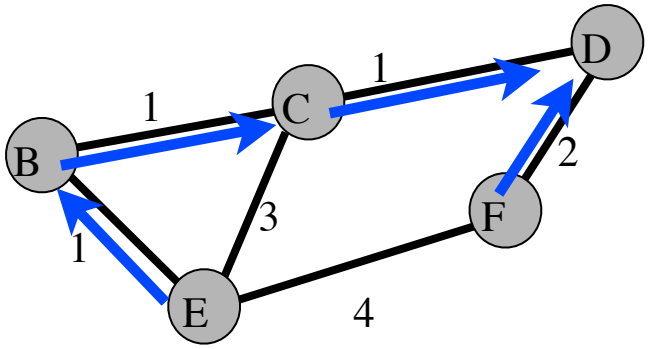
$d_3(B)+w(E,B)$	$d_3(C)+w(E,C)$	$d_3(D)+w(E,D)$	$d_3(E)+w(E,E)$	$d_3(F)+w(E,F)$
2 + 1	1 + 3	0 + inf	4 + 0	2 + 4

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

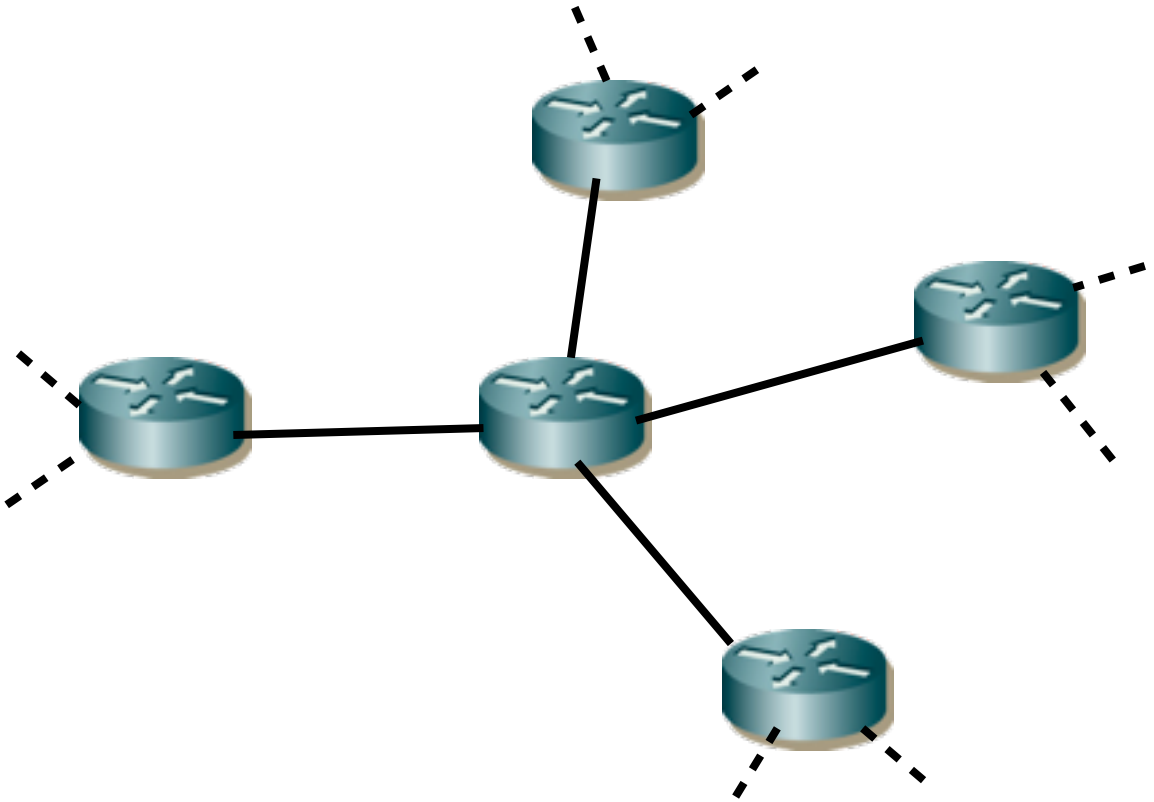
h	d(B) / s(B)	d(C) / s(C)	d(D) / s(D)	d(E) / s(E)	d(F) / s(F)
0	infinito / desc	infinito / desc	0 / soy yo	infinito / desc	infinito / desc
1	infinito / desc	1 / D	0 / soy yo	infinito / desc	2 / D
2	2 / C	1 / D	0 / soy yo	4 / C	2 / D
3	2 / C	1 / D	0 / soy yo	3 / B	2 / D
4	2 / C	1 / D	0 / soy yo	3 / B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

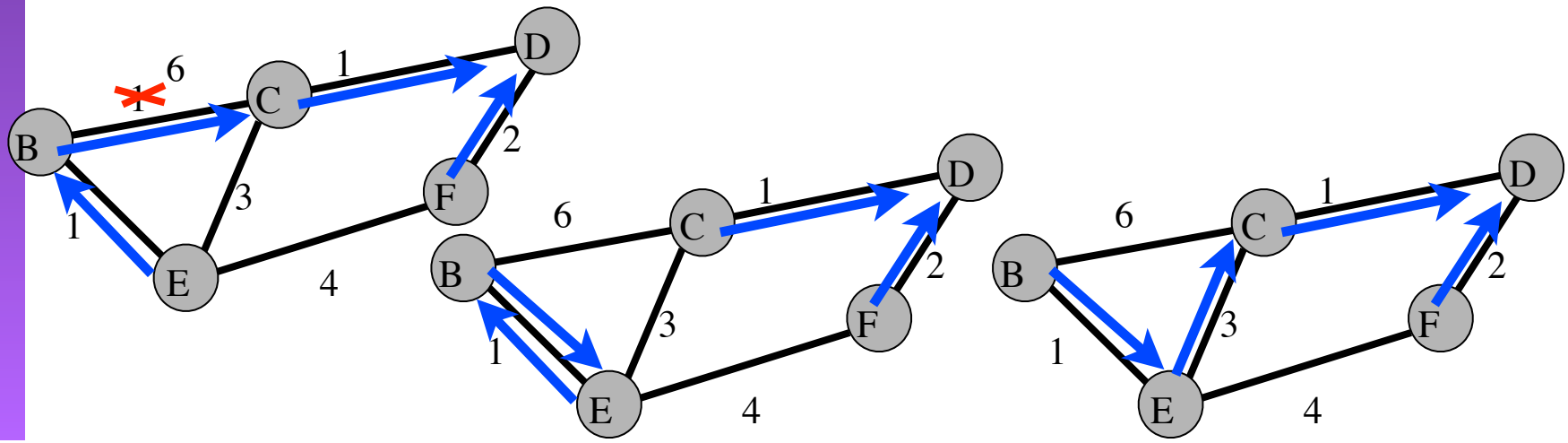
- El problema del cruce de caminos... En cada router
- Como calculo las rutas a todos los destinos?
- Solo con información mía o que pueda obtener de los vecinos
- Ni siquiera se cuales son todos los destinos !!!



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

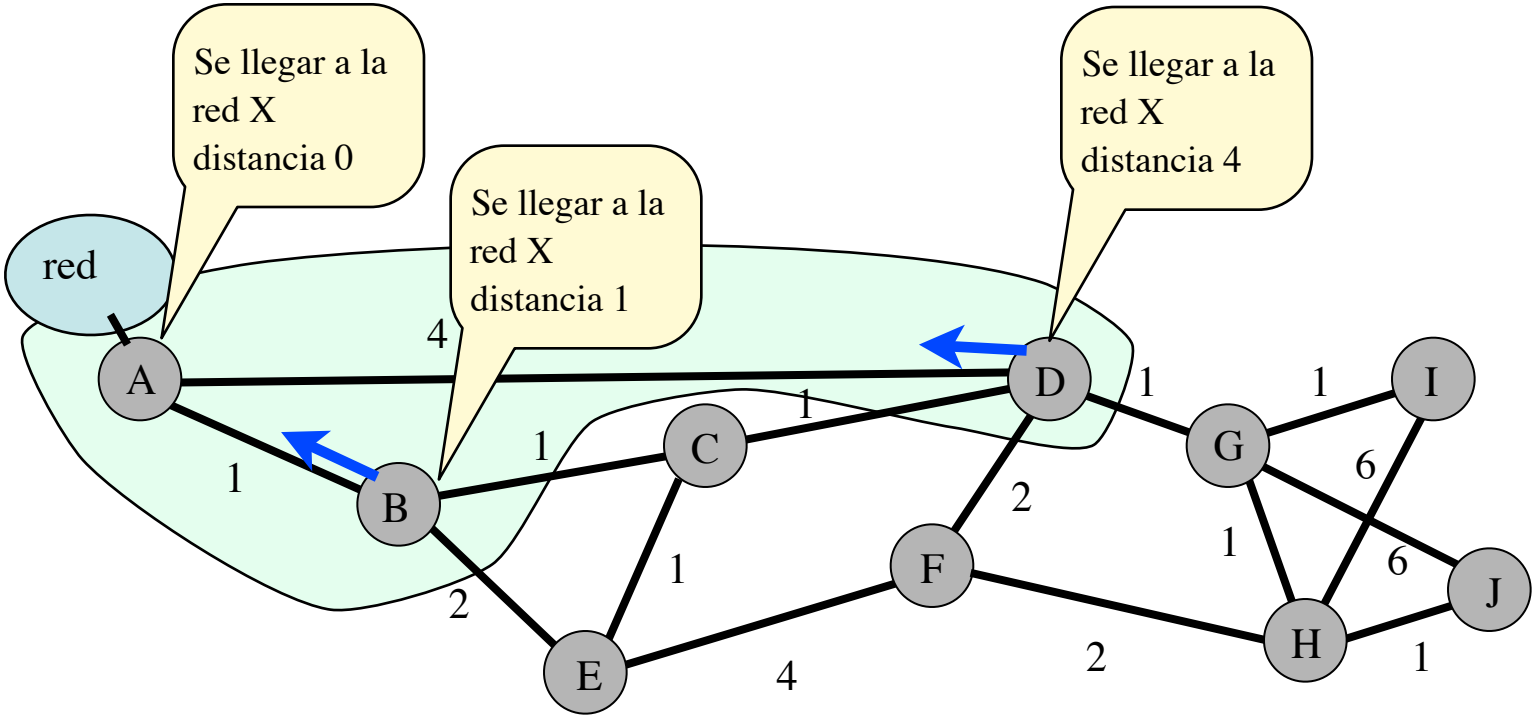


Distance-Vector

- 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...

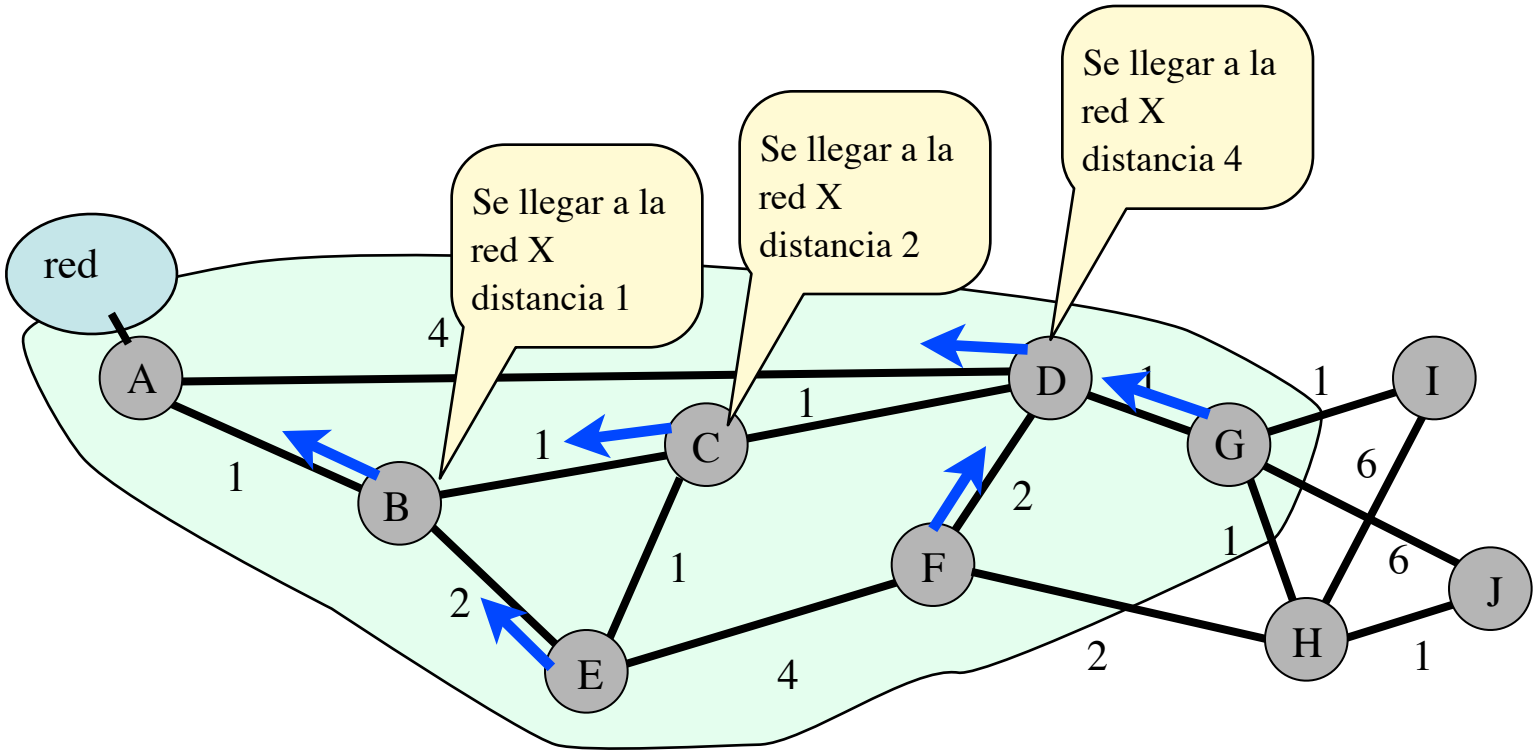
Y esto funciona?

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



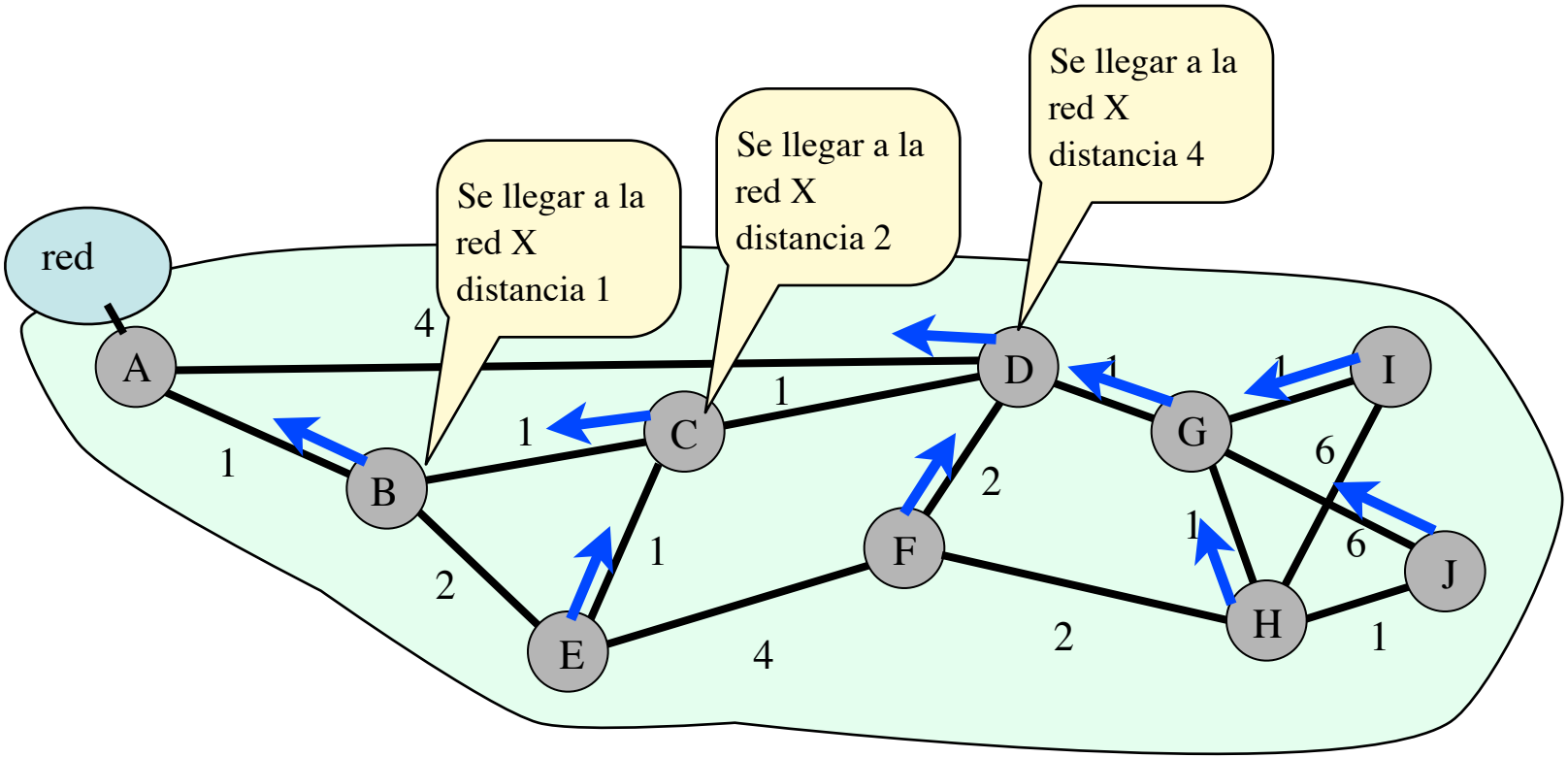
Y esto funciona?

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



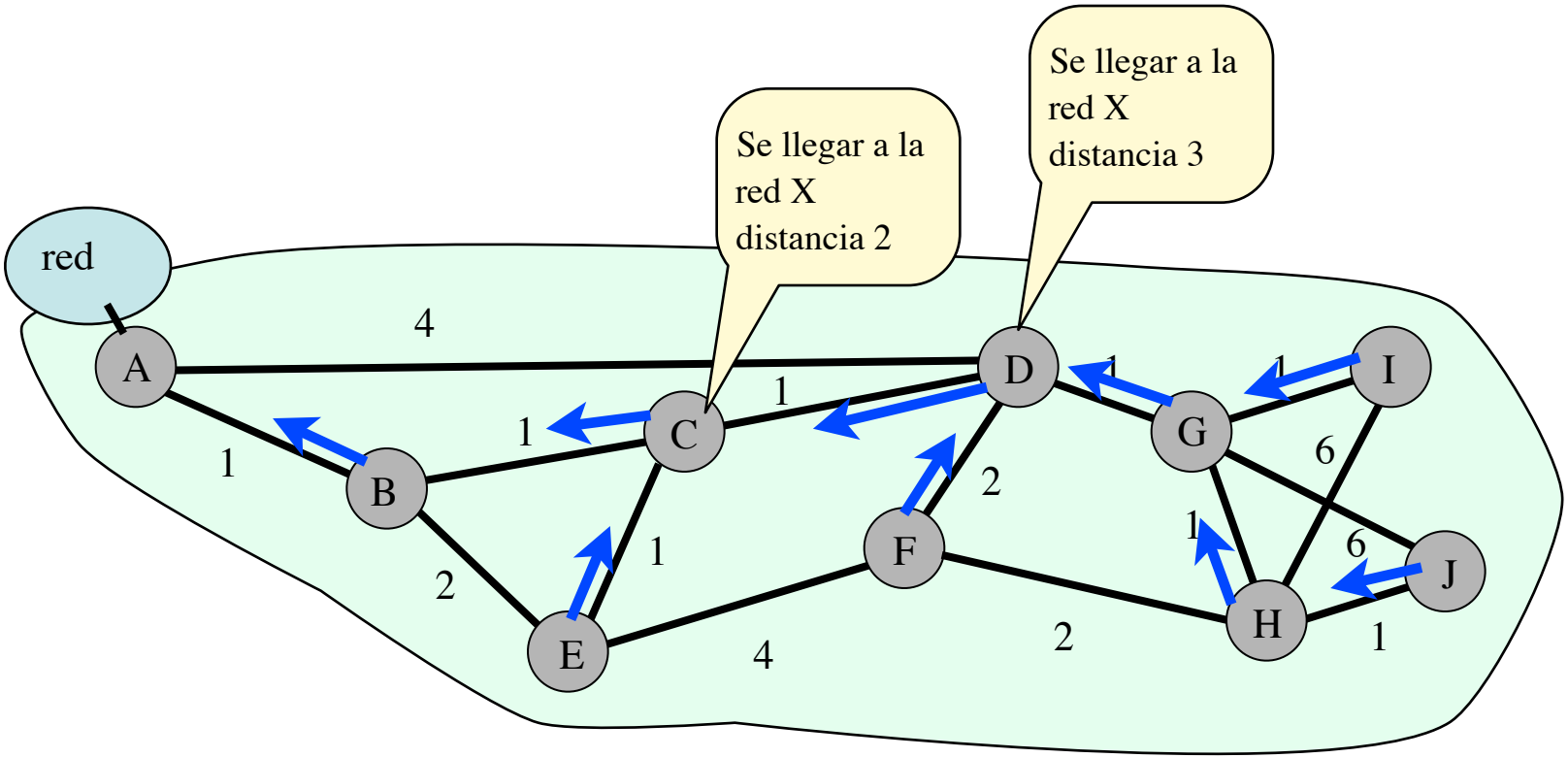
Y esto funciona?

- 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