

# Routing: Protocolos *Distance Vector*

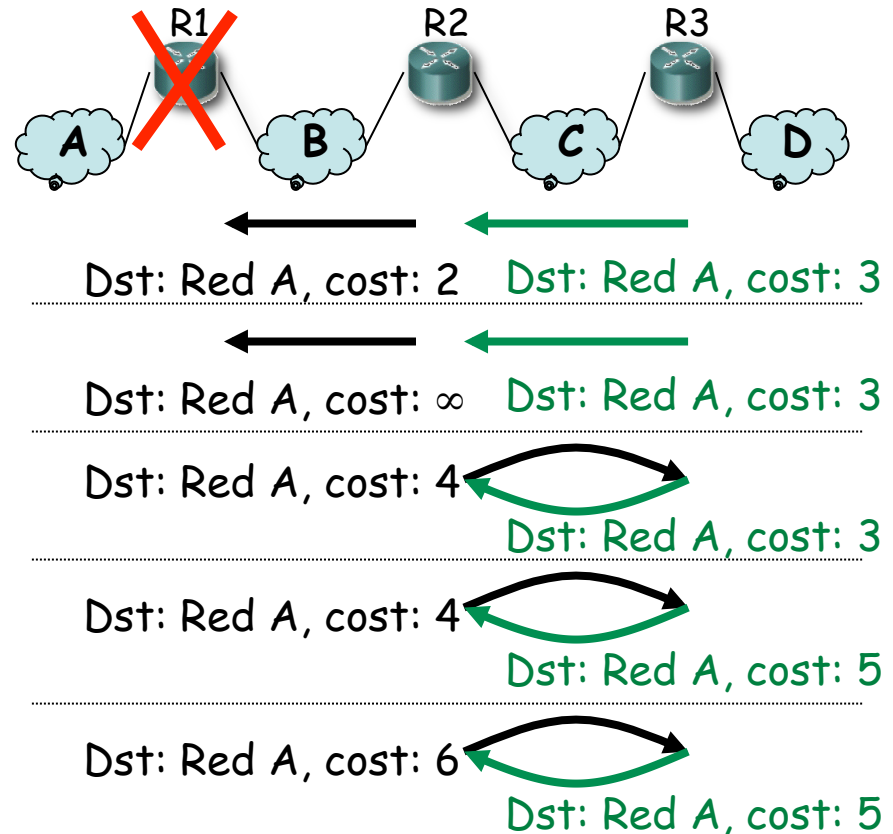
Area de Ingeniería Telemática  
<http://www.tlm.unavarra.es>

Grado en Ingeniería en Tecnologías de  
Telecomunicación, 3º

# RIP: Cuentas a infinito

# Bad news travel slowly

- Supongamos que R1 falla (...)
- Aprox. 3min después R2 marca la ruta como inválida (...)
- Si antes de que envíe el vector a R3 se lo envía él (...)
- ¡ Ahora piensa que se va por R3 !
- Pero cuando informa a R3 del nuevo camino éste verá un aumento en el coste (...)
- Y así *ad infinitum* (...)
- Proceso de cuenta a infinito
- Infinito = 16 !



# Cuentas a infinito

## Split horizon

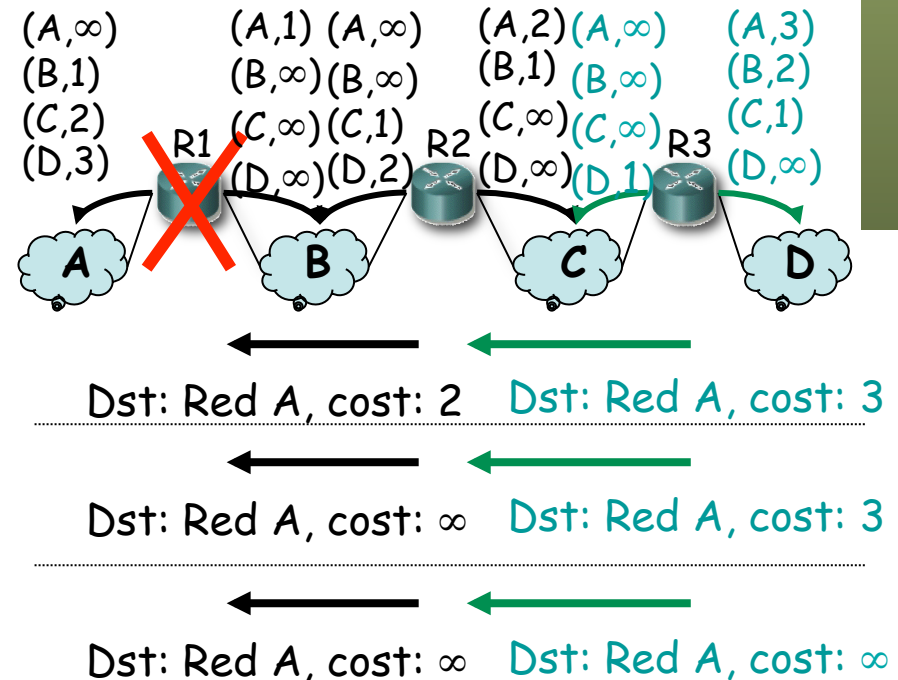
- Al enviar vector por un interfaz **no incluir** los destinos a los que se llega por él
- Mensajes más pequeños
- Evita el bucle anterior

Ejemplo (... ..):

- Caduca timer (180s) en R2 (...)
- Caduca timer (30s) en R2, envía vector (...)

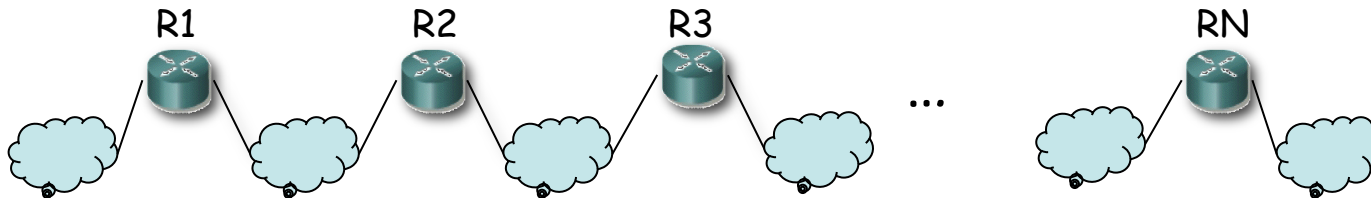
## Split horizon with poisoned reverse

- Al enviar vector por un interfaz anunciar los destinos a los que se llega por él con métrica  $\infty$
- No hay que esperar al timeout de la ruta
- Mensajes vuelven a ser grandes



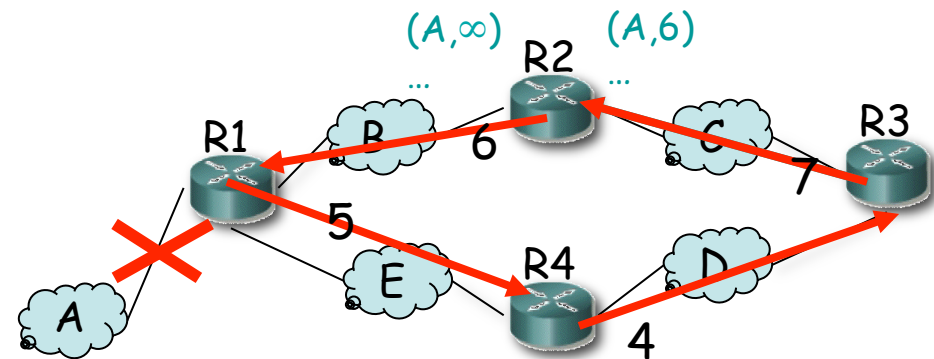
# *Bad news travel slowly*

- Convergencia lenta
- Ejemplos:
  - Actualización de información
    - Caso peor  $N \times 30\text{seg}$  para llegar al otro extremo
  - Pérdida de ruta
    - Caso peor  $N \times 180\text{seg}$  hasta el otro extremo
- ¿ Mejorar estos tiempos ?
  - **Triggered updates**: Enviar el vector en cuanto se produzca un cambio en el mismo



# Cuentas a infinito

- Supongamos la topología de la figura
- Usan *split horizon with poisoned reverse*
- Las flechas son las rutas hacia la Red A (...)
- Supongamos que falla el interfaz de R1 en la Red A (...)
- R1 anuncia coste  $\infty$  a R2 y R4 (...)
- Puede que antes de que avisen a R3 él envíe su actualización periódica (...)
- R4 introduce una entrada hacia la Red A por R3 (...)
- R4 anunciará esa ruta a R1 (...)
- R1 creerá que se llega por R4 con coste 5 (...)
- R1 lo anunciará a R2 (...)
- R2 creerá que se llega por R1 (...)
- Y luego R2 hasta llegar a R3 (...)



¡ Cuenta a infinito !

# Cuentas a infinito

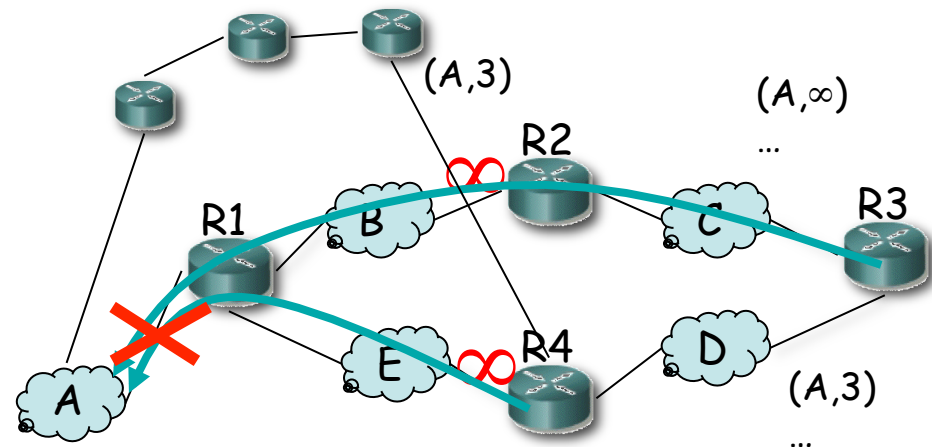
## Solución

- *Hold down period*
- Al marcar una ruta como inválida
- Esperar un tiempo antes de aceptar nuevas rutas a ese destino
- Ejemplo:
  - R4 entra en *hold down*
  - Ignora ruta anunciada por R3

## ¿Cuánto esperar?

- Depende del tamaño de la red
- Se sobredimensiona (120s)
- Si hay una ruta alternativa tardará en descubrirla (...)

Split horizon + posioned reverse +  
 Triggered updates + hold down interval  
**¡ Ya no es tan simple !**



# RIP: Versiones



# Otros problemas

- Para redes pequeñas
  - $16 = \infty$
  - Malos tiempos de convergencia (cuentas a infinito)
- Anuncia una ruta con la dirección de la red (sin máscara)
  - ¡ Solo sirve para redes *classful* !
  - También para subredes clásicas (*subnetting*) ¿Cómo? (...)

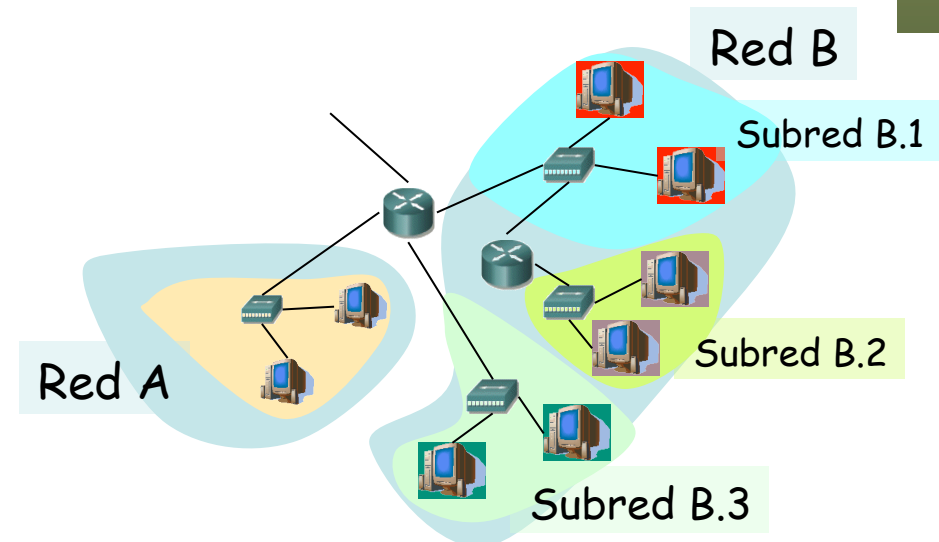
# RIPv1 y *subnetting* clásico

## Forwarding

- Router calcula el NetworkID de la red a la que pertenece la dirección destino (classful)
- ¿Tiene un interfaz en esa red?
  - No: Red destino identificada
  - Sí: Toma la máscara del interfaz que tiene en esa red y calcula el ExtendedNetworkID

## RIPv1

- Al recibir mensaje toma la máscara del interfaz
- Sirve mientras internamente se use la misma máscara en todas las subredes



# RIPv2

## Route Tag

- Para distinguir rutas internas de externas
- Debe mantenerse y reenviarse
- Ejemplo: AS number

## Subnet mask

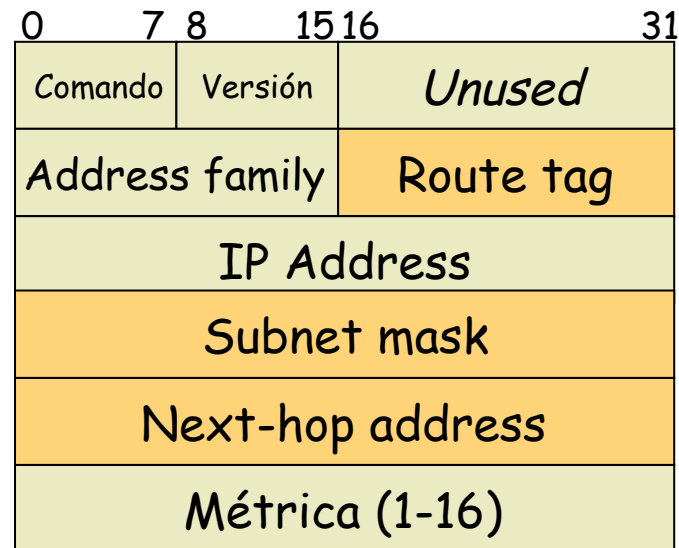
- Soporta CIDR

## Next-hop

- A quién reenviar
- 0.0.0.0 = este router
- Otro, debe ser directamente accesible

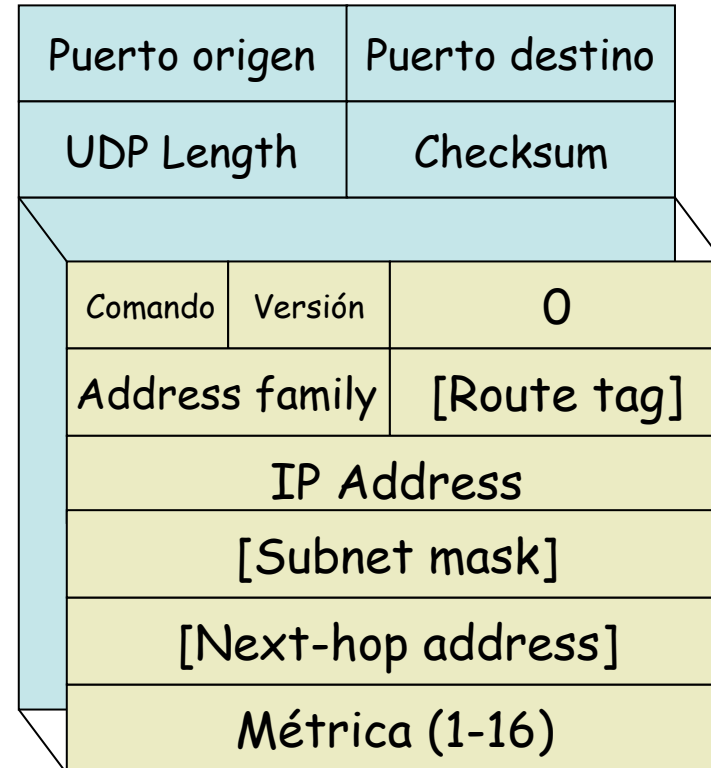
## Autenticación

- Primera entrada *addr. family* = 0xFFFF
- *Route tag* = tipo (2 ó 3)
  - 2 : password (texto plano en el resto)
  - 3 : autenticación criptográfica (RFC 4822)



# Transporte de RIP

- RIP se transporta dentro de datagramas UDP
- Puerto reservado: 520
- *Updates* periódicos enviados al puerto 520
- *Updates* enviados con puerto origen 520
- Respuestas a un *request* se envían al puerto origen del mismo
- IP destino:
  - RIPv1: Broadcast
  - RIPv2: Multicast (224.0.0.9 *RIP2 Routers*)



# (E)IGRP

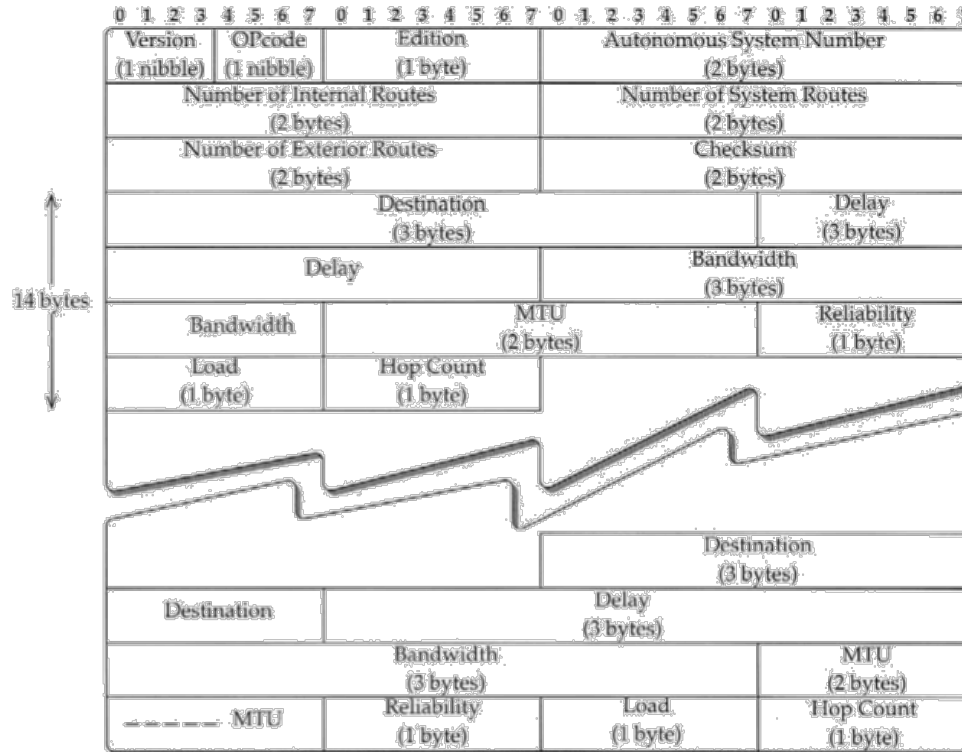
# IGRP

- Propietario de Cisco (Interior Gateway Routing Protocol)
- Distance-vector
- Classful (no soporta máscaras de longitud variable)
- Soportar redes más grandes ( $16 < \infty$ )
- Emplea split-horizon, poison-reverse y holdown-timer
- Updates cada 90s (+-)
- Paquetes a broadcast
- Directamente sobre IP (protocolo 9 reservado para un IGP)



# IGRP

- Puede calcular múltiples rutas a un destino para permitir balanceo (aunque no tengan el mismo coste)
- Puede transportar un ASN (distingue instancias concurrentes)
- Puede anunciar rutas al exterior que se emplean para seleccionar la ruta por defecto



# IGRP: métrica

- Métrica combinación no lineal con pesos ( $K_1 \dots K_5$ )
- Bandwidth (B)
  - $B = 10^7 / B_{raw}$ , ( $B_{raw}$  es la menor capacidad en kbps en el camino)
- Delay (D)
  - ante red descargada
  - $D = D_{raw} / 10$ ,  $D_{raw}$  acumulado en el camino, en  $\mu s$
- Reliability (R)
  - medida de paquetes que cruzan el enlace (1-255)
- Load (L)
  - carga de tráfico (1-255)
  - exponential weighted average de 5min actualizada cada 5s

$$C = \begin{cases} (K_1 \times B + K_2 \times \frac{B}{256-L} + K_3 \times D) \times \left( \frac{K_5}{R+K_4} \right), & \text{if } K_5 \neq 0 \\ K_1 \times B + K_2 \times \frac{B}{256-L} + K_3 \times D, & \text{if } K_5 = 0. \end{cases}$$



# IGRP: métrica

- Bandwidth (B), Delay (D), Reliability (R), Load (L)
- Anuncia todos los valores, no la combinación
- También anuncia la MTU y el número de saltos
- Por defecto  $K_1=K_3=1$  y  $K_2=K_4=K_5=0$
- Es decir, por defecto  $C = B + D$
- Métrica de 24bits

$$C = \begin{cases} (K_1 \times B + K_2 \times \frac{B}{256-L} + K_3 \times D) \times \left(\frac{K_5}{R+K_4}\right), & \text{if } K_5 \neq 0 \\ K_1 \times B + K_2 \times \frac{B}{256-L} + K_3 \times D, & \text{if } K_5 = 0. \end{cases}$$

# EIGRP

- Propietario de Cisco (Enhanced Interior Gateway Protocol)
- Classless
- Paquetes a multicast 224.0.0.10 (*IGRP Routers*)
- Es distance-vector, anuncia: {destino, next-hop, distancia}
- Directamente sobre IP (protocolo 88)
- Puede usar autenticación en los mensajes
- Métrica de 32bits
- $C_{EIGRP} = 256 \times C_{IGRP}$
- Vecinos se comunican los pesos y deben ser iguales



# EIGRP

- DV pero no emplea la ecuación de Bellman-Ford
- Emplea DUAL (Diffusing Update Algorithm)
- Con DUAL evita los bucles de enrutamiento (probado matemáticamente)
- Anuncios son confirmados (en unicast, es un *reliable multicast*)



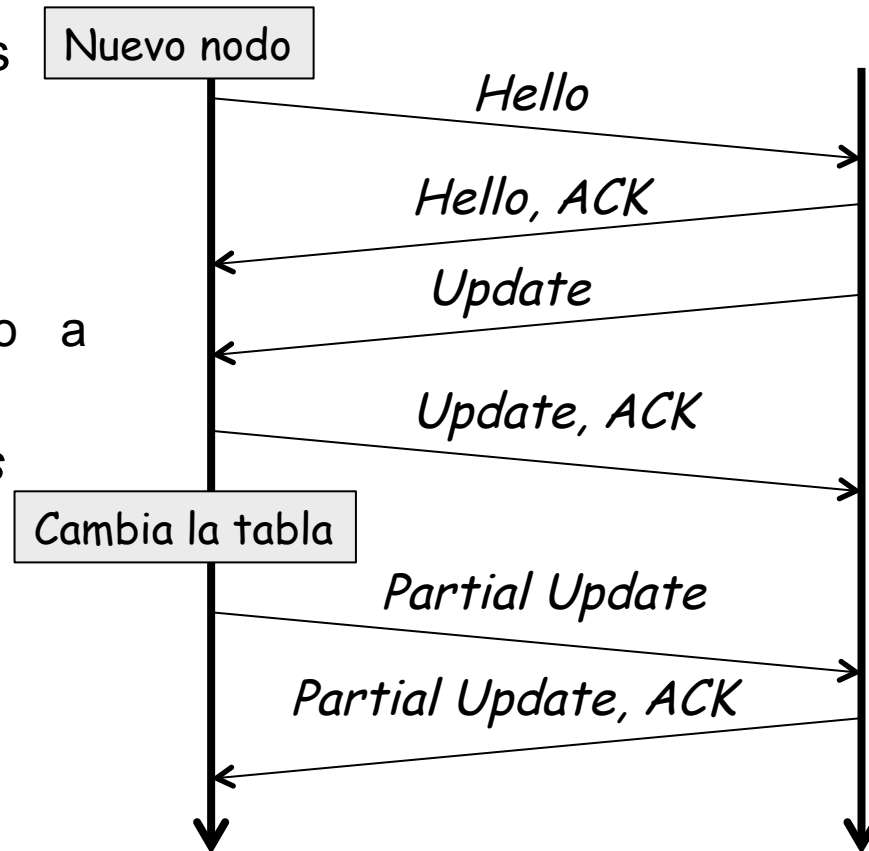
# DUAL

# DUAL / EIGRP

- Descubre nodos adyacentes y pérdida de conectividad
  - Mensajes *HELLO* (periódicos, multicast) en EIGRP no confirmados
  - Deben tener mismo ASN y pesos ( $K_i$ ) para ir a la lista de vecinos
  - Si de un vecino no se recibe ACK se retransmite en unicast
  - Vecino se considera inalcanzable tras 16 retransmisiones
- Updates fiables y ordenados
  - Stop&Wait
  - No son periódicos
  - Bajo demanda (unicast)
  - O Ante cambios solo a afectados (multicast)
- También hay *queries/replies*

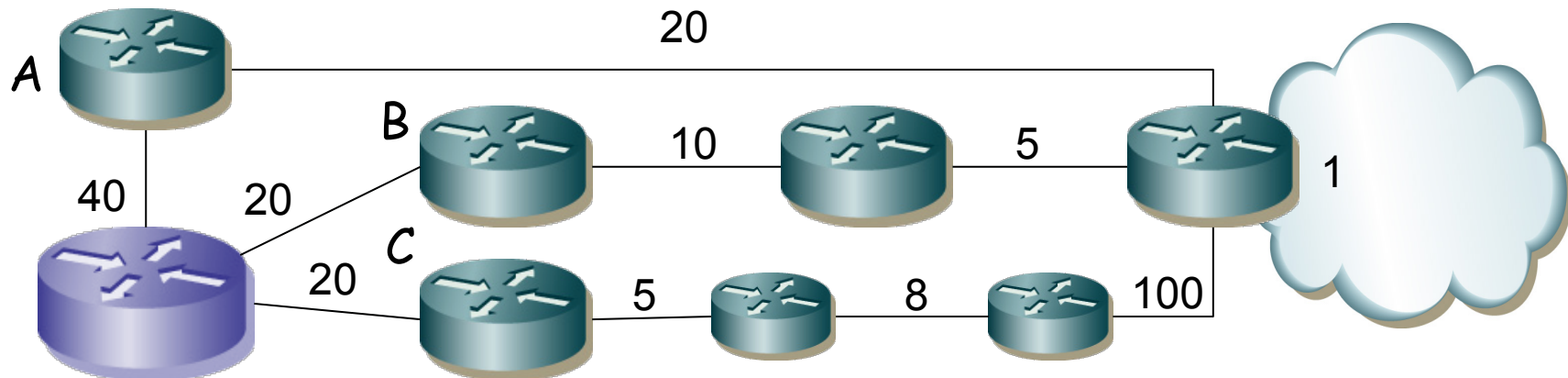


J.J. Garcia-Luna-Aceves



# DUAL

- La **distancia viable** es la menor al destino (*feasible distance*)
- **Condición de viabilidad:** un vecino la cumple para un destino si la distancia que anuncia es menor que la distancia viable del router (*feasibility condition*)
- Un **sucesor** es un vecino que cumple la condición de viabilidad y tiene el menor coste al destino (*successor*)
- Introduce en la tabla de rutas todos sucesores
- Un **sucesor viable** es un vecino que cumple la condición
- Un sucesor viable anuncia una ruta que no pasa por este nodo (pues el coste es menor) luego anuncia una ruta sin ciclos
- Ejemplo: {A,B}



# DUAL

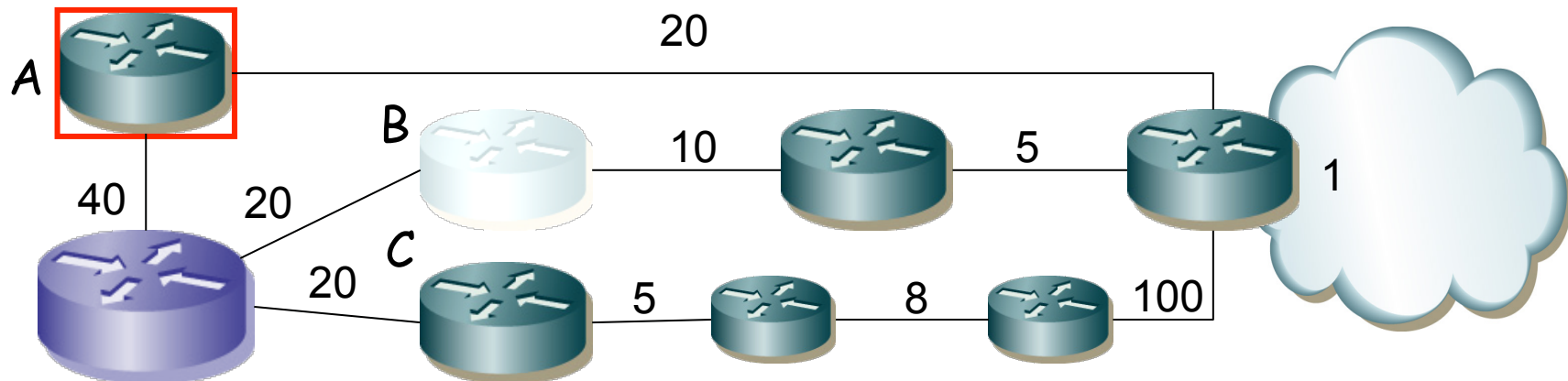
- Cada nodo tiene una tabla con todos los nodos y
  - La distancia viable
  - Los sucesores viables y sus distancias anunciadas
  - El coste al destino por cada sucesor viable
  - El interfaz por el que se encuentra cada sucesor viable
  - Estado (activo=recalculando o pasivo)

---

---

# DUAL

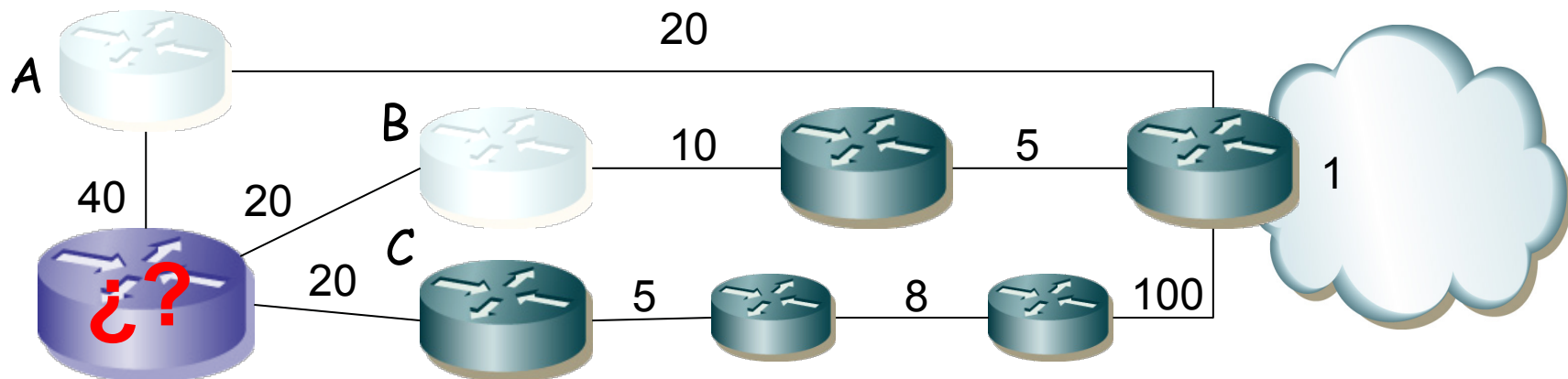
- Si la ruta deja de ser alcanzable por un sucesor pero hay uno viable se cambia a éste (sigue “pasivo”) y manda *updates*





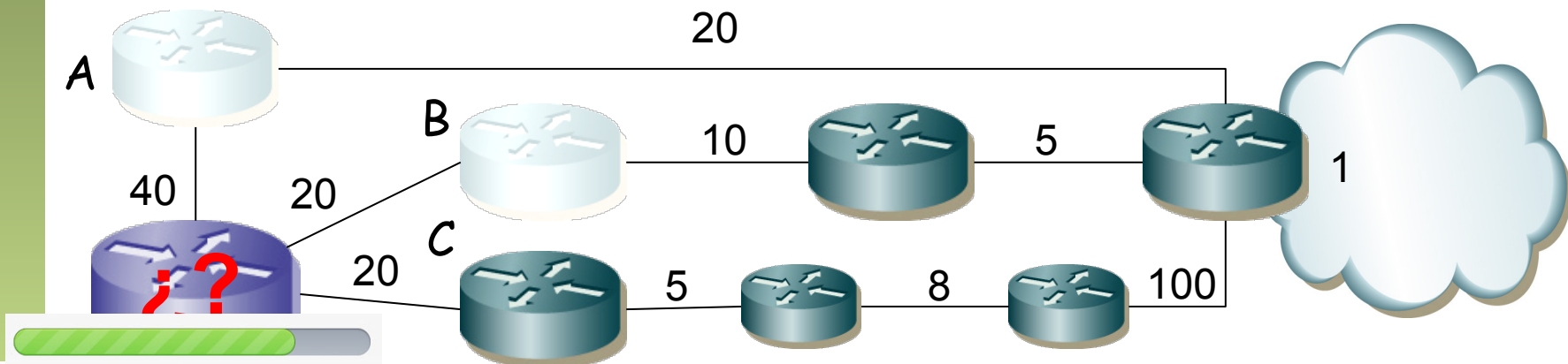
# DUAL

- El estado de la ruta pasa a “activo” cuando el router deja de tener un sucesor viable para un destino



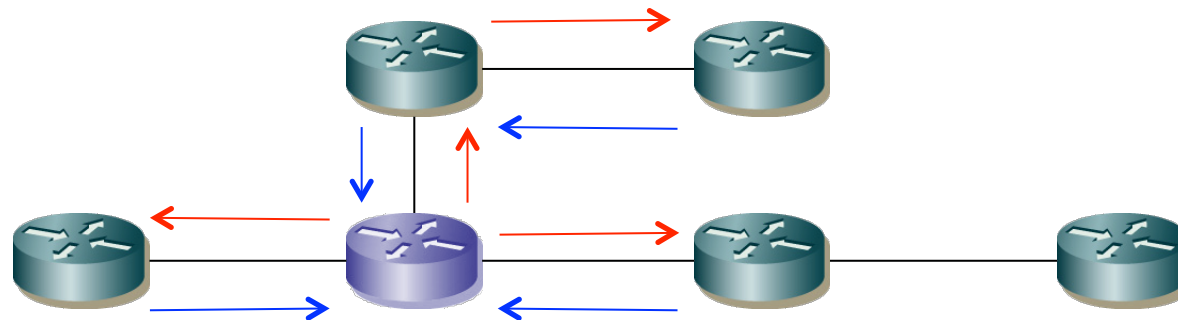
# DUAL

- Al pasar a activo inicia una *diffusing computation*
- En estado activo no puede:
  - Cambiar de sucesor para la ruta
  - Cambiar la distancia que anuncia para la route
  - Cambiar la distancia viable de la ruta
  - Iniciar otra *diffusing computation* para esta ruta



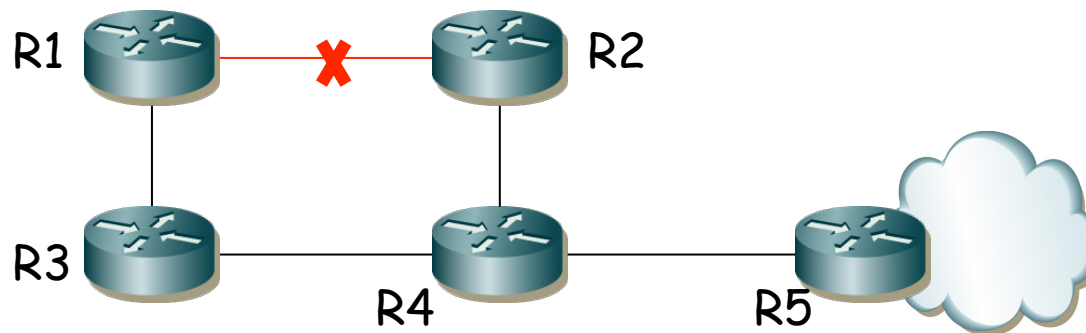
# Diffusing Computation

- Envía *queries* a todos sus vecinos
- Incluye su nueva distancia calculada al destino
- Cada vecino recalcula con esa nueva información
- Si el vecino tiene algún destino viable responde con su mínimo coste
- Si el vecino no tiene destino viable pasa la ruta a “activo” e inicia una *diffusing computation* (. . .)
- Se ha completado cuando se ha recibido respuesta de todos los vecinos (pasa al estado “pasivo”)
- Hay un timer para las respuestas y si caduca se elimina al vecino



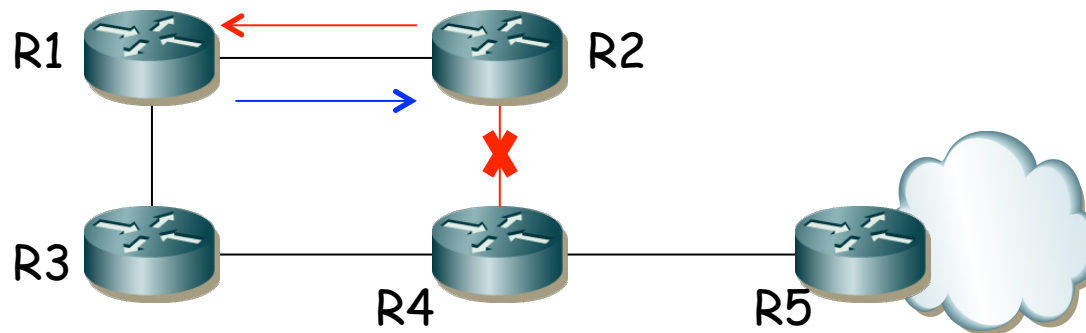
# Ejemplo

- Iguales costes
- Supongamos que falla el enlace R1-R2
- R1 tiene otro sucesor viable que es R3
- No hace falta ningún cálculo



## Ejemplo 2

- Iguales costes
- Supongamos que falla el enlace R2-R4
- R2 no tiene otro sucesor viable así que inicia una computación difusa
- Informa a R1 de que ha perdido a su sucesor (...)
- R1 sí tiene otro sucesor viable, R3
- Cambia a él y le notifica a R2 (...)
- Al recibir respuesta sabe R2 que ha terminado el cálculo por esa rama y puede tomarle como sucesor
- R3 y R4 no han tenido que hacer nada



# Otras características

- Ante cambios puede generar bastante tráfico en un periodo breve de tiempo
- Propietario
- Toma algunos mecanismos de protocolos *link-state*