Routing: Protocolos *Distance Vector*

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

Grado en Ingeniería en Tecnologías de Telecomunicación, 3º

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Tecnologías Avanzadas de Red
Área de Ingeniería Telemática

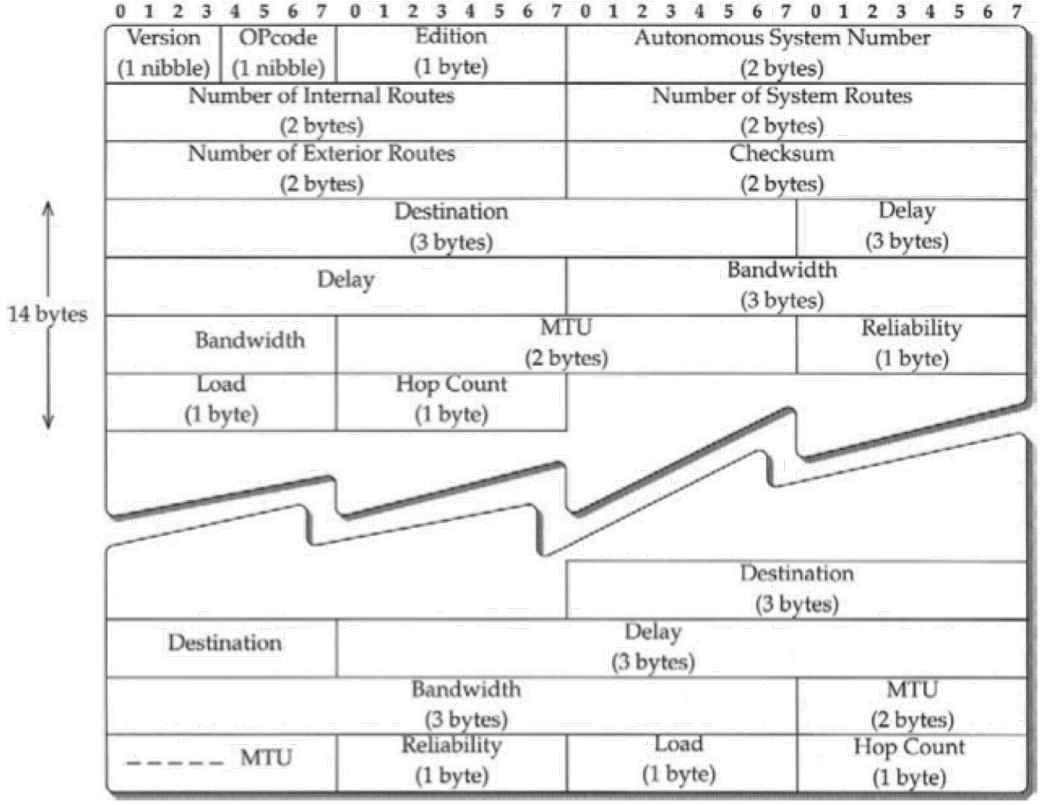
(E)IGRP

IGRP

- Propietario de Cisco (Interior Gateway Routing Protocol)
- Distance-vector
- Classful (no soporta máscaras de longitud variable)
- Soportar redes más grandes (16 < ∞)
- Emplea spit-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 ($K_1 \ldots K_5$)
- Bandwidth (B)
  - $B=10^7/Braw$, (Braw es la menor capacidad en kbps en el camino)
- Delay (D)
  - ante red descargada
  - D=Draw/10, Draw acumulado en el camino, en μs
- 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

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

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 K1=K3=1 y K2=K4=K5=0
- Es decir, por defecto C = B + D
- Métrica de 24bits

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

EIGRP

- Propietario Cisco (Enhanced Interior Gateway Protocol, 1993)
- http://www.cisco.com/go/eigrp
- RFC Informativa 7868 (2016)
- 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 autentificación 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*)

CISCO

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Tecnologías Avanzadas de Red
Área de Ingeniería Telemática

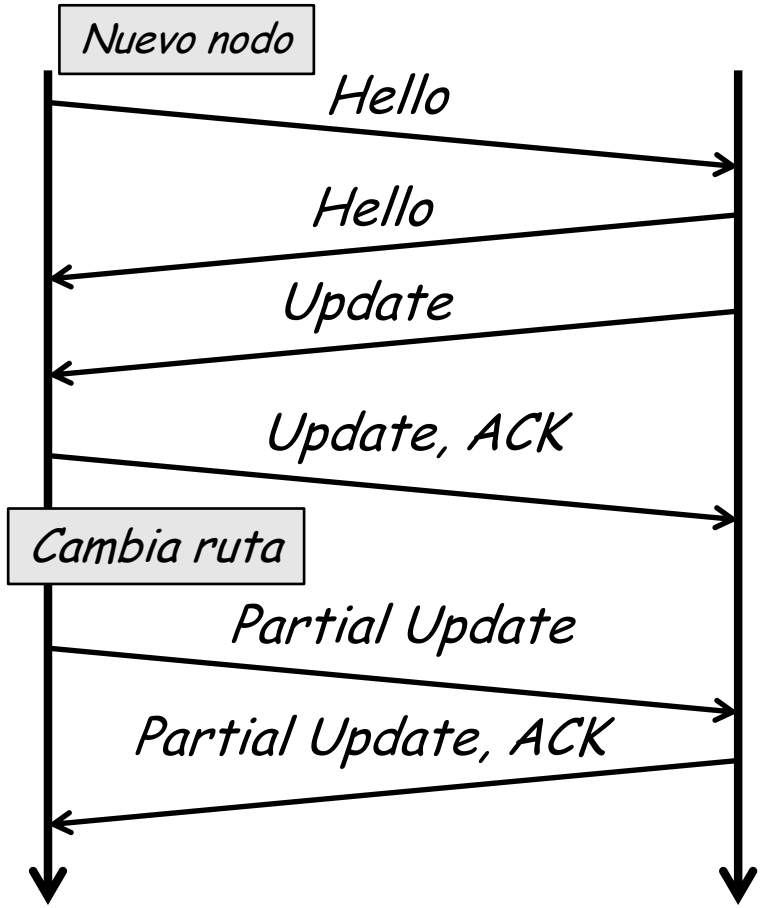
DUAL

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
- Updates fiables y ordenados
  - Si de un vecino no se recibe ACK se retransmite en unicast
  - Vecino inalcanzable tras 16 retransmisiones
  - Stop&Wait
  - No son periódicos
  - Bajo demanda (unicast)
  - O Ante cambios solo a afectados (multicast)
- También hay *queries/replies*

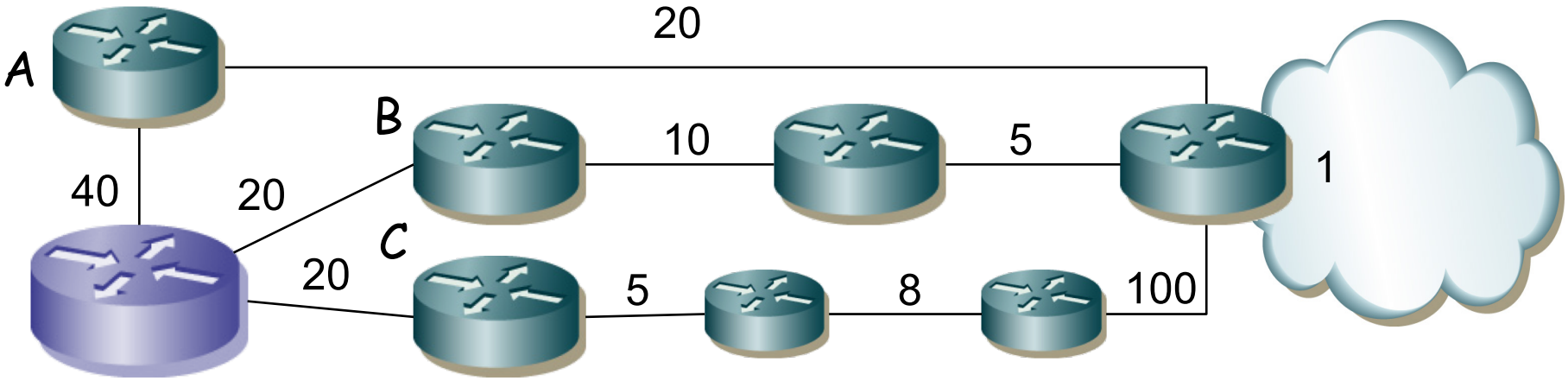
J.J. Garcia-Luna-Aceves

J.J. Garcia-Luna-Aceves, "Loop-Free Routing Using Diffusing Computations", IEEE/ACM Trans. On Networking vol. 1, n. 1, Feb. 1993



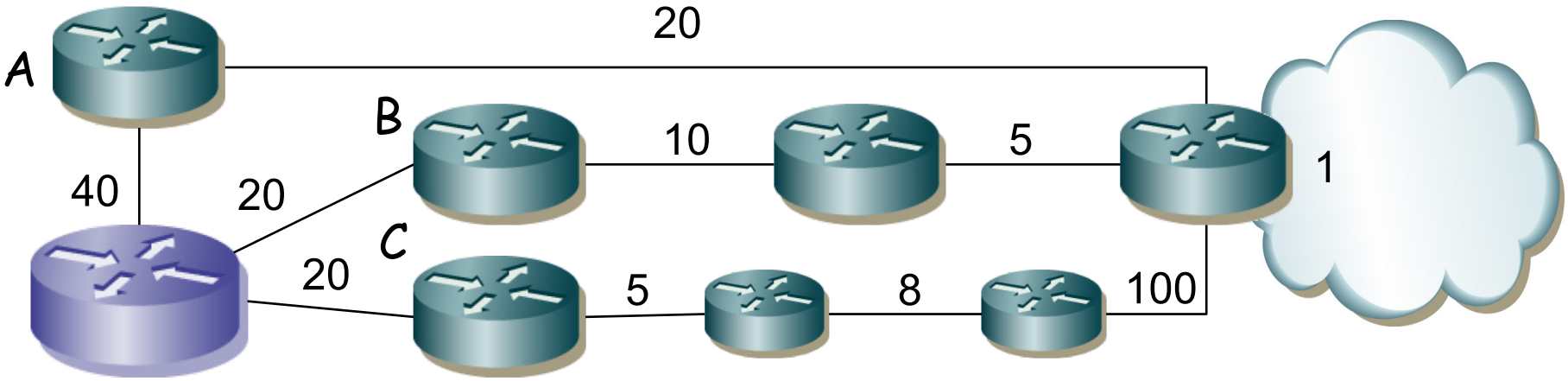
DUAL

- La **distancia viable** es la menor al destino (*feasible distance*)
- Ejemplo: 36 (por B)



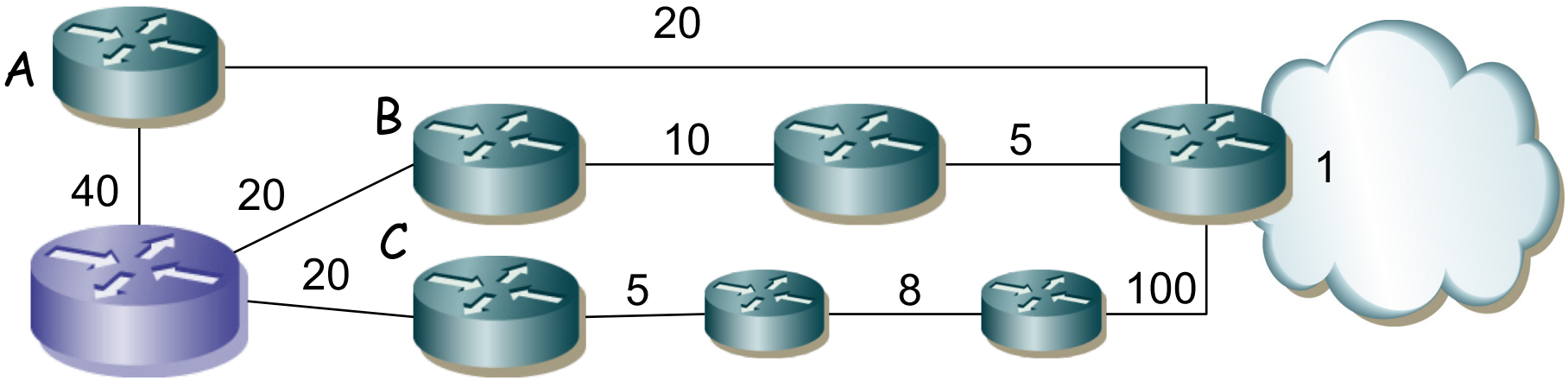
DUAL

- 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*)
- Ejemplo: distancia viable=36, vecinos que cumplen={A,B}



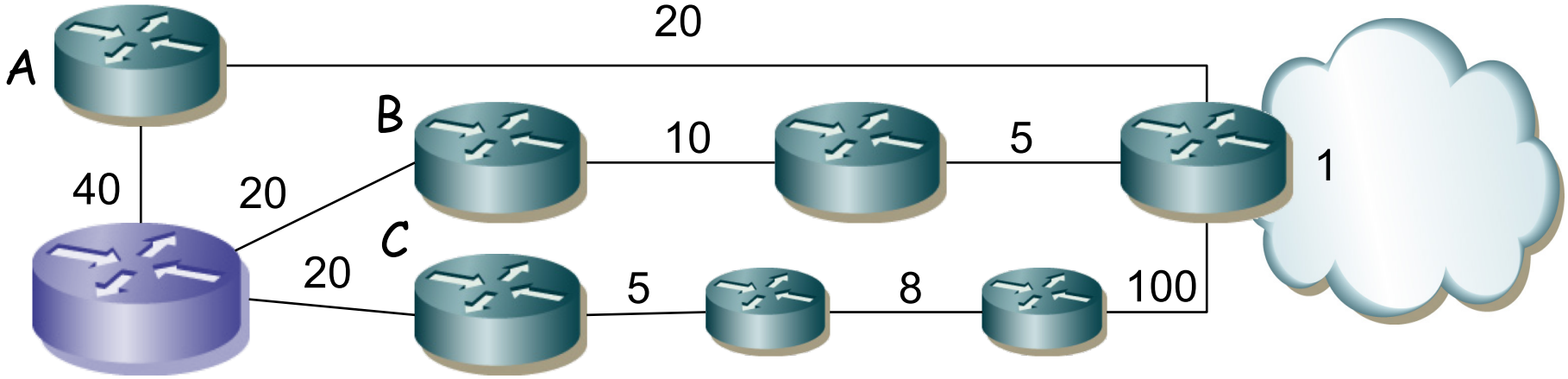
DUAL

- 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 los sucesores (podría añadir otros con coste ligeramente mayor)
- Ejemplo: B



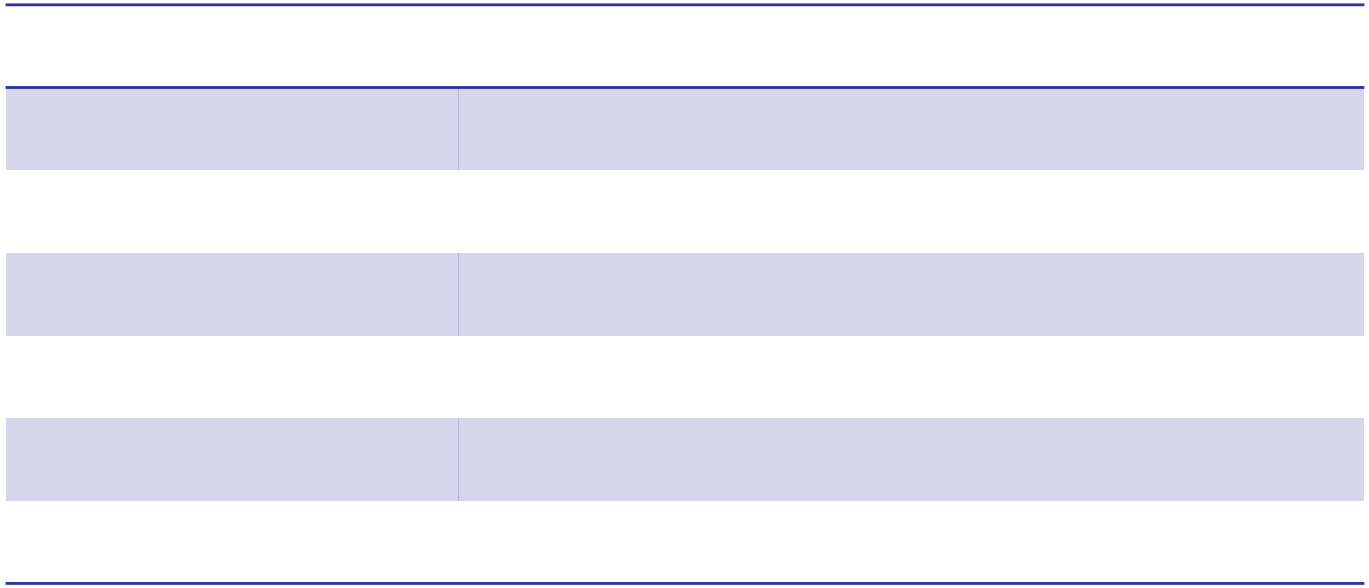
DUAL

- 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 los 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}



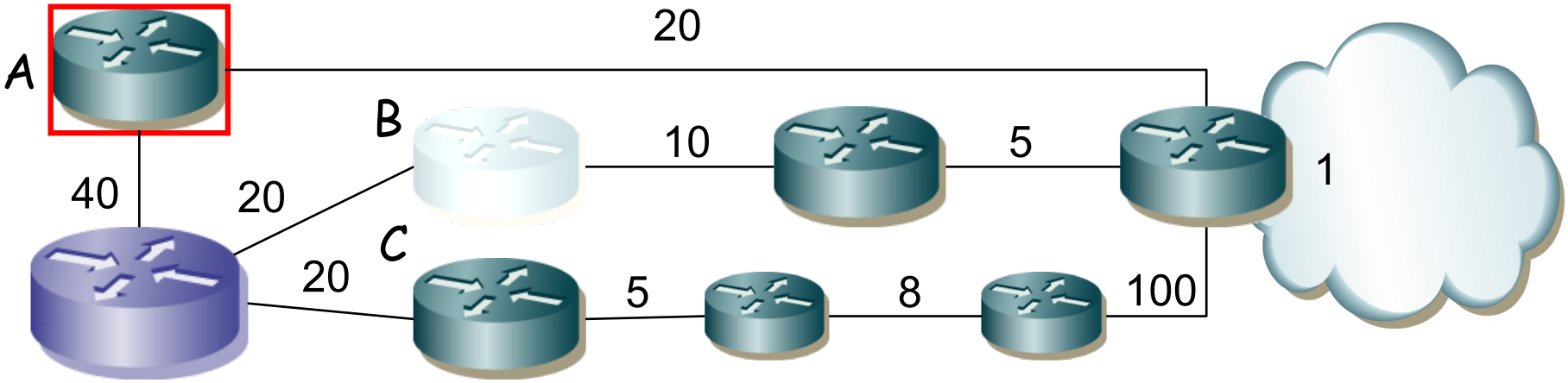
DUAL

- 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)



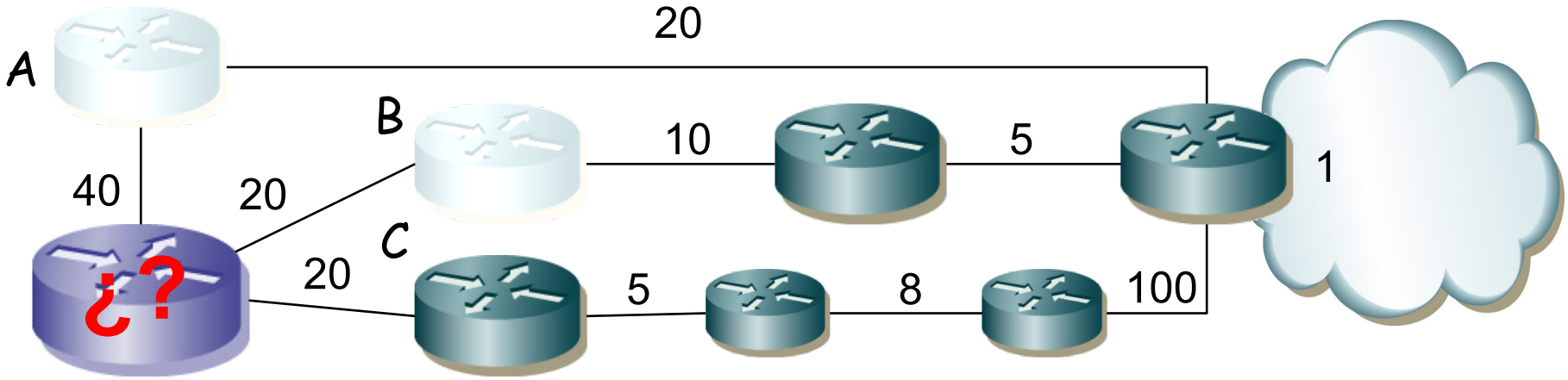
DUAL

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



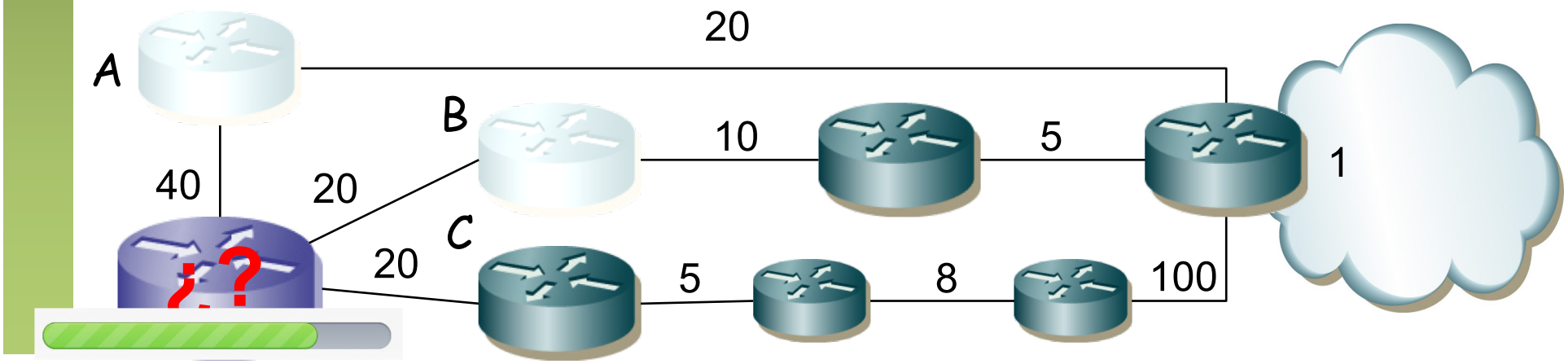
DUAL

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



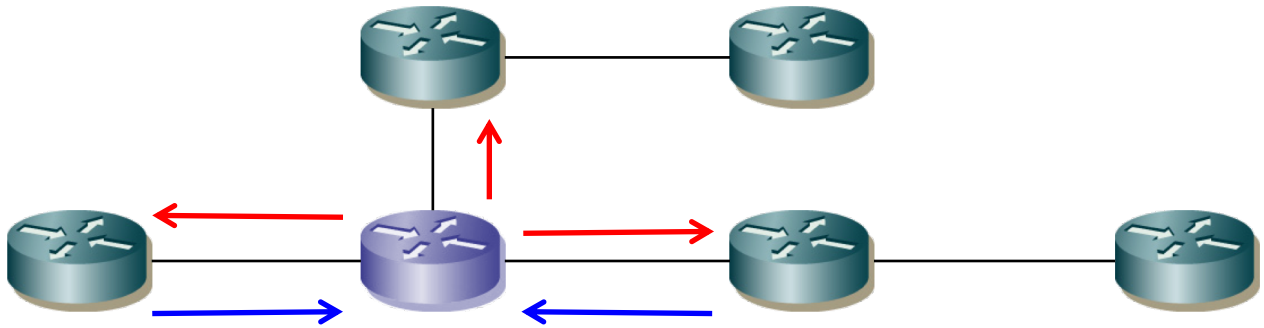
DUAL

- 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 ruta
  - Cambiar la distancia viable de la ruta
  - Iniciar otra *diffusing computation* para esta ruta



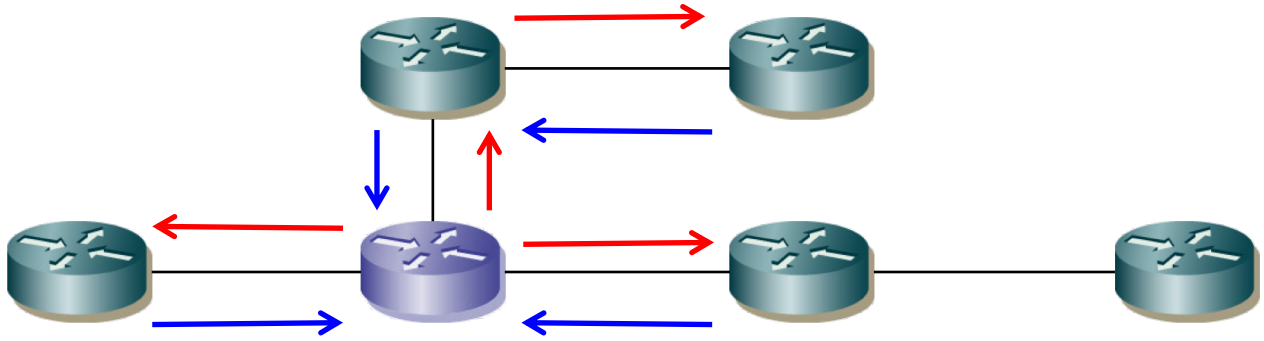
Diffussing Computation

- 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 (…)



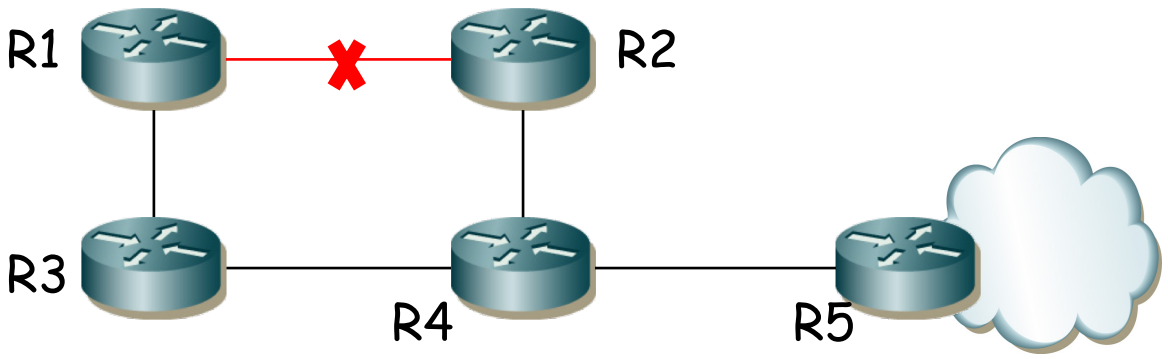
Diffussing Computation

- 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



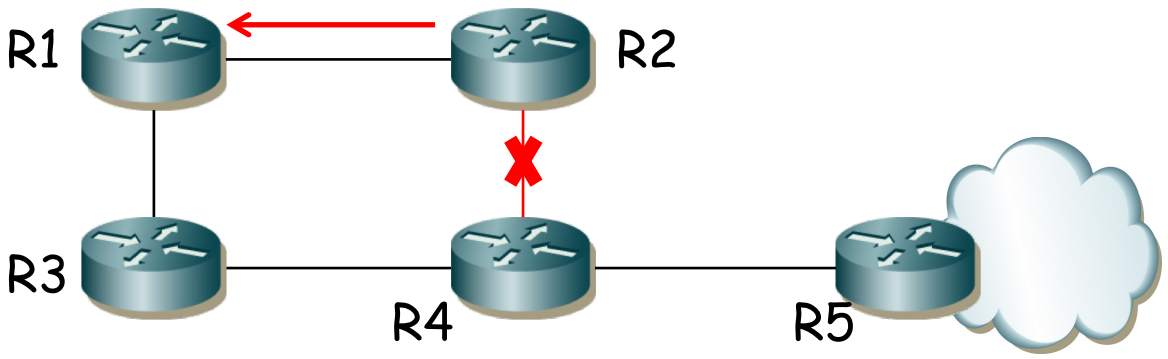
Ejemplo

- Ruta de R1 a la red de R5
- Iguales costes
- Supongamos que falla el enlace R1-R2
- R1 tiene otro sucesor viable que es R3
- No hace falta ningún cálculo



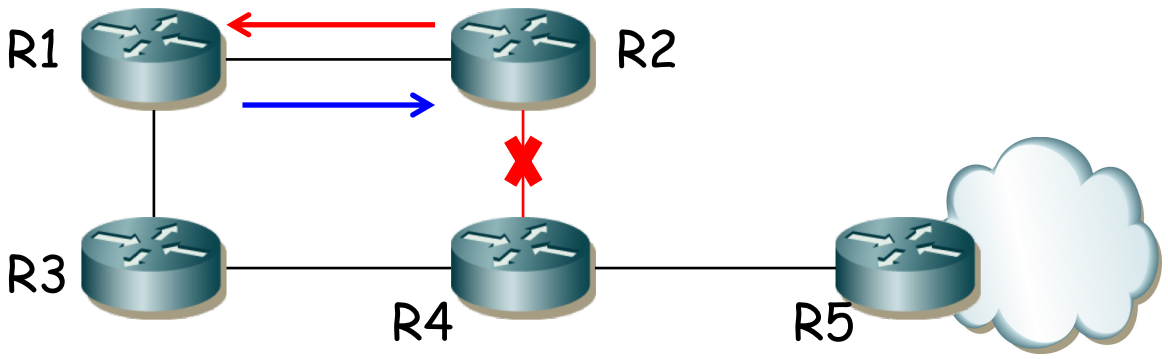
Ejemplo 2

- Ruta de R2 a la red de R5
- 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 (…)
- (…)



Ejemplo 2

- Ruta de R2 a la red de R5
- 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 (si antes usaba a R2) y le notifica a R2 (…)
- Al recibir respuesta sabe R2 que ha terminado el cálculo por esa rama y puede tomar a R1 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? (RFC 7868, 2016)
- Toma algunos mecanismos de protocolos *link-state*