# Encaminamiento en Internet
# 2. RIP
## Redes - I

Departamento de Sistemas Telemáticos y Computación (GSyC)

Octubre de 2009

# Contenidos







# Contenidos

# RIP *(Routing Information Protocol)*

- RIP es el protocolo interior más usado en Internet, aunque poco a poco se va reemplazando por OSPF.
- Distribuido originalmente con UNIX BSD, demonio routed, en 1982.
- Deriva de GGP (*Gateway to Gateway Protocol*), usado en los primeros tiempos de Internet.
- Es un protocolo basado en vectores de distancia.
- Versiones de RIP:
  - RIPv1 (RFC-1058, Jun 1988).
  - RIPv2 (RFC-2453, Nov 1998).





# Contenidos

# Métrica

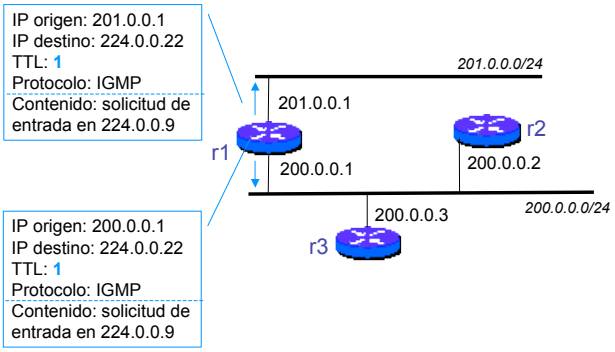
- Coste = Número de saltos *(routers)* por los que hay que pasar para alcanzar un destino.
- Con tecnologías lentas se incrementa la distancia de forma artificial.
- Una red directamente conectada a un *router* tiene coste=1.
- Se limita el número máximo de saltos a 15, lo que fija el diámetro máximo de la red.
  - Problema en redes muy grandes ⇒ no escala a toda Internet
- Un coste de 16 representa un coste infinito, es decir, un destino inalcanzable.





# RIP utiliza IP multicast

- La dirección IP multicast 224.0.0.9 está reservada para RIP.
- Cuando arranca el *router* RIP r1 envía (por todas las interfaces donde tiene activado el protocolo RIP) un mensaje IGMP de solicitud para entrar en el grupo multicast 224.0.0.9
  - Este mensaje irá dirigido al grupo 224.0.0.22, al que pertenecen todos los *routers* IGMP.
  - Este mensaje lleva TTL=1 ya que sólo sirve para informar de dicha solicitud a los *routers* IGMP locales que están conectados a la/s misma/s subred/es que r1.
- A partir de ese momento, cualquier mensaje RIP de los *routers* directamente conectados a r1 que vaya dirigido a la dirección 224.0.0.9, será recibido por r1.
- El *router* r1 utilizará la dirección destino 224.0.0.9 y TTL=1 para comunicarse con sus *routers* vecinos y enviarles la información de encaminamiento del protocolo RIP.



## Envío de información

- La información se envía a todos los *routers* vecinos, **casi siempre por multicast** (con mensajes dirigidos a la 224.0.0.9)
- El envío de información se origina de dos formas:
  - Periódicamente: Cada 30 segundos ($\pm 50\,\%$ para que todos los *routers* no terminen transmitiendo exactamente a la vez).
  - Disparado por un evento: Como consecuencia de cambios en las tablas o en respuesta a solicitudes de otros *routers*.
- Se desaconseja que "escuchen" los mensajes RIP las máquinas finales (*hosts* que no son *routers*) para mantener su tabla de encaminamiento. Es mejor que sus tablas se configuren de forma estática.



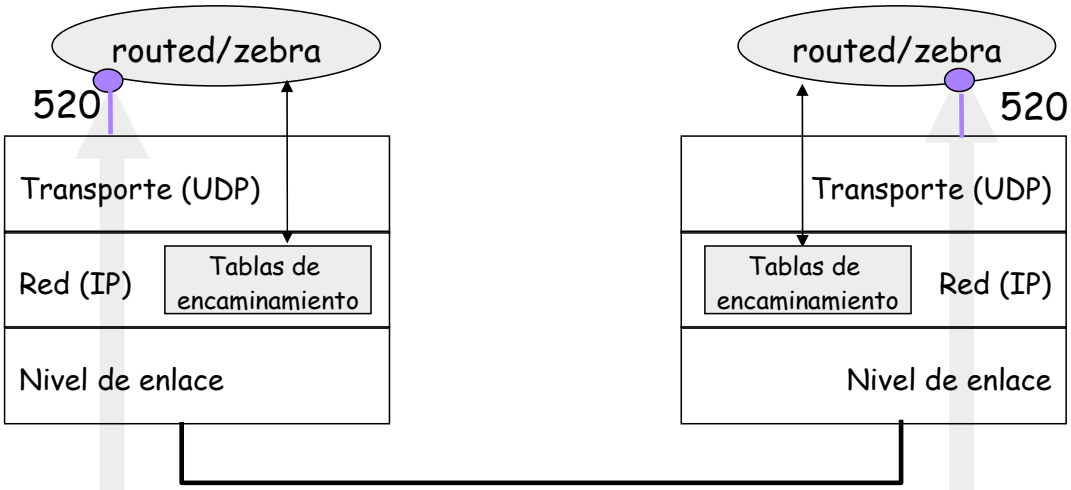


## Actualización de entradas

- RIP no actualiza una ruta existente en una tabla de encaminamiento con otra diferente si ésta no tiene una distancia estrictamente menor a la ruta actual (a no ser que la información venga del mismo *router* que está anotado en la tabla como siguiente salto para esa ruta).
  - Evita oscilaciones entre rutas de igual coste
- Si en 180 segundos (aprox. 6 períodos de actualización periódica) no se ha recibido información sobre una ruta, se elimina de la tabla de encaminamiento.

# Implementación de RIP

- RIP utiliza paquetes UDP (!!!) para intercambiar la información de encaminamiento, a través del puerto 520.



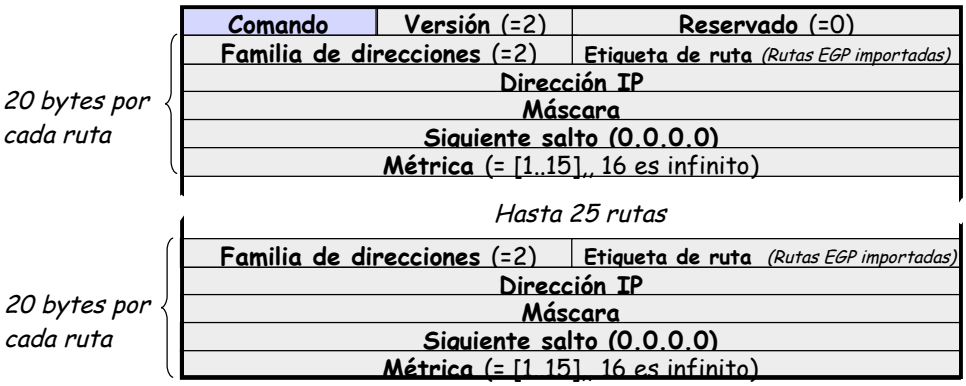





# Contenidos

# Tipos

- REQUEST (comando=1): mensaje de solicitud de información. Se envía:
  - cuando comienzan a ejecutar RIP (por multicast)
  - en situaciones de diagnóstico (no lo veremos)
- RESPONSE (comando=2): mensaje de actualización de tablas de encaminamiento. Se envía:
  - como actualización periódica, destinado a todos los vecinos (por multicast)
  - como respuesta a una solicitud, destinado a quien la ha realizado
  - si cambia algún dato en la tabla de encaminamiento, destinado a todos los vecinos (por multicast)

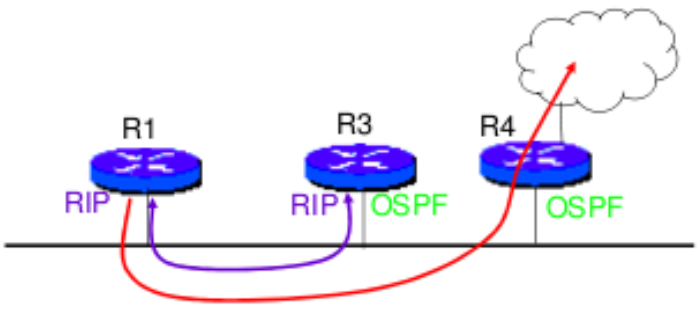







# Campo "Siguiente salto"

- El receptor de un mensaje RESPONSE utiliza este campo para saber cuál es el *router* vecino al que debe enviar los paquetes dirigidos a una cierta ruta.
- Si este campo vale 0.0.0.0, indica que el *router* vecino es el emisor del mensaje RESPONSE.
- En el siguiente ejemplo puede verse un caso en el que es conveniente utilizar el campo siguiente salto con un valor distinto de cero:



## Mensaje REQUEST

- Un mensaje REQUEST se envía normalmente a todos los *routers* de sus subredes (un mensaje de multicast 224.0.0.9 por cada una de las de las interfaces donde tiene activado RIP) cuando un *router* arranca y quiere rellenar rápidamente su tabla de encaminamiento.
- Cuando el mensaje REQUEST se recibe en los vecinos, se procesa cada una de sus entradas.
  - Si sólo hay una entrada en el mensaje REQUEST, la familia de direcciones es cero y su métrica 16, este mensaje es un mensaje de solicitud de todas las rutas existentes en la tabla del receptor.
  - En cualquier otro caso es un mensaje de solicitud de un conjunto de rutas.
    - Se utiliza normalmente para funciones de diagnóstico.
    - Para cada entrada del mensaje recibido, se comprueba la tabla de encaminamiento y si existe una ruta, se añade al mensaje de respuesta junto con el campo métrica. Si no hay una ruta, se especifica métrica infinito (valor 16).





## Mensaje RESPONSE

Una respuesta puede recibirse por alguno de los siguientes motivos:

- Respuesta a un REQUEST. En este caso, el mensaje de respuesta se envía **por unicast al solicitante**.
- Actualización periódica, cada 30 segundos (respuesta no solicitada). En este caso se envía **por multicast a todos los vecinos** (224.0.0.9).
- Actualización explícita, provocada por un cambio de ruta.
  - Llamadas Triggered Updates
  - Las implementaciones deben tratar con cuidado las actualizaciones explícitas porque pueden sobrecargar la red
    - Se puede retrasar el envío de la actualización explícita si queda poco para enviar una actualización periódica, o no enviar toda la tabla.
  - En este caso se envía **por multicast a todos los vecinos** (224.0.0.9)

# Ejemplo (I)

- Inicialmente, cuando **arranca R1** tiene en su tabla sólo las redes a las que está directamente conectado.
  - R2 y R3 ya llevan conectados un tiempo con las tablas de la figura
- R1 se conecta a las redes 15.0.0.0/24 y 16.0.0.0/24 y envía mensajes REQUEST en ellas para que sus vecinos le envíen la tabla completa.

Tabla de encaminamiento de R1

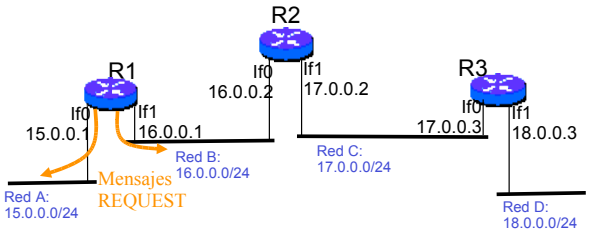
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |








# Ejemplo (II)

- R2 envía por unicast su vector de distancias en un mensaje RESPONSE
  - Se omite de la respuesta la entrada de ruta de la subred por la que se envía el paquete RESPONSE (todos los *routers* que reciben ese paquete están directamente conectados a esa subred). En este caso, el mensaje enviado a la red B omite la ruta de 16.0.0.0.

Tabla de encaminamiento de R1

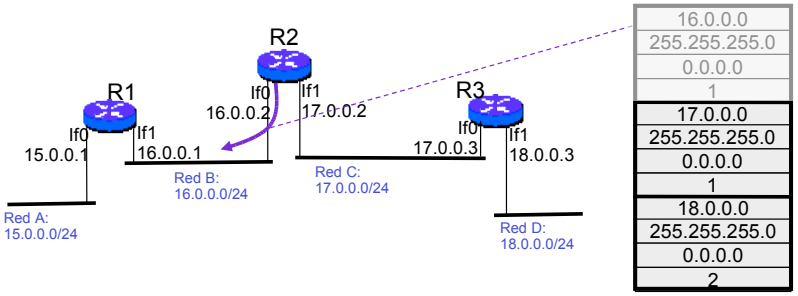
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |



# Ejemplo (III)

- R1 actualiza su tabla de encaminamiento con la información procedente del mensaje recibido de R2:
  - Coste 2 para la red 17.0.0.0/24
  - Coste 3 para la red 18.0.0.0/24

Tabla de encaminamiento de R1

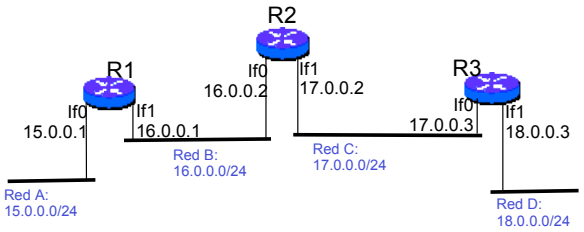
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |








# Ejemplo (IV)

- Tras aprender nuevas rutas, R1 envía un mensaje RESPONSE por *triggered update* (o cuando expire su *timer* periódico, si está próximo a hacerlo):

Tabla de encaminamiento de R1

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

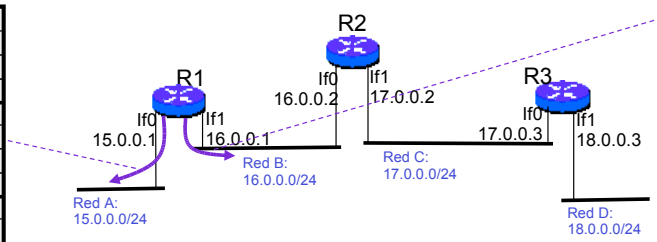
Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |

| 16.0.0.0 |
|---|
| 255.255.255.0 |
| 0.0.0.0 |
| 1 |
| 17.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 2 |
| 18.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 3 |




| 15.0.0.0 |
|---|
| 255.255.255.0 |
| 0.0.0.0 |
| 1 |
| 17.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 2 |
| 18.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 3 |

# Ejemplo (V)

- R2 actualiza su tabla de encaminamiento con el mensaje recibido de R1:
  - Coste 2 para la red 15.0.0.0/24

Tabla de encaminamiento de R1

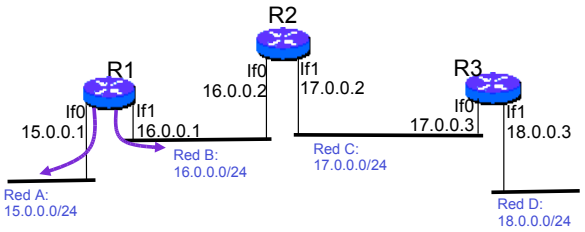
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| 15.0.0.0 | 255.255.255.0 | 16.0.0.1 | If0 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |








# Ejemplo (VI)

- R2 envía su vector de distancias en mensaje RESPONSE a todos sus vecinos (por *triggered update* o cuando venza el *timer* si está próximo a hacerlo):

Tabla de encaminamiento de R1

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

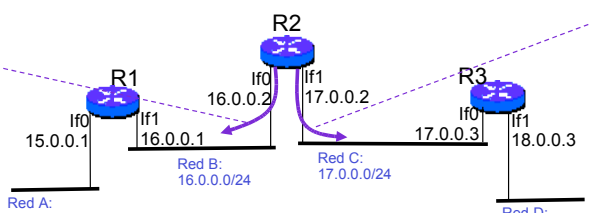
Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| 15.0.0.0 | 255.255.255.0 | 16.0.0.1 | If0 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |

| 17.0.0.0 |
|---|
| 255.255.255.0 |
| 0.0.0.0 |
| 1 |
| 18.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 2 |
| 15.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 2 |




| 16.0.0.0 |
|---|
| 255.255.255.0 |
| 0.0.0.0 |
| 1 |
| 18.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 2 |
| 15.0.0.0 |
| 255.255.255.0 |
| 0.0.0.0 |
| 2 |

# Ejemplo (VII)

- R3 actualiza su tabla de encaminamiento con el mensaje recibido de R2:
  - Coste 3 para la red 15.0.0.0/24
- R1 no recibe información que le haga actualizar su tabla

Tabla de encaminamiento de R1

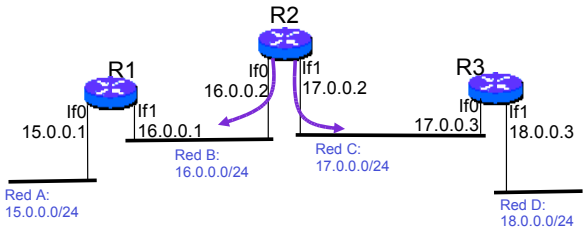
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| 15.0.0.0 | 255.255.255.0 | 16.0.0.1 | If0 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |
| 15.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 3 |








# Contenidos

# Ejemplo de Cuenta al Infinito

- Cuando R1 deja de estar accesible (p.ej. se apaga) (1), deja de enviar su vector periódicamente.
- Pasados 180 seg, no se ha refrescado la ruta hacia 15.0.0.0/24 en R2 y R2 la borrará (2). Sin embargo, hasta ese momento R2 habrá estado enviando en sus mensajes periódicos la ruta a esa red y en particular R3 la tendrá en su tabla de encaminamiento con coste 3.
- Cuando R2 reciba el siguiente mensaje periódico que envía R3 con la ruta 15.0.0.0/24 y coste 3 (3), R2 introducirá de nuevo esa ruta en su tabla (4), ahora con coste 4 y a través de R3.
- Se crea un bucle entre R2 y R3 para la ruta 15.0.0.0/24 y el coste se irá incrementando hasta 16 ($\infty$).
  - Cuando el coste de esa ruta en R2 llegue a 16 ($\infty$), la borrará y dejará de anunciarla.
  - R3 anuncia cada 30" esa ruta con coste 15, pero a los 180 segundos (6 periodos) la borrará y dejará de anunciarla.

Tabla de encaminamiento de R1

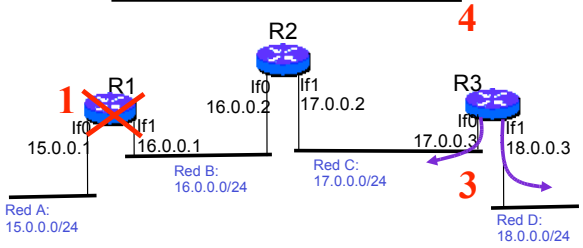
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| ~~15.0.0.0~~ | ~~255.255.255.0~~ | ~~16.0.0.1~~ | ~~If0~~ | ~~2~~ |
| 15.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 4 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |
| 15.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 3 |








# *Split Horizon*

- Cuando en RIP está activado *Split Horizon*, **por una interfaz NO se anuncian las rutas que se han aprendido por ella**.
- Esto soluciona en algunos casos el problema de la cuenta al infinito
- En el ejemplo R3 no anuncia a R2 la ruta hacia 15.0.0.0/24
  - A los 180" R3 borrará la entrada para esa ruta
- Para eliminar una ruta de las tablas de encaminamiento una cadena de *n routers* serán necesarios *n x 180"*.

Tabla de encaminamiento de R1

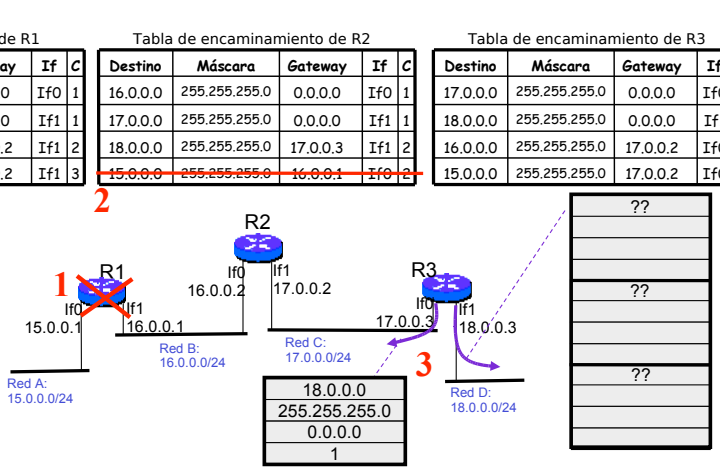
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| ~~15.0.0.0~~ | ~~255.255.255.0~~ | ~~16.0.0.1~~ | ~~If0~~ | ~~2~~ |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |
| 15.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 3 |



# Split Horizon + Poison Reverse (I)

- Cuando en RIP está activado *Split Horizon + Poison Reverse*, por una interfaz SÍ se anuncian las rutas que se han aprendido por ella, **pero con coste 16 (infinito)**.
  - R3 anuncia a R2 las rutas hacia 15.0.0.0/24 y 16.0.0.0/24 con coste 16

Tabla de encaminamiento de R1

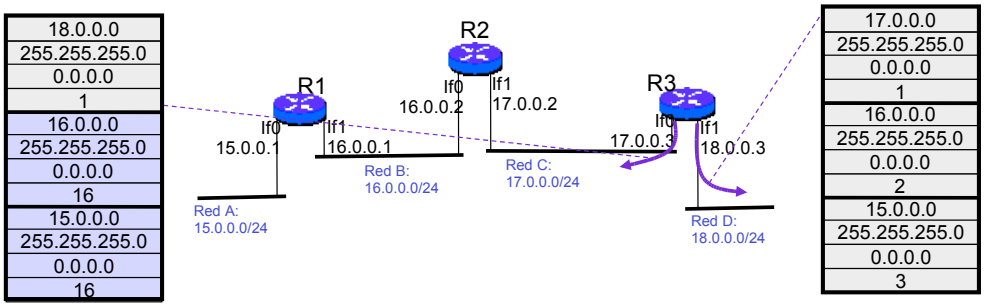
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| 15.0.0.0 | 255.255.255.0 | 16.0.0.1 | If0 | 2 |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |
| 15.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 3 |








# Split Horizon + Poison Reverse (II)

Si un *router* recibe anuncio de ruta con coste infinito, **y él no tiene esa ruta**, contesta con anuncio de ruta con coste infinito:

- Se cae R1 (1)
- Tras 180 segs. R2 elimina la ruta hacia 15.0.0.0/24 (2)
- R3 sigue anunciando la red 15.0.0.0/24 con coste 16 (3)
- R2 al recibir anuncio de R3 hacia 15.0.0.0/24 con coste 16, le contesta a R3, con ruta hacia 15.0.0.0/24 de coste 16 (4)
- Esto provoca que R3 borre de su tabla la ruta hacia 15.0.0.0/24 (5), ya que la había aprendido a través de R2, y deje de anunciarla a otros.

Tabla de encaminamiento de R1

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

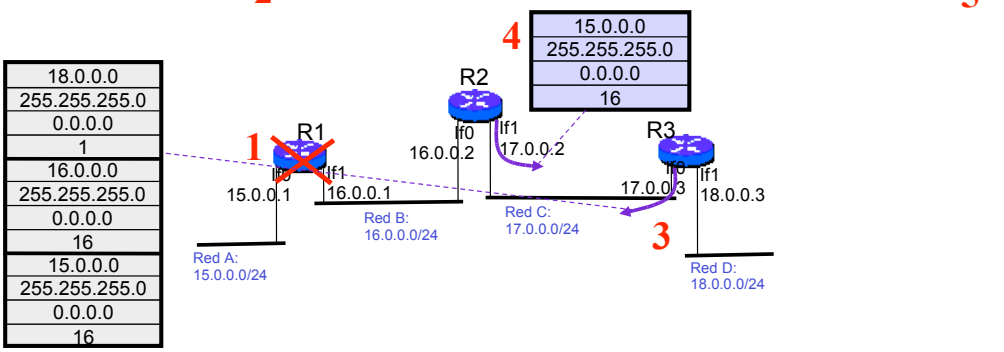
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| ~~15.0.0.0~~ | ~~255.255.255.0~~ | ~~16.0.0.1~~ | ~~If0~~ | ~~2~~ |

2

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |
| ~~15.0.0.0~~ | ~~255.255.255.0~~ | ~~17.0.0.2~~ | ~~If0~~ | ~~3~~ |

5



# *Split Horizon* vs *Split Horizon + Poison Reverse*

- *Split Horizon + Poison Reverse* mejora el tiempo que se tarda en eliminar una ruta de las tablas de encaminamiento de una cadena de *n routers*:
  - **Sin** *Poison Reverse*, por cada *router* es necesario esperar 180" para que éste borre esa entrada de su tabla de encaminamiento. Para *n routers* en cascada, el borrado de esa ruta necesitaría aproximadamente:
    $$n \times 180''$$
  - **Con** *Poison Reverse*, sólo es necesario esperar 180" para que el primer *router* borre dicha entrada de su tabla. A continuación, cuando este *router* reciba un mensaje RIP periódico con dicha ruta y coste 16 (después de 30" como máximo), provocará el envío de un mensaje de actualización explícita de borrado de ruta (coste 16) que borrará dicha ruta en el siguiente *router*. Y así sucesivamente. Para *n routers* en cascada, en el caso peor, el borrado de esa ruta necesiaría aproximadamente:
    $$180'' + (n-1) \times 30'' + (n-1) \times \delta$$
    donde $\delta$ representa el tiempo necesario para la propagación en cadena del mensaje de borrado por los *n-1 routers*.
- En *Split Horizon + Poison Reverse* los mensajes de anuncio de rutas son más largos ya que siempre se anuncian las rutas por la misma interfaz por la que se han aprendido, con coste 16.





# *Triggered Update* por ruta obsoleta y *Garbage Collect Timer* (I)

- Es una técnica que consigue efectos parecidos a *Split Horizon + Poison Reverse*: sólo se anuncian algunas rutas con coste 16 durante y sólo durante un cierto intervalo de tiempo.
- Cuando un *router* tiene que eliminar una ruta de su tabla de encaminamiento, emite un *Triggered Update* con un anuncio de esa ruta con coste 16 (infinito) por todas las interfaces salvo por donde la había aprendido.
- El *router*, en vez de eliminar la ruta de su **tabla RIP**, todavía la mantiene en ella durante un determinado tiempo (*Garbage Collect Timer*, por defecto 120 segundos) pero con coste 16 para indicar que esa ruta es inalcanzable. Por tanto, esa ruta viajará en los mensajes RESPONSE que emita dicho *router* mientras ésta permanezca en la tabla RIP.
- Transcurrido el tiempo dado por el *Garbage Collect Timer*, la ruta se elimina definitivamente de la tabla RIP y el *router* dejará de anunciarla.
- Algunas implementaciones de RIP (ej.: Zebra) usan esta técnica en vez de *Split Horizon + Poison Reverse*

# *Triggered Update* por ruta obsoleta y *Garbage Collect Timer* (II)

- Se cae R1 (**1**)
- R2, tras 180 segs, debe borrar la ruta de su tabla de encaminamiento (**2**). Emite un anuncio por If1 (**3**) con ruta hacia 15.0.0.0/24 con coste 16.
- R2 mantiene en su tabla RIP la ruta hacia 15.0.0.0/24 con coste 16, y la anuncia en sus RESPONSE periódicos con ese coste.
- Transcurrido el *Garbage Collect Timer* (120 segs), R2 elimina la ruta de su tabla RIP.

Tabla de encaminamiento de R1

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

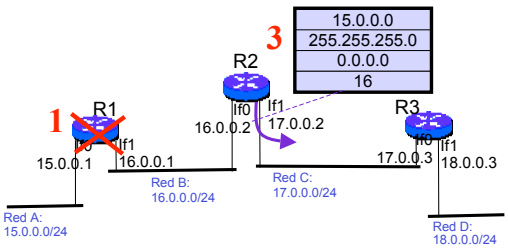
Tabla de encaminamiento de R2

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| ~~15.0.0.0~~ | ~~255.255.255.0~~ | ~~16.0.0.1~~ | ~~If0~~ | ~~2~~ |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |
| 15.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 3 |

**2** *(en la tabla de RIP de R2 se mantiene esa ruta con coste 16 durante un intervalo de tiempo dado por Garbage Collect Timer)*








# *Triggered Update* por ruta obsoleta y *Garbage Collect Timer* (III)

- Los *routers* vecinos que reciban un anuncio de ruta con coste 16 y tengan anotada esa ruta a través del *router* origen de ese anuncio, borrarán esa ruta, y darán lugar a su vez a un *Triggered Update* por ruta obsoleta y arrancarán el *Garbage Collect Timer*
  - En el ejemplo: R3 borra la ruta hacia 15.0.0.0/24 (**4**) porque la tiene anotada a través de R2. R3 emitirá entonces, a su vez, un anuncio por If1 con ruta hacia 15.0.0.0/24 con coste 16 (**5**).

Tabla de encaminamiento de R1

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 15.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 17.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 2 |
| 18.0.0.0 | 255.255.255.0 | 16.0.0.2 | If1 | 3 |

Tabla de encaminamiento de R2

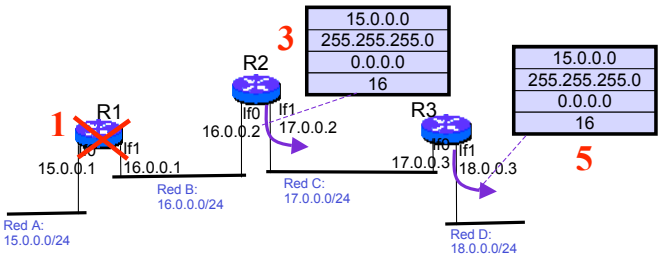
| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 16.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 18.0.0.0 | 255.255.255.0 | 17.0.0.3 | If1 | 2 |
| ~~15.0.0.0~~ | ~~255.255.255.0~~ | ~~16.0.0.1~~ | ~~If0~~ | ~~2~~ |

Tabla de encaminamiento de R3

| Destino | Máscara | Gateway | If | C |
|---|---|---|---|---|
| 17.0.0.0 | 255.255.255.0 | 0.0.0.0 | If0 | 1 |
| 18.0.0.0 | 255.255.255.0 | 0.0.0.0 | If1 | 1 |
| 16.0.0.0 | 255.255.255.0 | 17.0.0.2 | If0 | 2 |
| ~~15.0.0.0~~ | ~~255.255.255.0~~ | ~~17.0.0.2~~ | ~~If0~~ | ~~3~~ |

**2** *(en la tabla de RIP de R2 se mantiene esa ruta con coste 16 durante un intervalo de tiempo dado por Garbage Collect Timer)*

*(en la tabla de RIP de R3 se mantiene esa ruta con coste 16 durante un intervalo de tiempo dado por Garbage Collect Timer)* **4**



# *Split Horizon + Poison Reverse* vs *Triggered Update + Garbage Collect Timer*

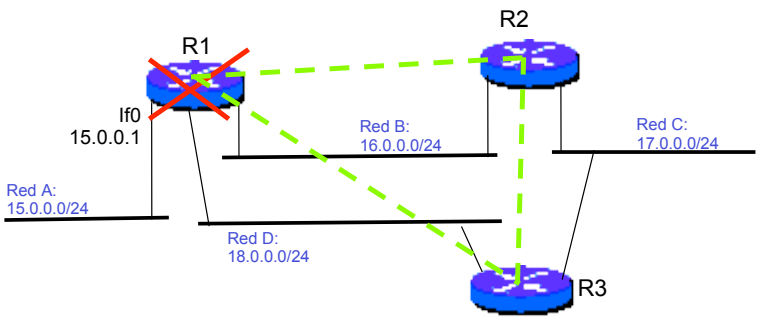
- *Triggered Update + Garbage Collect Timer* mejora el tiempo que se tarda en eliminar una ruta de las tablas de encaminamiento de una cadena de *n routers* en cascada:
  - *Split Horizon + Poison Reverse* en el caso peor tardaría:
    $$180'' + (n-1) \times 30'' + (n-1) \times \delta$$
  - Con *Triggered Update + Garbage Collect Timer* sólo es necesario esperar 180" para que el primer *router* borre dicha entrada de su tabla. A continuación, este *router* enviará un mensaje de actualización explícita con dicha ruta y coste 16, provocando el borrado de la ruta en el siguiente *router*. Y así sucesivamente. En el caso peor se necesitaría aproximadamente:
    $$180'' + (n-1) \times \delta$$
    donde $\delta$ representa el tiempo necesario para la propagación en cadena del mensaje de borrado por los *n-1 routers*.
- Los mensajes con *Triggered Update + Garbage Collect Timer* son más cortos que con *Split Horizon + Poison Reverse* ya que sólo se anuncian rutas con coste 16 en el momento en el que se eliminan de la tabla de encaminamiento de un *router* y sólo durante un intervalo de tiempo, dado por *garbage collect timer*.





# *Hold-down Timers*

- Es otra técnica utilizada por algunos fabricantes (Cisco) junto a *Poison Reverse* (no forma parte de la especificación de RIP, RFC 2453).
  - Soluciona el problema de cuenta al infinito con 3 encaminadores en triángulo.
- Cuando una ruta queda obsoleta en una tabla de encaminamiento de un *router*, este arranca un *Hold-down Timer*, por defecto 120".
  - Ej. R2, tras 180" sin recibir vector de R1 borra ruta hacia 15.0.0.0/24 y arranca el *Hold-down Timer*.
- Mientras dura el *Hold-Down Timer*, el *router* ignora posibles actualizaciones que se reciban relativas a esa ruta, a no ser que procedan del *router* a través del que se alcanzaba la ruta eliminada.
  - Ej. R2 no aprende la ruta con coste 2 a 15.0.0.0/24 que le anuncia R3 a través de la red C.
- Si el bucle no es triángulo, sino con más nodos, el temporizador deberá ser más grande.
  - Normalmente es de 120"
- El inconveniente de esta técnica es que, al utilizarla, se tarda más tiempo en aprender las nuevas rutas alternativas legítimas que podría haber hacia las redes cuya entrada queda obsoleta:
  - Ej. Se estropea en enlace de R1 con D, sólo ese
  - R3 tras 180 segs. declara obsoleta la ruta hacia 15.0.0.0/24 a través de R1
  - R3 tarda 120 segs. en aprender la nueva ruta legítima hacia la 15.0.0.0 a través de R2 y R1. Sin *Hold Down Timers* la aprendería al recibir el primer RESPONSE de R2.



# Contenidos







# Referencias

- Charles M. Kozierok, **TCP/IP GUIDE. A Comprehensive, Illustrated Internet Protocols Reference**, No Starch Press, 2005.
- RFC2453, **RIP versión 2**: `http://www.faqs.org/rfcs/rfc2453.html`