

# Enrutamiento Distance-Vector: RIP

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

Laboratorio de Programación de Redes  
3º Ingeniería Técnica en Informática de Gestión

# Objetivos

- Descripción detallada de un protocolo DV
- Ver los principales problemas de estos protocolos con ejemplos claros
- Analizar las posibles soluciones

# Contenido

## RIP

- Características
- Formato
- Funcionamiento
- Cuenta a infinito
  - Situaciones y soluciones
- RIPv2

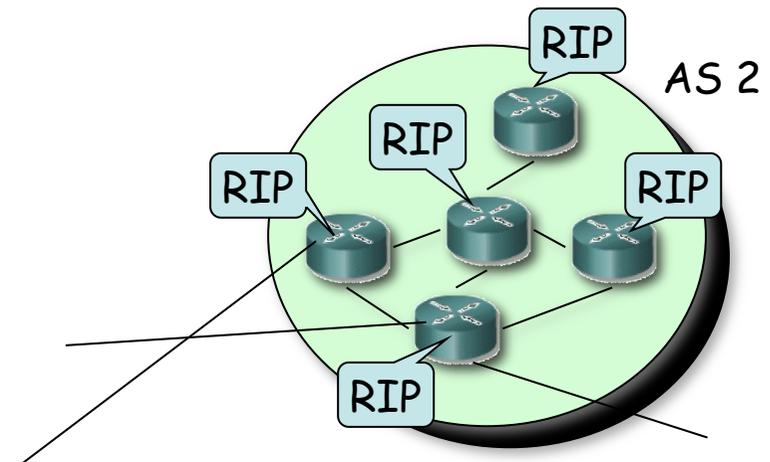
# Distance Vector

- Cada nodo tiene unas distancias estimadas a cada destino (vector de distancias)
- Se las envía a todos sus vecinos periódicamente
- Algoritmo de Bellman-Ford distribuido
- No necesitan conocer la topología completa de la red
- Usado en la ARPANET hasta 1979
- Ejemplos: RIP, Xerox XNS RIP, IPX RIP, Cisco IGRP, DEC's DNA Phase IV, Apple's RTMP

# RIP

## Características

- Routing Information Protocol
- Distance Vector
- IGP
- RFCs 1058 (v1), 2453 (v2)
- routed en Unix BSD
- Emplea UDP
- Métrica:
  - Número de saltos
  - 16 =  $\infty$
- Se envía el vector de distancias cada 30 segs
- Cambios en la topología:
  - Ruta a red N por router G
  - Si no recibimos vector de G en 180segs marcar como inválida ( $\infty$ )
- No escala para redes grandes
- Para redes con enlaces homogéneos
- Simple
- Malos tiempos de convergencia

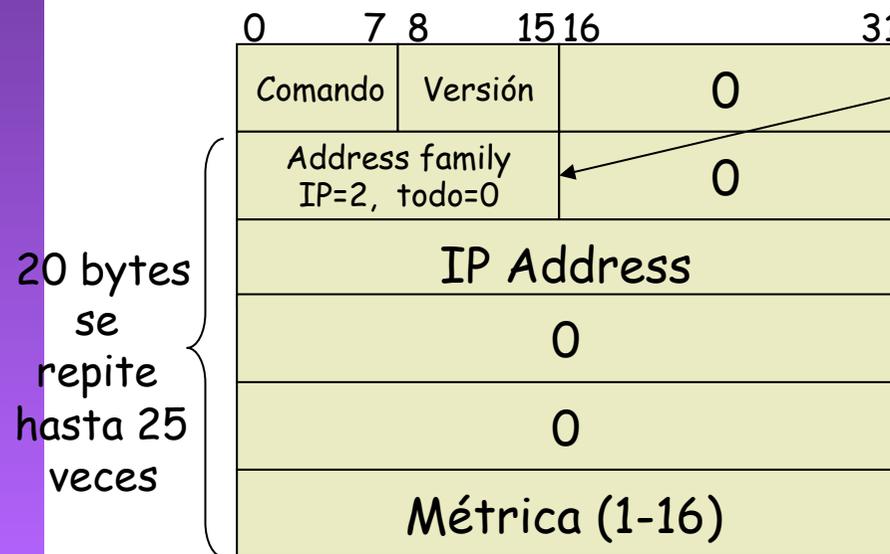


# RIP

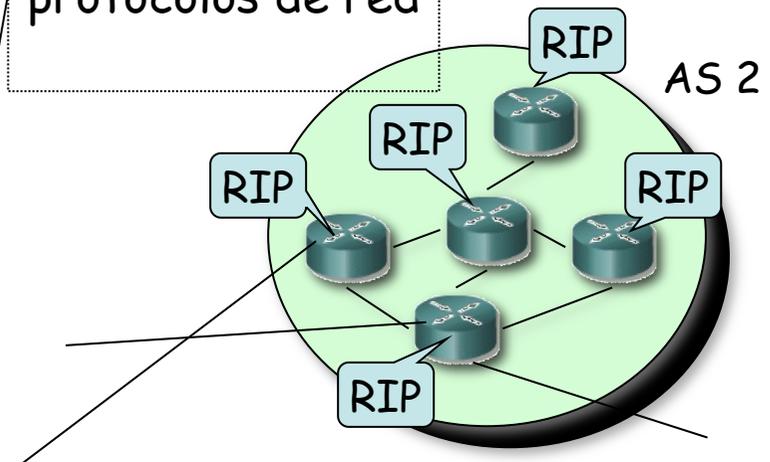
## Formato

### Tipos de PDUs:

- *Request*
  - Comando=1
  - Se puede pedir el coste a unos destinos o a todos
- *Response*
  - Comando=2
  - El *next-hop* es la IP que envía la PDU
  - Periódico o en respuesta a un *request*



Permitiría otros protocolos de red



# RIP

## Funcionamiento

### Inicialización

- Manda un *request* especial por cada interfaz
- IP destino *broadcast*

### Recibe un *request*

- Si es de inicialización manda todo el vector
- Si no responde con los valores solicitados

### Periódicamente

- Timer 30seg (de 25 a 35)
- Manda un *response* con todo el vector por cada interfaz
- IP destino broadcast

### Recibe *response*

- **Actualiza** su vector y tabla de rutas
- Si la tiene reinicializa timer

### Caduca timer de una ruta

- Timer de 180s para cada una
- Pasa a coste  $\infty$
- Inicia timer para borrarla

### Timer de borrado

- Timer de 120s para una ruta invalidada

# RIP

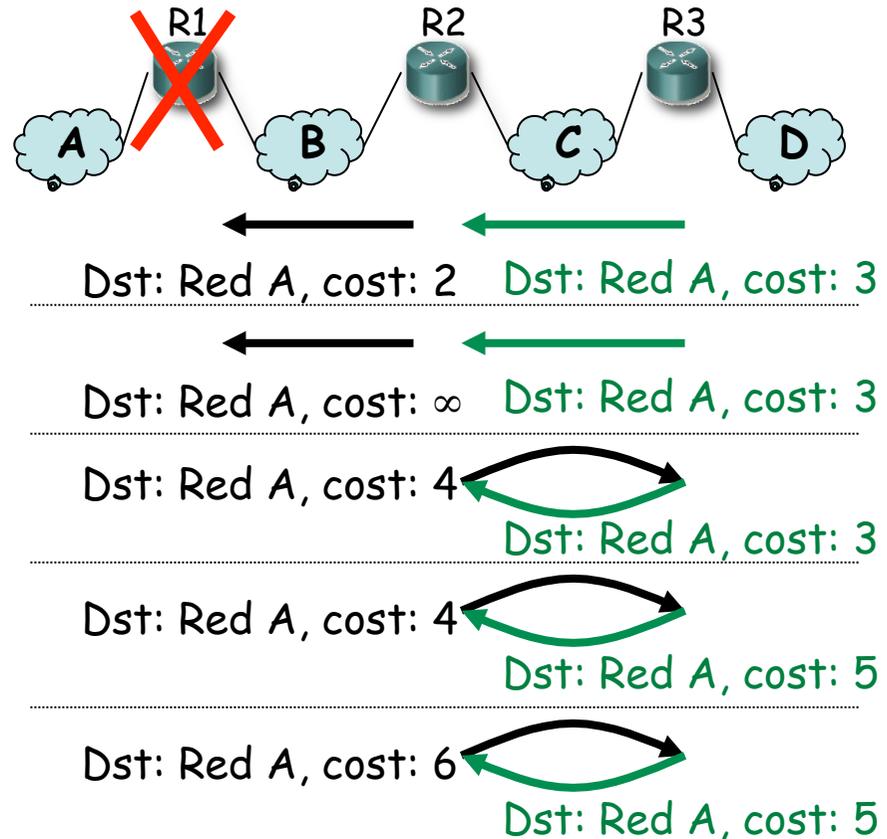
## Actualización

1. Añadir 1 a la métrica de cada destino anunciado en el paquete de RIP recibido
2. Para cada entrada en el paquete
  - Si el destino no está en la tabla de rutas
    1. Añadirlo
  - Si no (sí está en la tabla)
    1. Si el siguiente salto en la tabla es el mismo que quien ha mandado el paquete de IP
      - Sustituir el coste por el nuevo
    2. Si no (diferente *next-hop*)
      - Si la coste es menor que el de la tabla
        - o Sustituir el coste y el *next-hop*

# RIP

## 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 enviá 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 !



# Contenido

## RIP

- Características
- Formato
- Funcionamiento
- **Cuenta a infinito**
  - **Situaciones y soluciones**
- RIPv2

# RIP

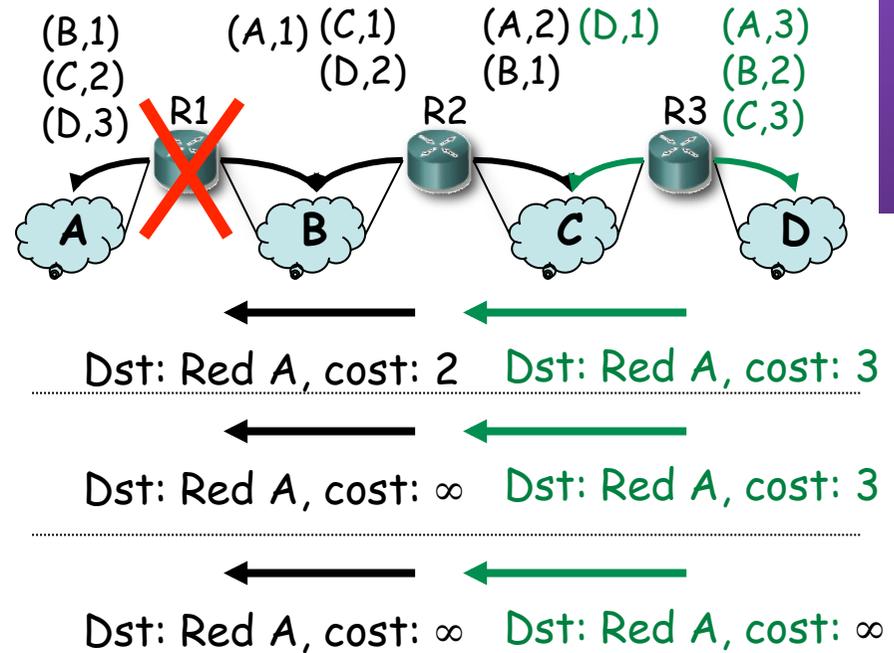
## Evitar las cuentas a infinito

### Split horizon

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

Ejemplo (... ..):

- Caduca timer (180s) en R2 (...)
- Caduca timer (180s) en R3 (...)



# RIP

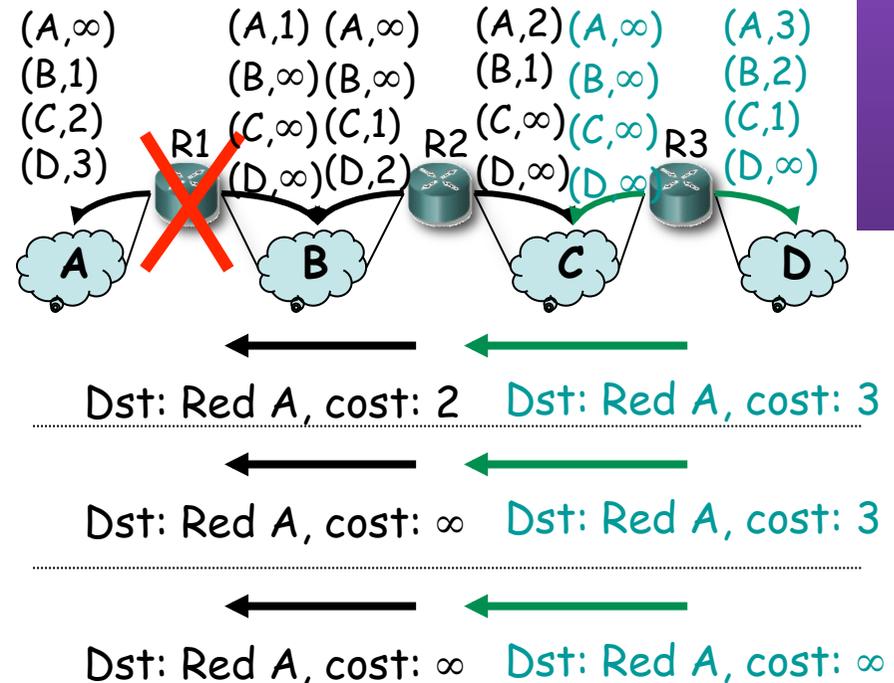
## Evitar las cuentas a infinito

### 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

Ejemplo (... ..):

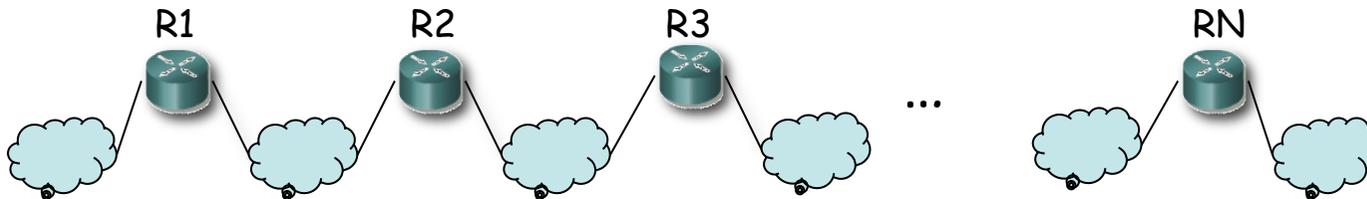
- Caduca timer (180s) en R2 (...)
- Caduca timer (30s) en R2, envía vector (...)



# RIP

## *Bad news travel slowly*

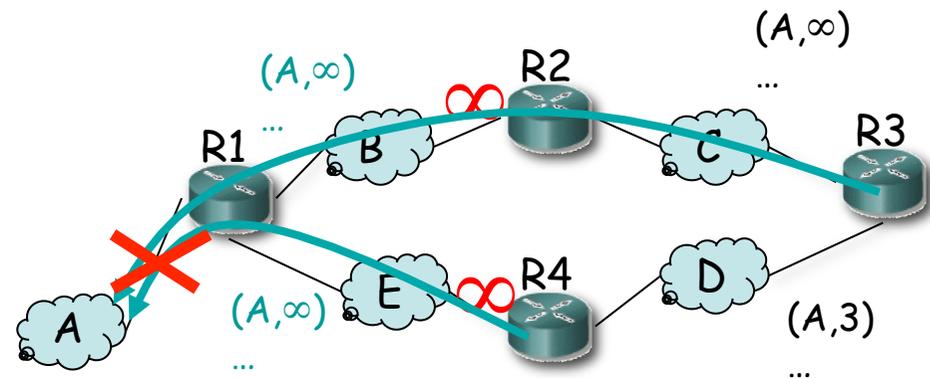
- Convergencia lenta
- Ejemplos:
  - Actualización de información
    - Caso peor  $N \times 30\text{seg}$  para llegar al otro extremo
  - Pérdida de ruta
    - Caso peor  $N \times 180\text{seg}$  hasta el otro extremo
- ¿ Mejorar estos tiempos ?
  - **Triggered updates**: Enviar el vector en cuanto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



# RIP

## Cuenta a infinito

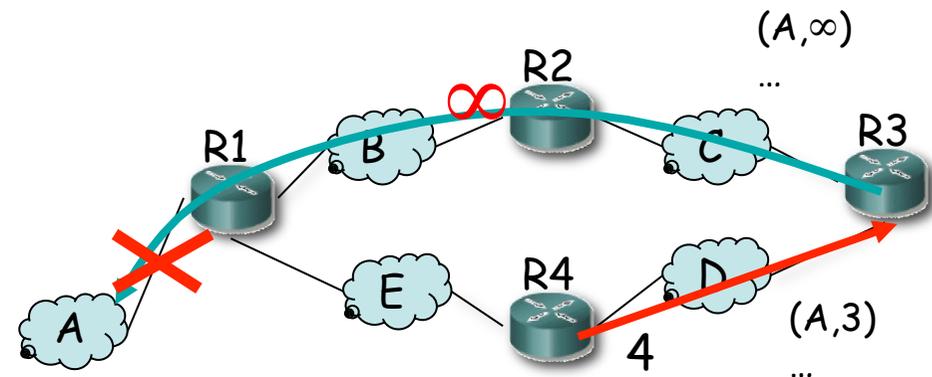
- Supongamos la topología de la figura
- Usan *split horizon with poisoned reverse*
- Las flechas son las rutas hacia la Red A (...)
- Supongamos que falla el interfaz de R1 en la Red A (...)
- R1 anuncia coste  $\infty$  a R2 y R4 (...)
- Puede que antes de que avisen a R3 él envíe su actualización 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 (...)



# RIP

## Cuenta a infinito

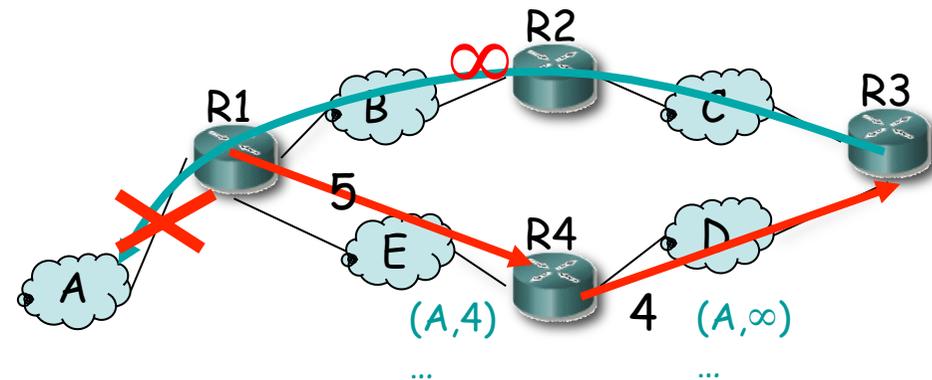
- Supongamos la topología de la figura
- Usan *split horizon with poisoned reverse*
- Las flechas son las rutas hacia la Red A (...)
- Supongamos que falla el interfaz de R1 en la Red A (...)
- R1 anuncia coste  $\infty$  a R2 y R4 (...)
- Puede que antes de que avisen a R3 él envíe su actualización 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 (...)



# RIP

## Cuenta a infinito

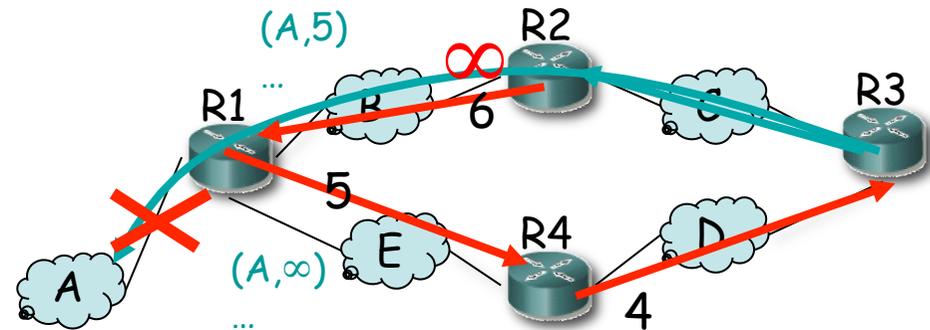
- Supongamos la topología de la figura
- Usan *split horizon with poisoned reverse*
- Las flechas son las rutas hacia la Red A (...)
- Supongamos que falla el interfaz de R1 en la Red A (...)
- R1 anuncia coste  $\infty$  a R2 y R4 (...)
- Puede que antes de que avisen a R3 él envíe su actualización 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 (...)



# RIP

## Cuenta 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 creará que se llega por R4 con coste 5 (...)
- R1 lo anunciará a R2 (...)
- R2 creará que se llega por R1 (...)
- Y luego R2 hasta llegar a R3 (...)

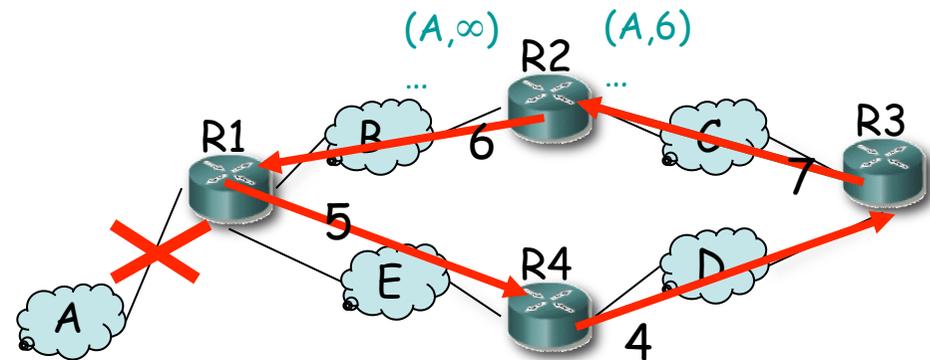


# RIP

## Cuenta 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 !



# RIP

## Cuenta a infinito

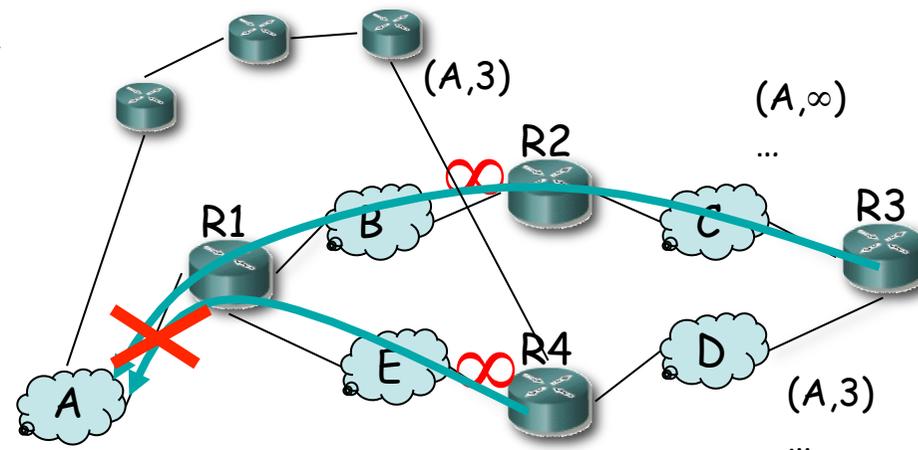
### Solución

- *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

## Otros problemas

- Anuncia una ruta con la dirección de la red
  - ¡ Solo sirve para redes *classful* !
  - También para subredes clásicas (*subnetting*) ¿Cómo?
    - Por diseño
    - Manteniendo la misma máscara en toda la red
  - Para soportar VLSM o CIDR necesita anunciar la máscara también
- Para redes pequeñas
  - $16 = \infty$
  - Malos tiempos de convergencia (cuentas a infinito)

# Contenido

## RIP

- Características
- Formato
- Funcionamiento
- Cuenta a infinito
  - Situaciones y soluciones
- **RIPv2**

# RIPv2

## Route Tag

- Asignado a la ruta
- 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

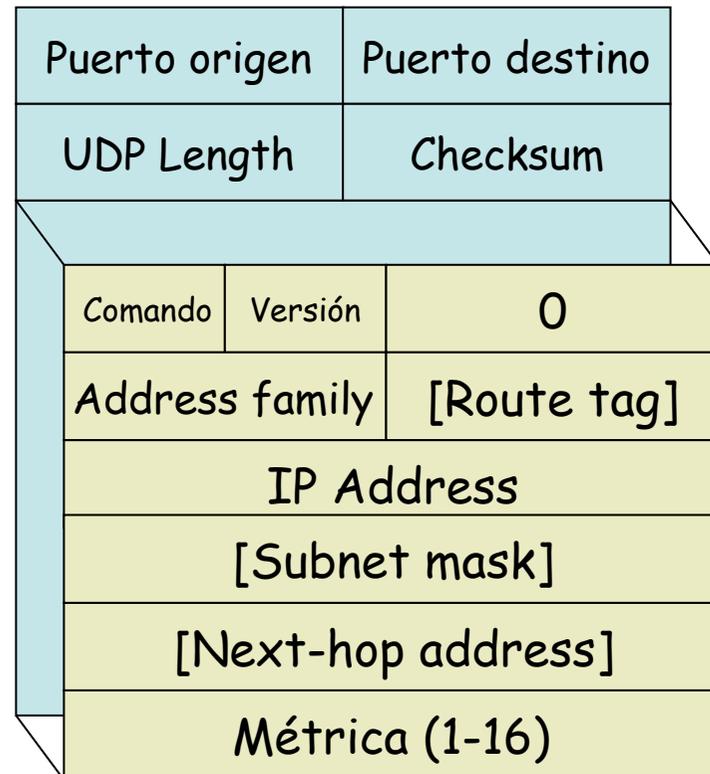
## Autenticación

- Primera entrada *family* = 0xFFFF
- *Route tag* = tipo (2 ó 3)
  - 2 : password (texto plano en el resto)
  - 3 : MD5

0	7	8	15	16	31
Comando		Versión		0	
Address family				Route tag	
IP Address					
Subnet mask					
Next-hop address					
Métrica (1-16)					

# RIP y UDP

- 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.)



# Resumen

- Protocolo DV simple
- Presenta problemas de convergencia:  
cuentas a infinito
- Las soluciones
  - *Split horizon*
  - *Poisoned reverse*
  - *Triggered updates*
  - *Hold down interval*
  - Añaden complejidad
  - No resuelven perfectamente el problema