# Tema 3.2

# Algoritmos vector-distancia: RIP (Routing Information Protocol)

# Índice

# Algoritmos vector-distancia (I)

- Idea:
  - Cada nodo almacena en su tabla de encaminamiento información para cada red (RedDestino;GW;**Distancia**).
    - RedDestino: Identificación de la red de destino (p.e. dirección IP de red).
    - GW (salto al siguiente): Nodo vecino a través del cual llega a RedDestino.
    - Distancia: Coste según una determinada métrica (por ejemplo número de saltos), que el nodo origen tiene estimado para llegar a RedDestino.
  - Periódicamente, los nodos de la red envían **a cada uno de sus nodos vecinos** mensajes de vector de distancias: un conjunto de pares (RedDestino1 ; distancia1), (RedDestino2 ; distancia2),... De la misma manera, periódicamente, cada nodo recibe de cada uno de sus vecinos un vector de distancias.
  - Cada nodo emplea esta información para aprender/actualizar los caminos más rápidos de llegar a los destinos.

# Algoritmos vector-distancia (II)

- Ejemplos de algoritmos vector-distancia:
  - RIP y RIPv2 *(Routing Information Protocol)*
  - IGRP (*Interior Gateway Routing Protocol*)
- Ventajas de este tipo de algoritmos:
  - Sencillez
  - Número de mensajes enviados por cada nodo es pequeño (= número de vecinos).
- Desventajas:
  - Convergencia lenta ante cambio en la topología.
  - Se pueden producir situaciones en que el algoritmo no converge (cadenas de fallos que provocan que partes de la red se queden aisladas). Esto obliga a añadir técnicas que intenten evitar estas posibilidades.
  - El tamaño de los mensajes crece con el número de redes de mi sistema.
- Veremos el algoritmo RIP (*Routing Information Protocol*), como ejemplo de algoritmo tipo vector-distancia.

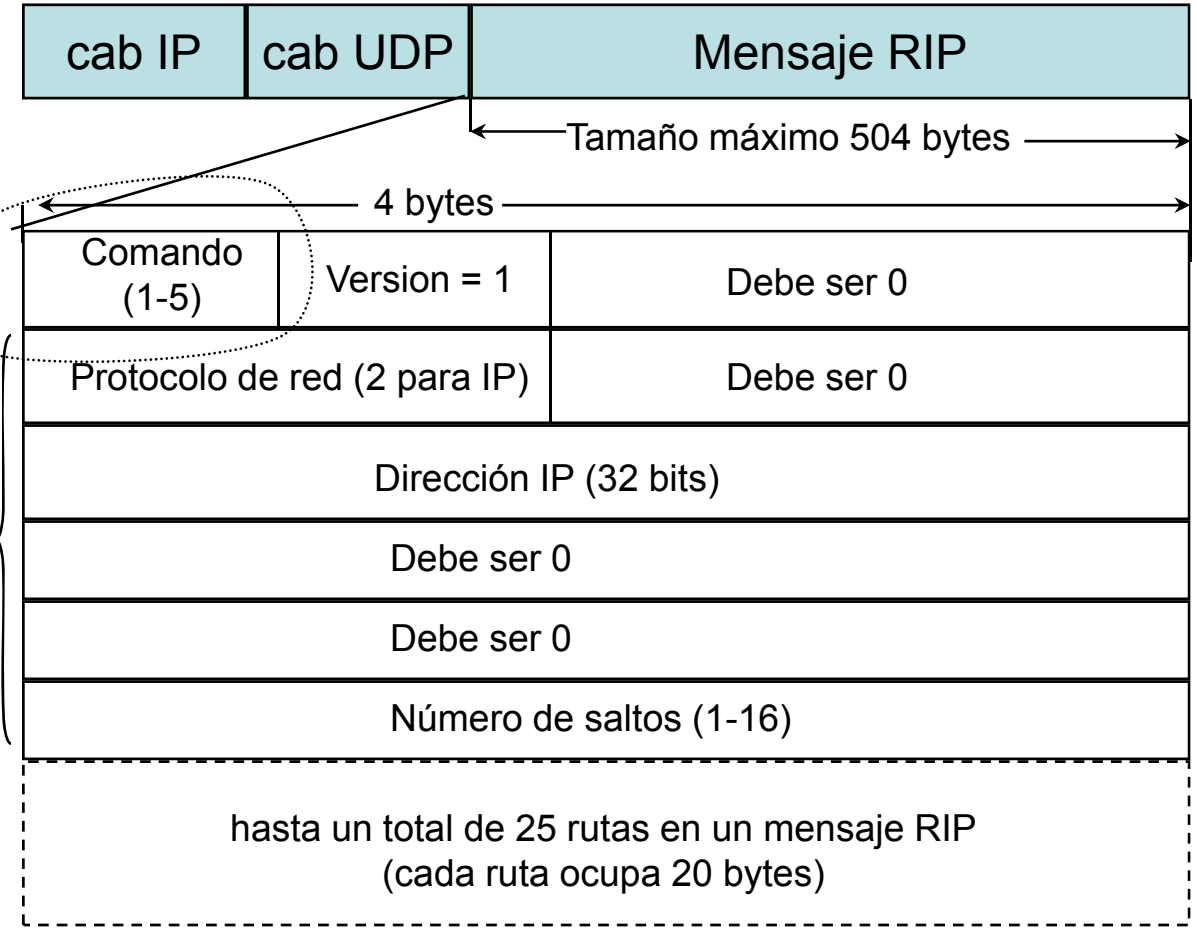
# RIP

- Protocolo de encaminamiento dinámico, que implementa un algoritmo de tipo vector-distancias, con métrica el número de saltos.
- Diseñado en la Universidad de Berkeley (California, EEUU), y distribuido inicialmente con Unix BSD, lo que catapultó su difusión.
- Versión 1 (1988) - RFC 1058, 1388.
- Versión 2 - RFC 1721...1724, 2453 (año 1998).
- Se basa en el intercambio de mensajes RIP *Request* y *Response* entre *routers* vecinos. El puerto empleado es UDP 520 (reservado para RIP).
- Nosotros describiremos el protocolo RIP v1, como ejemplo de algoritmo tipo vector-distancias.

# RIP v1 (I)

**Tipos de mensajes:**
1- request
2- response
3- traceon (no usamos)
4- traceoff (no usamos)
5- reserved

- El protocolo RIP puede ser utilizado con cualquier protocolo de nivel de red (no sólo redes IP).
- Si es necesario informar de más rutas que 25, se envían varios mensajes RIP.

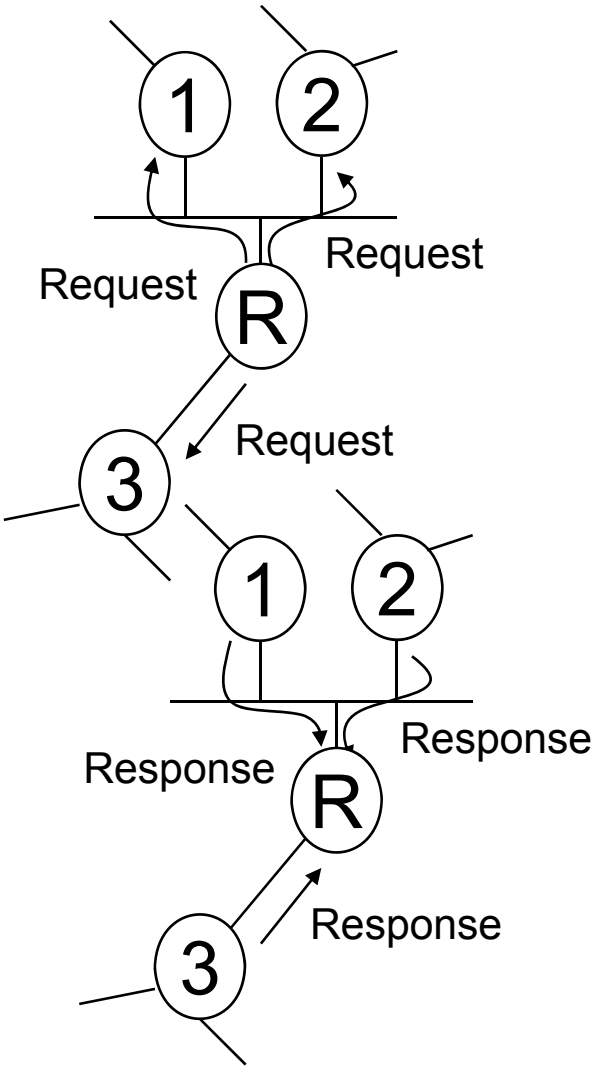


# RIP v1 (II)

- Cada nodo en la red ejecutando el protocolo RIP mantiene una tabla con los siguientes campos (los marcados con * son los incluidos en la tabla de encaminamiento).
  - * IP red destino.
  - * Máscara de red.
  - * Interfaz de salida.
  - * Salto al siguiente.
  - Número de saltos.
  - Flag de actualización reciente (para *triggered updates*).
  - Temporizadores asociados a esta ruta.

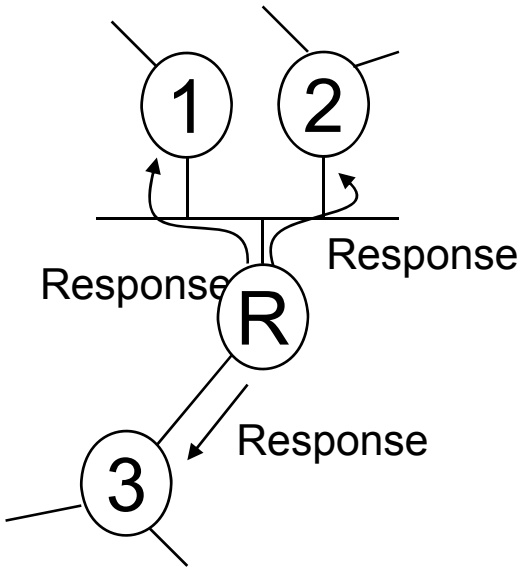
# RIP v1 (III)
## Mensajes RIP

- Mensajes *Request*. Los mensajes *request* solicitan información sobre las rutas a *routers* vecinos. Un *router* envía un mensaje *Request*:
  - Durante el arranque del protocolo, para conocer la información completa por parte de sus vecinos. En este caso se envía un mensaje *request* por cada interfaz del router, con dirección IP de destino *broadcast*.
  - En algunas situaciones en que se solicita información de actualización.
- Un *router* que recibe un mensaje *Request*:
  - Responde con un mensaje *Response* informando de sus rutas conocidas en su tabla.
  - En ocasiones los mensajes *Request* solicitan información de una ruta a una red destino específica. En ese caso el mensaje *Response* informa de esa ruta/s únicamente.



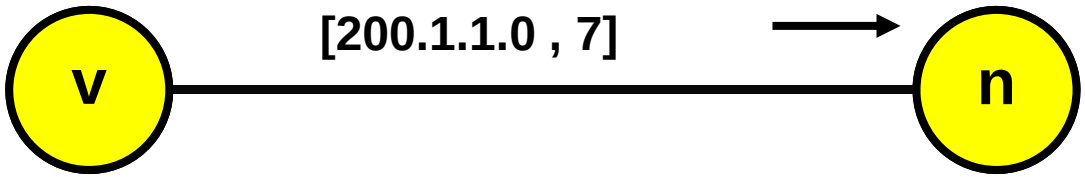
# RIP v1 (IV)
## Mensajes RIP

- Tras el arranque, los *routers*:
  - Envían periódicamente (por defecto según un temporizador de 30 segundos) mensajes *Response* informando de sus rutas conocidas y la distancia en número de saltos a las mismas.
    - Las redes directamente conectadas se anuncian como
      - Distancia 1 si están activas.
      - Distancia infinita (16 saltos) si no existe conectividad.
    - Los mensajes Response se envían por cada interfaz activa del *router*.
    - En caso de informar de más de 25 rutas, se envían más mensajes *Response*.
    - Dirección IP destino *broadcast*.
  - Procesan los mensajes *Response* recibidos de sus nodos vecinos.
  - Responden a los mensajes *Request* recibidos.



# RIP v1 (V)
## Mensajes RIP

- Cada vez que se recibe un mensaje *Response*:
  - Si la ruta (IPred_destino , dist) es nueva, se añade a la tabla de encaminamiento, con GW el del *router* vecino que ha informado del mensaje, distancia (dist+1).
  - Si la ruta no es nueva,
    - Si el mensaje *Response* proviene del nodo GW por donde yo llego a $IP_{red\_destino}$, según mi tabla actual => independientemente del número de saltos que me informen, adopto la nueva ruta.
    - Sino, si el número de saltos del que me informan indica una ruta más corta, aprendo la nueva ruta.
    - Sino, descarto la ruta.
- Si en la T.Enc., la ruta a un destino tiene distancia infinita => los datagramas con ese destino NO son encaminados.




**v** dice: yo llego a la red 200.1.1.0 en 7 saltos.

# RIP v1 (VI)

## Problema de "contar hasta infinito"

Tabla Enc. (A)

| Dest | GW | Dist |
|---|---|---|
| C | B | 2 |

Tabla Enc. (B)

| Dest | GW | Dist |
|---|---|---|
| C | C | 1 |

El enlace B-C se pierde

| | | |
|---|---|---|
| C | B | 2 |

| | | |
|---|---|---|
| C | - | ∞ |

C 2 → ← C ∞

| | | |
|---|---|---|
| C | - | ∞ |

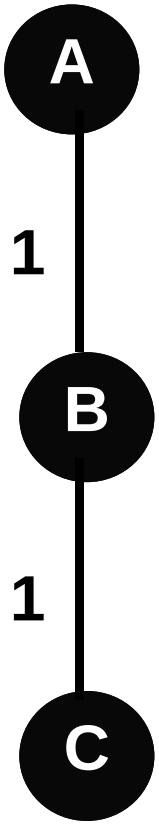
| | | |
|---|---|---|
| C | A | 3 |

C ∞ → ← C 3

| | | |
|---|---|---|
| C | B | 4 |

| | | |
|---|---|---|
| C | - | ∞ |

C 4 → ← C ∞



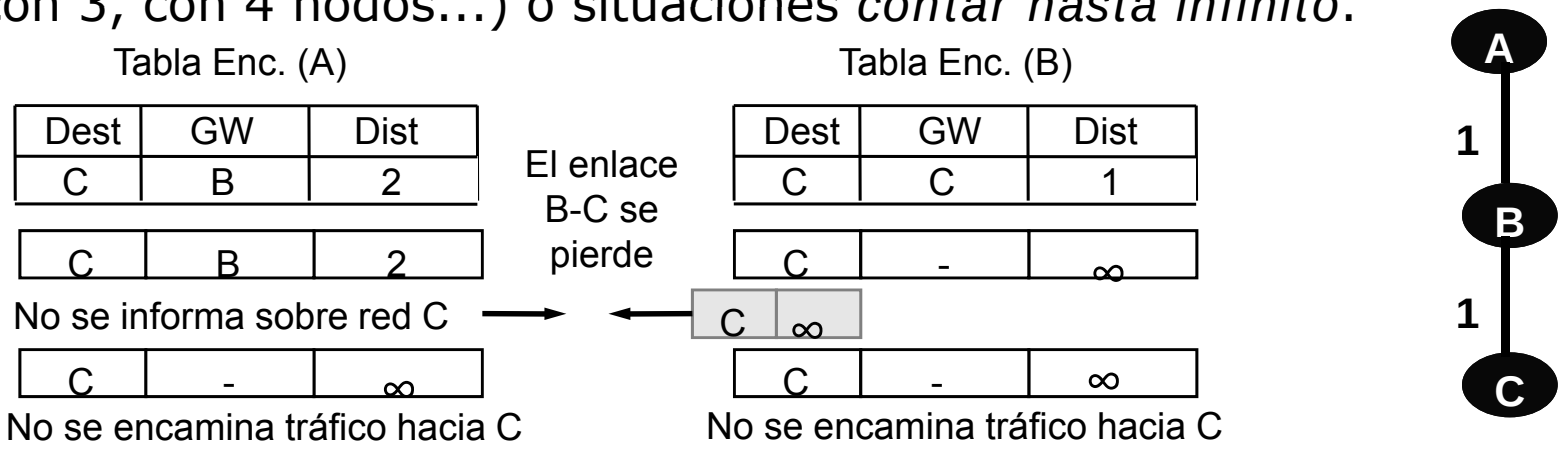

..... (así hasta que ambos lleguen a dist=∞)

# RIP v1 (VII)
## Problema de "contar hasta infinito"

- El problema de contar hasta infinito tiene como consecuencia que, en determinados escenarios, la convergencia de las tablas de encaminamiento ante determinados cambios en la topología, es lenta.
- Se elige un número "infinito" pequeño: en RIP 16 saltos equivale a distancia infinita.
- Sin embargo, la convergencia lenta y la indeterminación de los mensajes periódicos puede provocar bucles en el encaminamiento, como muestra el siguiente ejemplo.

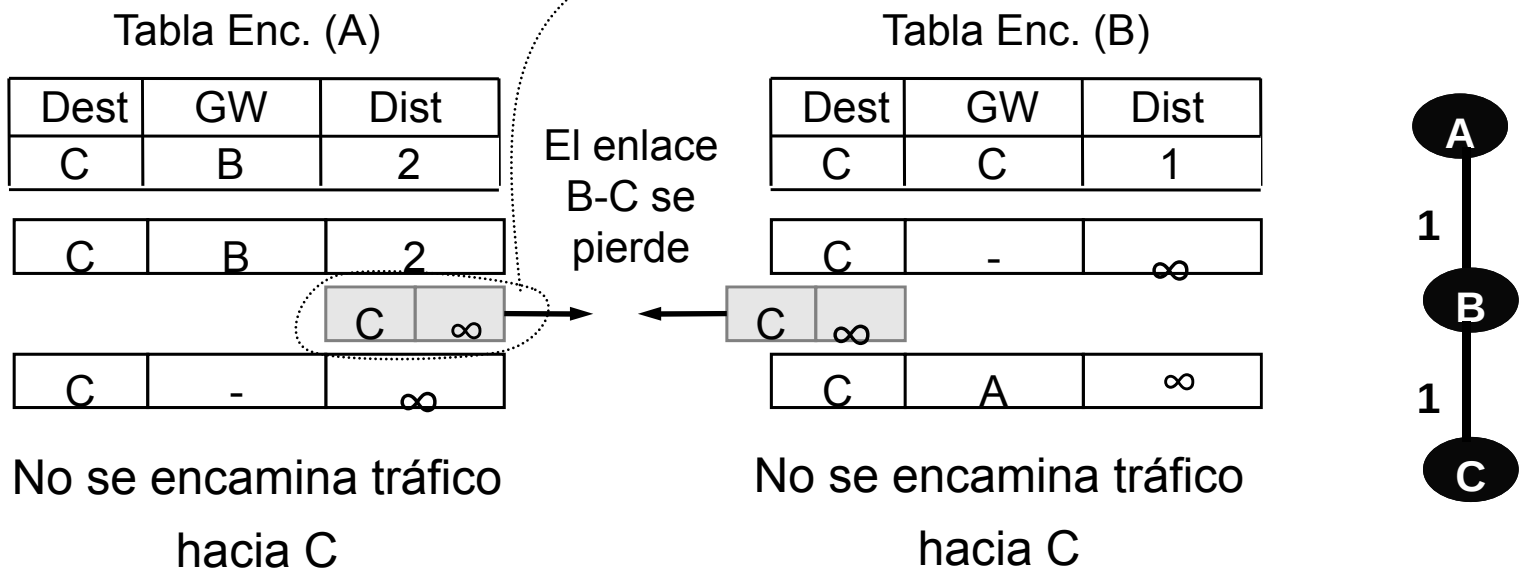
# RIP v1 (VIII)
## Horizonte dividido

- Problemas que surgen:
  - A cree que puede llegar a red R a través de B.
  - B cree que puede llegar a red R a través de A.
- Para evitar este tipo de engaños, se aplica la técnica de *horizonte dividido* (*split horizon*).
  - Cuando se envía un mensaje *Response* por una interfaz I, se filtran (no se transmiten) aquellas entradas que han sido aprendidas desde un vecino directamente conectado a la red en I.
- El *split horizon* previene los ciclos de tamaño dos nodos, y algunos escenarios del problema de *contar hasta infinito*. Sin embargo, no soluciona todos los escenarios en que se pueden producir ciclos (con 3, con 4 nodos...) o situaciones *contar hasta infinito*.



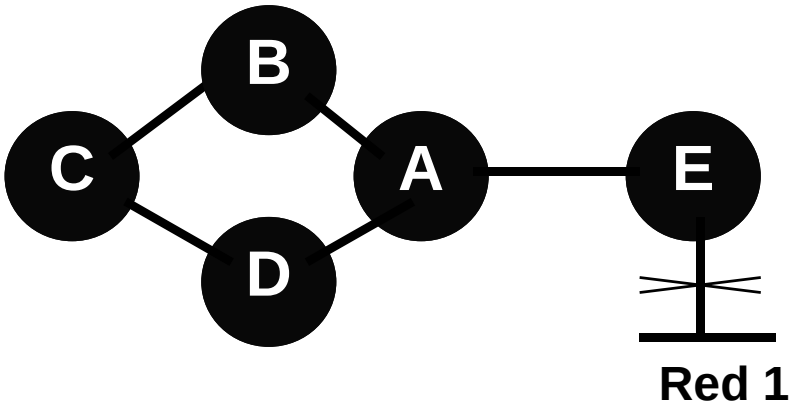
# RIP v1 (IX)
## Inversión de ruta

- En ocasiones se aplica la técnica de "horizonte dividido con inversión de ruta" (*split horizon with poisoned reverse*):
  - Cuando se envía un mensaje *Response* por una interfaz I, aquellas entradas que han sido aprendidas desde un vecino directamente conectado a la red en I, se transmiten con distancia infinita
  - => si hubiera un ciclo de orden 2 => se rompería inmediatamente.
    - Ejemplo: Si B fuera hacia C por A, el mensaje de A eliminaría esa ruta incorrecta.



# RIP v1 (X)

## Problema: bucles en el encaminamiento




1. Situación inicial: tablas de todos los routers han convergido a valores correctos.
- TE Router C: (red 1,B,d=4).
2. La red 1 falla => E anuncia a A (red 1,dist = ∞) =>
- TE Router A: (red 1,E, ∞).
3. A envía actualización a B, D (red1,d=∞) =>
- TE router B,D: (red 1,A, ∞).
4. C envía actualización periódica a D (no a B), indicando (red 1,d=4), pero no a B =>
- TE Router D: (red 1,C,d=5).
5. D envía nueva tabla a A =>
- TE Router A: (red 1,D,d=6).
6. A envía nueva tabla a B =>
- TE Router B: (red 1,A,d=7).

provocan ciclo

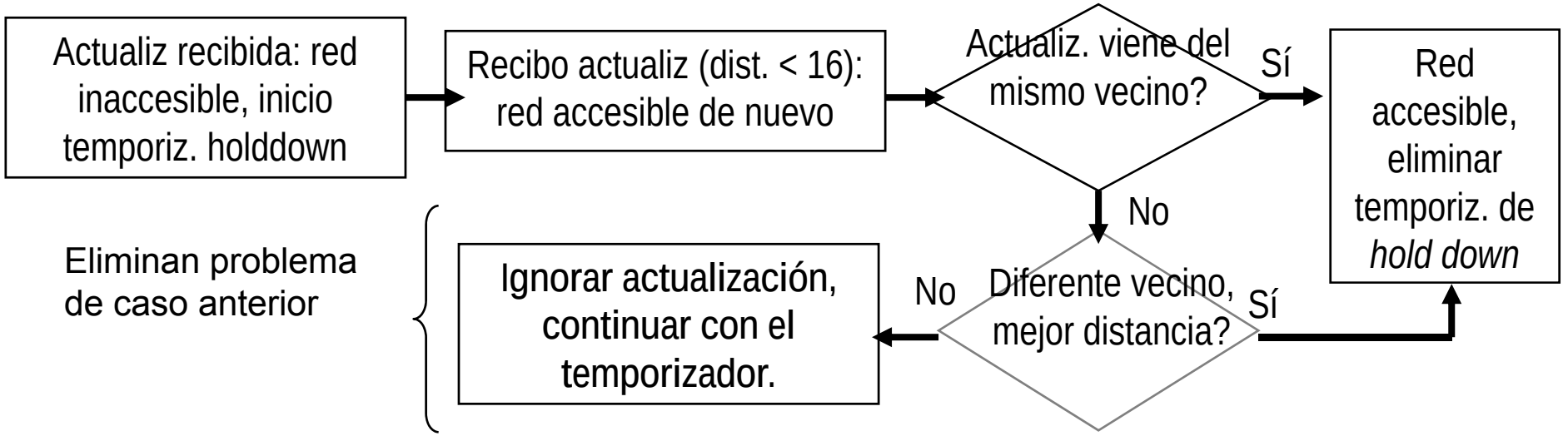
Split Horizon

Ciclo: dat destino red 1: C→B→A→D→C.

# RIP v1 (XI)
## Temporizadores de *hold down*

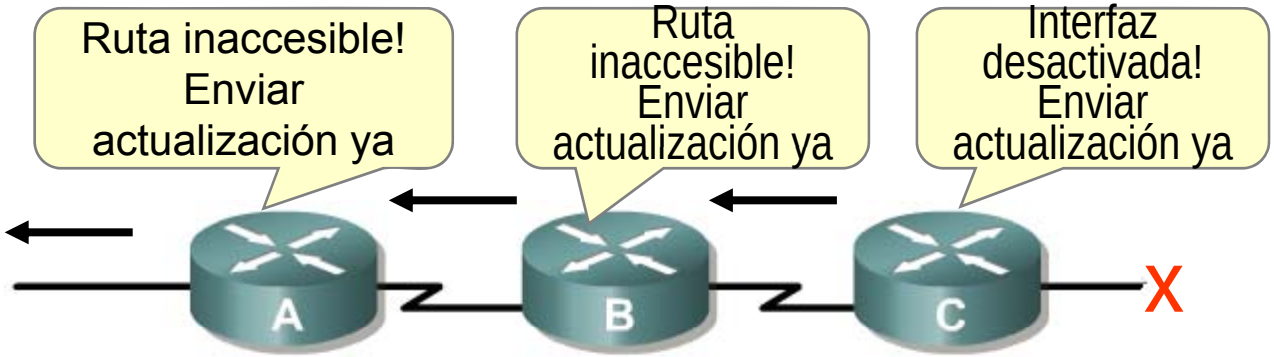
- Cuando un router almacena una entrada que indica que una red no es accesible (distancia infinita) =>

1. No se encaminan más datagramas a esa red destino.
2. Se lanza un temporizador de *hold down* (180 seg. por defecto). Si no cambia la distancia a ese destino en este tiempo => la ruta es definitivamente borrada de la tabla.



# RIP v1 (XII)
## Actualizaciones automáticas (triggered updates)

- Los ciclos de orden mayor que 2, pueden suceder a pesar del mecanismo de *split horizon* y *poissoned reverse*.
- Si estos ciclos aparecen cuando una red se hace inaccesible, no se resuelven hasta que los nodos acaben de "contar hasta infinito".
- El mecanismo de triggered updates consiste en que cuando la distancia a una red se actualiza => se envía un mensaje a los routers vecinos automáticamente, sin esperar al periodo de 30 segundos.
- En el mensaje de actualización, únicamente se incluyen las rutas que han cambiado.



# Ejemplo (RIP)

- RIP con *split horizon*, simple (sin *poissoned reverse*).
- Tras alcanzar la convergencia:
  - Mostrar tablas de encaminamiento.
  - Mostrar mensajes enviados por router 1 y router 2.

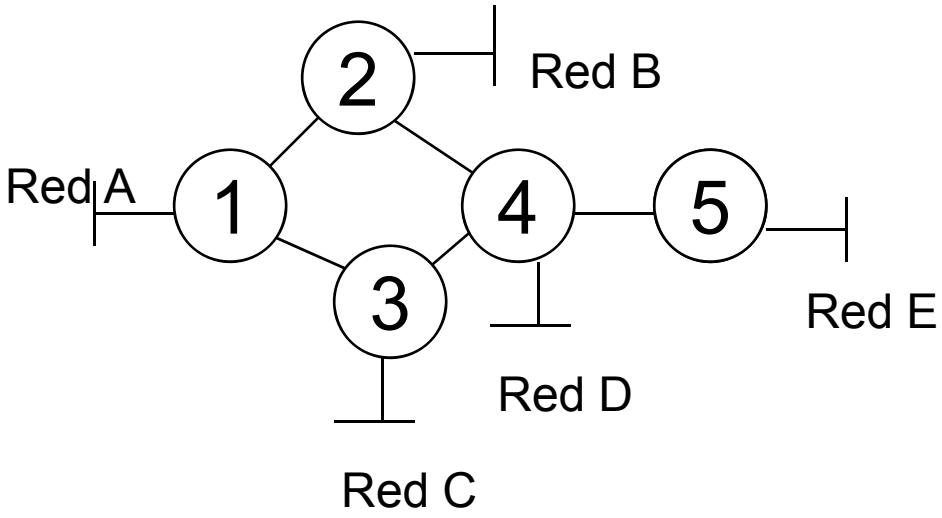



**Tabla encaminamiento**

**(red,GW,distancia)**

| | |
|---|---|
| 1 | (A;--;1);(B;2;2);(C;3;2);(D;2;3);(E;2;4) |
| 2 | (A;1;2);(B;--;1);(C;1;3);(D;4;2);(E;4;3) |
| 3 | (A;1;2);(B;1;3);(C;--;1);(D;4;2);(E;4;3) |
| 4 | (A;2;3);(B;2;1);(C;3;2);(D;--;1);(E;5;2) |
| 5 | (A;4;4);(B;4;3);(C;4;3);(D;4;2);(E;--;1) |

**Mensajes enviados**

| | |
|---|---|
| 1→A | (B;2);(C;2);(D;3);(E;4) |
| 1→2 | (A;1);(C;2) |
| 1→3 | (A;1);(B;2);(D;3);(E;4) |
| 2→B | (A;2);(C;3);(D;2);(E;3) |
| 2→1 | (B; 1);(D;2);(E; 3) |
| 2→4 | (A;2);(B;1);(C;3) |

# Bibliografía recomendada

- Douglas E. Comer, "Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture. 4th Edition", Prentice Hall 2000.
  - Capítulo 15: Routing Within an Autonomous System (RIP, OSPF).
- RIP versión 1 - RFC 1058, 1721.
- RIP version 2 - RFC 2453, 1388.