DD2490 p4 2010

Distance-vector and RIP

Olof Hagsand KTHNOC/NADA

Literature

RIP-lab

RFC 2453: RIPv2.

Sections 1-2 contains some introduction that can be useful to understand the context in which RIP is specified.
3.1-3.4 is an excellent overview of distance vector routing and the RIP protocol. It is all highly relevant and easy to read.
3.5 is too technical and can be skipped
3.6 outlines the RIP-1 header which is unfortunate since we deal only with RIP-2 in this course. However, read 3.6 together with section 4 to understand the formats of the RIP-2 messages.
3.7 is too technical and can be skipped.
3.8 describes RIP timers and is included.
3.9-3.10 are optional, not included.
4 should be read together with 3.6 only
5 and 6 are not included.

Routing Information Protocol - RIP

- RIP is a distance-vector protocol
- RIP uses Bellman-Ford to calculate routes
- RIP-1 (RFC 1058)
- RIP-2 (RFC 2453)
- Rip uses UDP as transport
  - Multicast (RIP-2) or Broadcast (RIP-1)
- Metric is Hop Counts
  - 1: directly connected
  - 16: infinity
  - RIP cannot support networks with diameter > 15.
- RIP uses distance vector
  - RIP messages contain a vector of hop counts.
  - Every node sends its routes to its neighbours
  - Route information gradially spreads through the network
  - Every node selects the route with smallest metric.

The Distance-Vector protocol

- Each router sends a list of distance-vectors (route with cost) to each neighbour periodically
- Every router selects the route with smallest metric.
- Metric is a positive integer
  - The cost to reach a destination: number of hops
  - Hop-count is limited to 1-15, 16 is “infinity”

Distance-Vector algorithm / Bellman-Ford (according to RFC 2453)

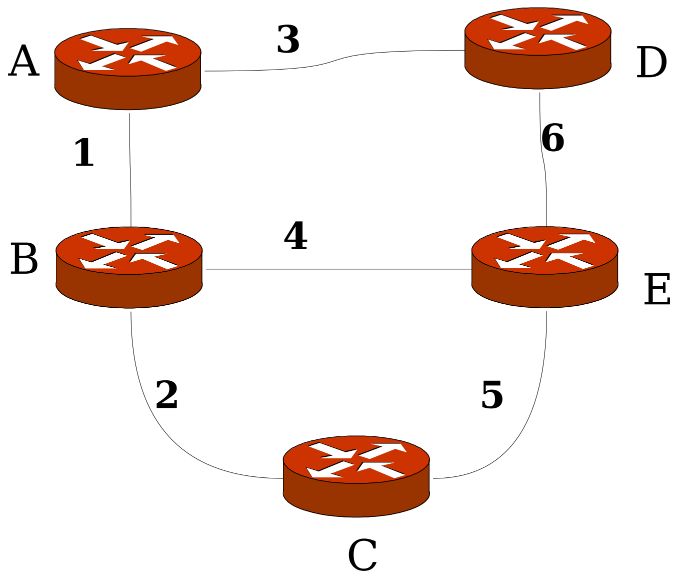
```
If it is possible to get from entity i to entity j directly, then a
cost, d(i,j), is associated with the hop between i and j. The cost is
infinite if i and j are not immediate neighbors. Let D(i,j) represent
the metric of the best route from entity i to entity j.

Then, the best metric must be described by
            D(i,i) = 0,                         all i
            D(i,j) = min [d(i,k) + D(k,j)],  otherwise
                       k

The algorithm:
Entity i gets its neighbors k to send it their estimates of their
distances to the destination j.  When i gets the estimates from k, it
adds d(i,k) to each of the numbers.  This is simply the cost of
traversing the network between i and k.  Now and then i compares the
values from all of its neighbors and picks the smallest.

It can be proven that this algorithm will converge to the correct
estimates of D(i,j) in finite time in the absence of topology
changes.
```

Example: Bellman-Ford



d(i,j)

	A	B	C	D	E
A		1		3	
B	1		2		4
C		2			5
D	3				6
E		4	5	6	

D_0(i,j)

	A	B	C	D	E
A	0				
B		0			
C			0		
D				0	
E					0

A's Distance-Vector

D_n(i,j)

	A	B	C	D	E
A	0	1	3	3	5
B	1	0	2	4	4
C	3	2	0	6	5
D	3	4	6	0	6
E	5	4	5	6	0

Note: this is a graph, not a real network!

Notes

- Keep a table with an entry for each destination N in the network.
- Store the distance D and next-hop G for each N in the table.
- Periodically, send the table to all neighbors (the distance-vector).
- For each update that comes in from neighbor G' (to N with a new distance):
  - Add the cost of the link to G' to the new distance to get D'.
  - Replace the route if D' < D.
  - If G = G', always replace the route.

Going to real networks

- “Now and then i compares the values from all of its neighbors and picks the smallest.”
  - Just keep track of the smallest cost
  - But many (eg Junos) implements RIP equal-cost multipath: save all routes with smallest costs
- IP networks require destinations and nexthops (not just nodes)
  - Destinations are networks eg 192.16.32.0/24
  - Next-hops are IP addresses, eg 192.16.32.1
- Suppose the topology changes , eg routers, links crash
  - There is no way to remove entries
  - Use timers (counters) and age the entries
  - Send updates every 30s
  - If you do not hear from a router in 180s, mark it as invalid

Route update example (1)

1. A sends its DV to B and D:

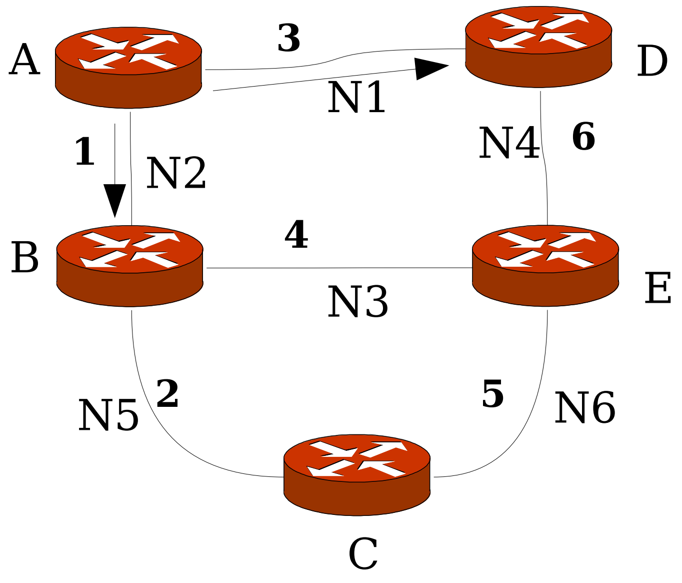
Dest	Cost	NextHop
N1	3	-
N2	1	-

2. B adds the cost of the link to A's DV (cost=1) and adds it to its DV

◦ runs split horizon and poison reverse?

Dest	Cost	NextHop
N2	1	-
N3	4	-
N5	2	-
N1	4	A

(Here A is really the IP address A has on N2)

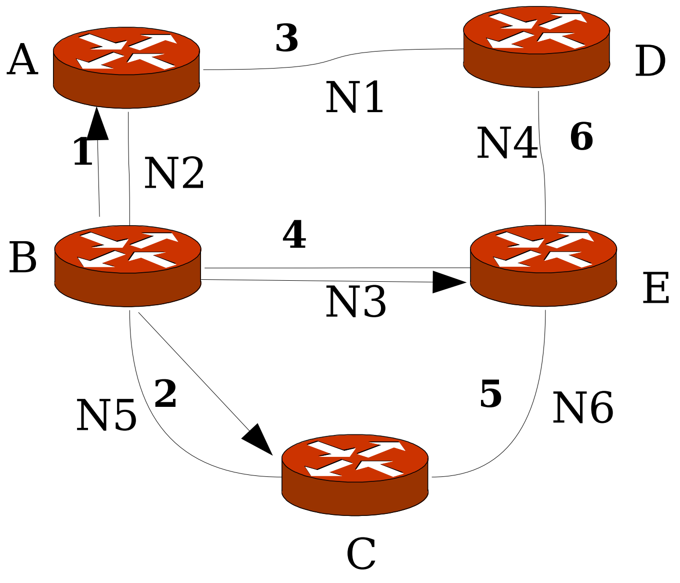


Note: this is a real network!

Route update example (2)

3. B Forwards its DV in its next update to its neighbors, A, C and E, etc.
The state of A:

Dest	Cost	NextHop
N1	3	-
N2	1	-
N3	5	B
N5	3	B



4. And so on: After a number of updates, the network reaches a stable state.
A:s final stable table:

Dest	Cost	NextHop
N1	3	-
N2	1	-
N3	5	B
N4	9	D
N5	3	B
N6	8	B

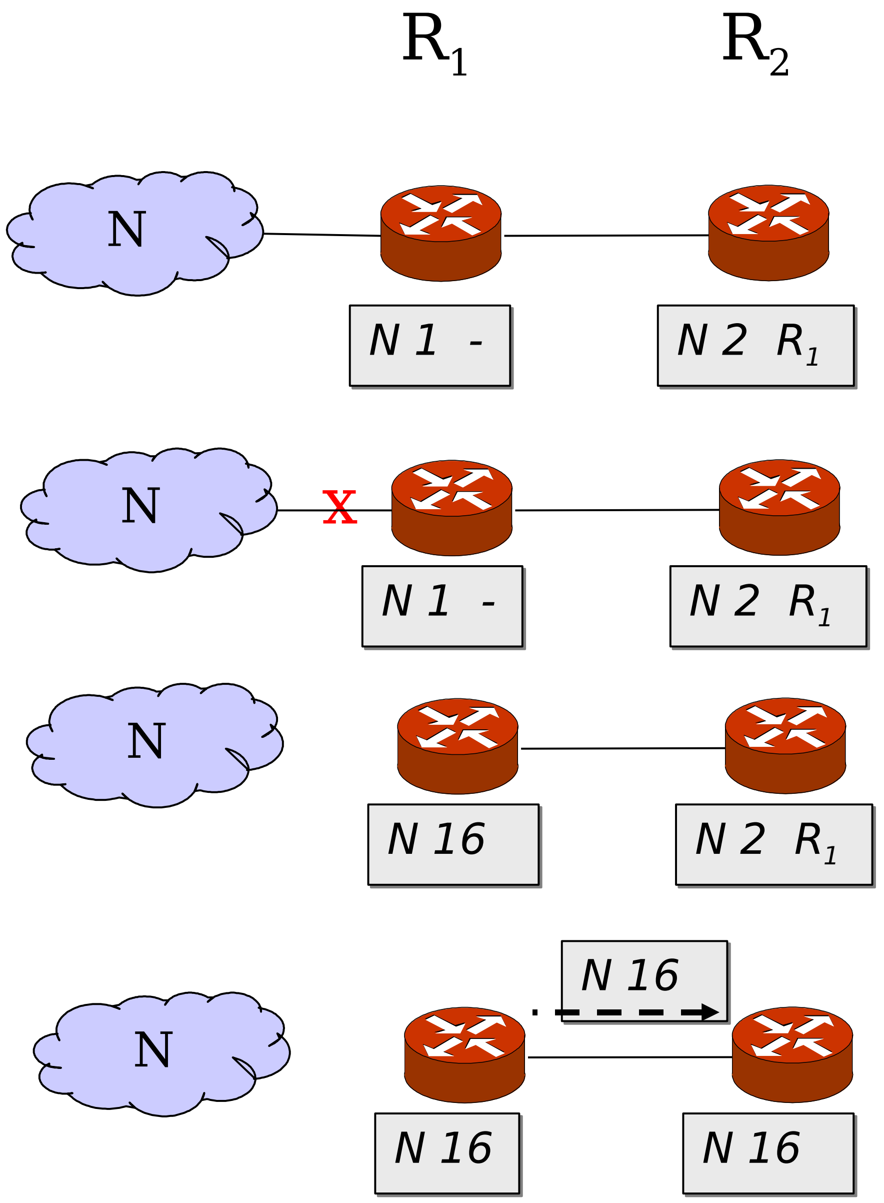
RIP Problem: Count to Infinity

Initially, R_1 and R_2 both have a route to N with metric 1 and 2, respectively.

The link between R_1 and N fails.

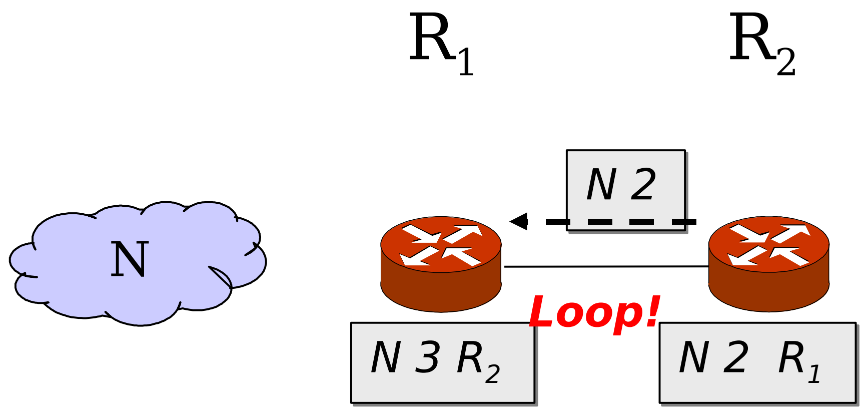
Now R_1 removes its route to N, by setting its metric to 16 (infinity).

Now two things can happen: Either R_1 reports its route to R_2. Everything is fine.

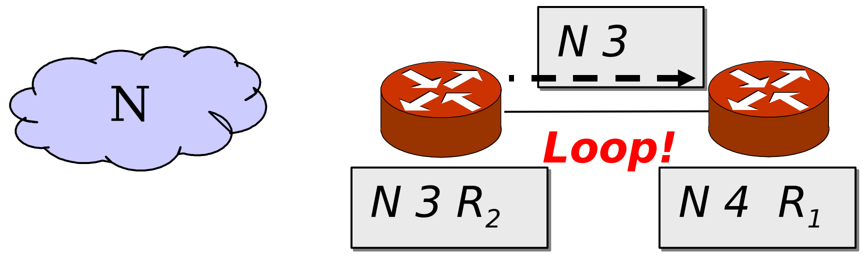


RIP Problem: Count to Infinity

The other alternative is that R_2, which still has a route to N, advertises it to R_1. Now things start to go wrong: packets to N are looped until their TTL expires!

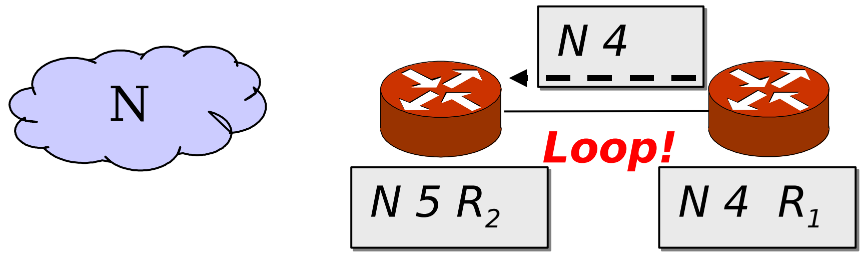


Eventually (~10-20s), R_1 sends an update to R_2. The cost to N increases, but the loop remains.

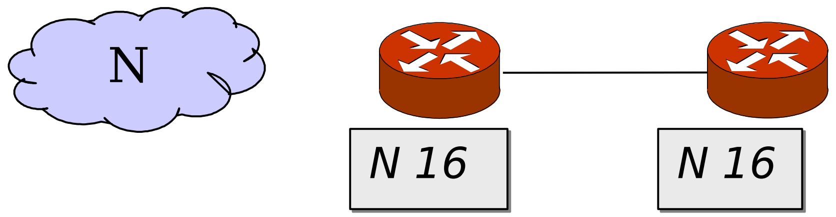


Yet some time later, R_2 sends an update to R_1.

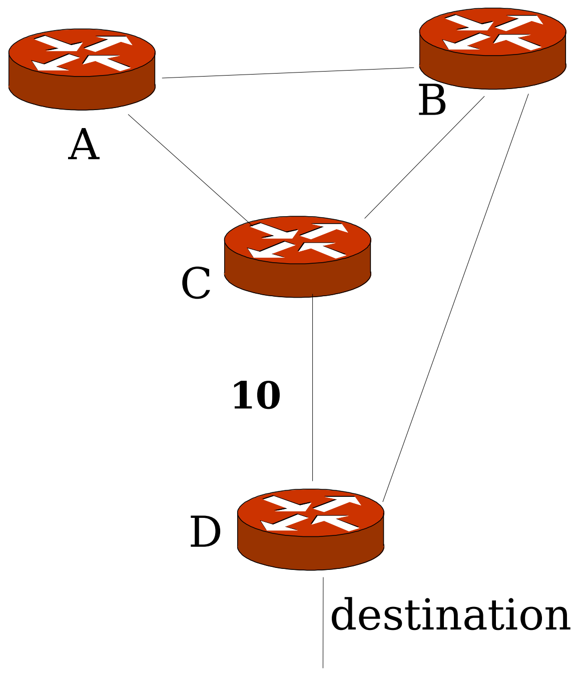
...



Finally, the cost reaches infinity at 16, and N is unreachable. The loop is broken!



More elaborate situation



RFC 2453, sec 2.2
All costs 1, except C – D: 10
Initial tables:

```
   D:  directly connected, metric 1
   B:  route via D, metric 2
   C:  route via B, metric 3
   A:  route via B, metric 3
```

Now, B -D fails

```
time ------>

 D: dir, 1   dir, 1   dir, 1   dir, 1  ...  dir, 1   dir, 1
 B: unreach  C,   4   C,   5   C,   6       C,  11   C,  12
 C: B,   3   A,   4   A,   5   A,   6       A,  11   D,  11
 A: B,   3   C,   4   C,   5   C,   6       C,  11   C,  12
```

The route via B lives on in the system causing a long-lasting loop between A and C

Solutions to count-to-infinity in RIP

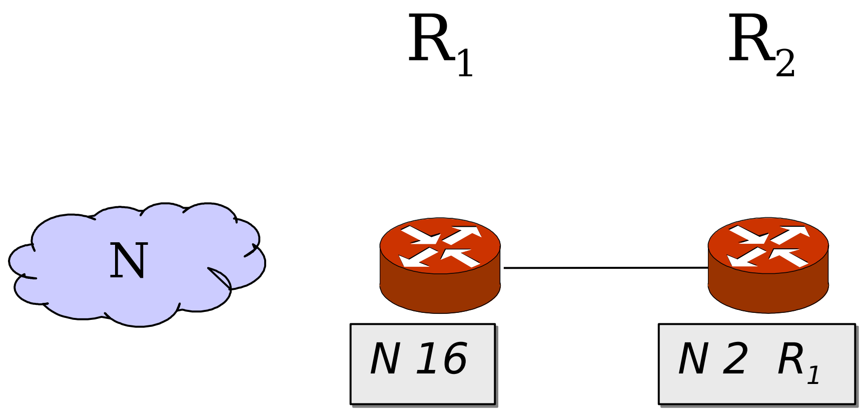
1. Infinity
2. Split Horizon
3. Split Horizon with Poison Reverse
4. Triggered Updates
5. Hold-down

Infinity

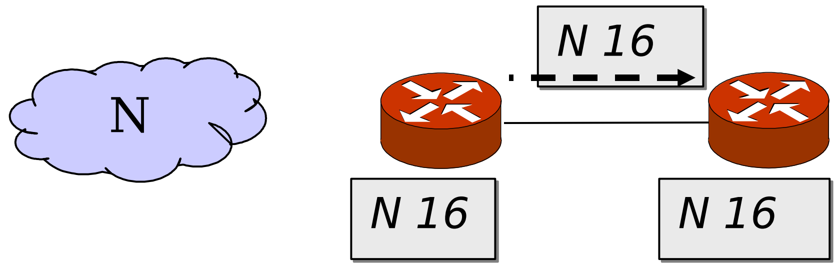
- Counting to infinity takes a long time.
- Thus infinity is set something more limited, namely 16.
- This limits the routing domain to 15 hops, and also makes counting to infinity a little faster,...

Split Horizon

- Do not send routes back over the same interface from which the route arrived.
- This helps in avoiding "mutual deception": two routers tell each other they can reach a destination via each other.
- Split Horizon MUST be supported on all RIP routers



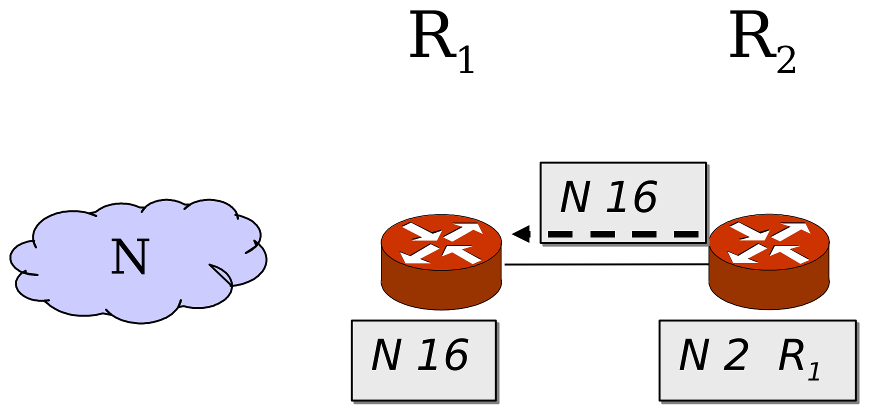
R_2, does not announce the route to N to R_1 since that is where it came from.



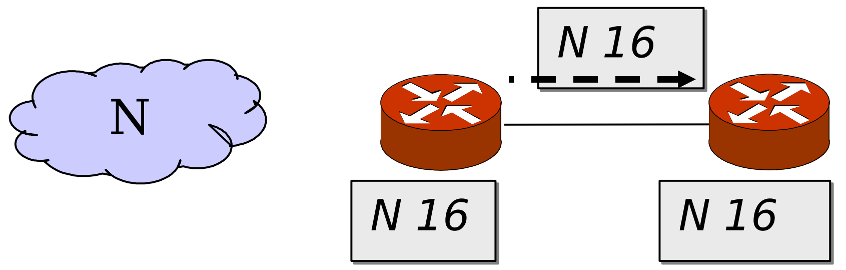
Eventually, R_1 reports its route to R_2 and everything is fine.

Split Horizon + Poison Reverse

- Advertise reverse routes with a metric of 16 (i.e., unreachable).
- Does not add inormation but breaks loops faster
- Adds protocol overhead
- Poison reverse SHOULD be supported by all RIP implementations, but it is OK to be able to turn it off.



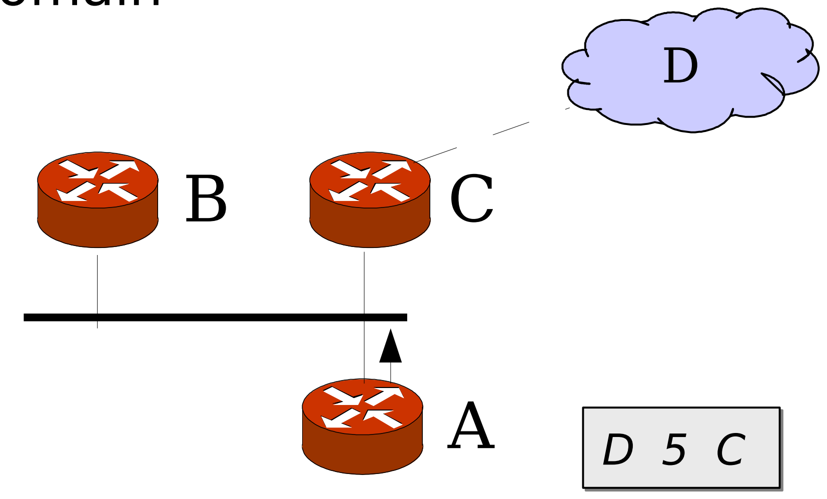
R_2 always announces an unreachable route to N to R_1.



Eventually, R_1 reports its route to R_2 and everything is fine.

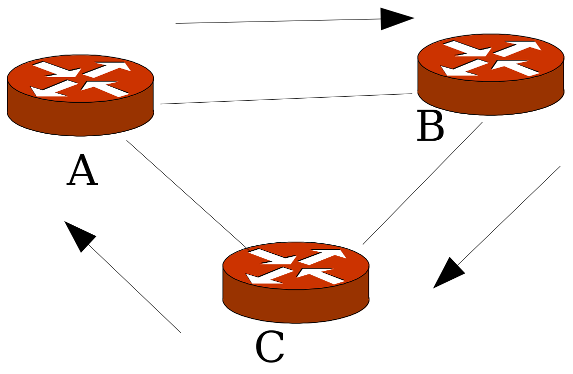
Interfaces vs next-hops

- The definition of Split horizon and Poison reverse is on *interfaces* not next-hop *routers*. Why?
- Consider a router A, a next-hop C and a destination D and A and C connected by a broadcast network N.
- No other routers (eg B) on N need to know that D is reachable via A since this would give an indirection and possibly a loop.
- -> A applies split horizon + poison reverse to the RIP update message sent via multicast on the interface
  - Per-next-hop would require specific messages on the broadcast domain



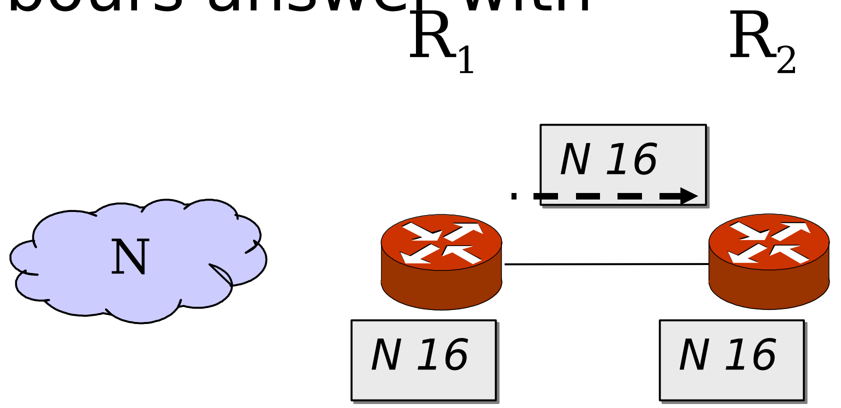
Remaining problems

- More than two routers involved in mutual deception
  - A may believe it has a route through B, B through C, and C through A
- In this case, split horizon with poison reverse does not help



Triggered Update

- Send out update immediately when metric changes
- But only the changed route, not the complete table
- This may lead to a cascade of updates
  - Apply the rule above recursively!
  - RIP filters these updates by not allowing more than one every 1-5 seconds.
- A router may use triggered update only when deleting routes (16).
- CISCO also implements “flash updates”: on boot, broadcast a request -> all neighbours answer with updates.

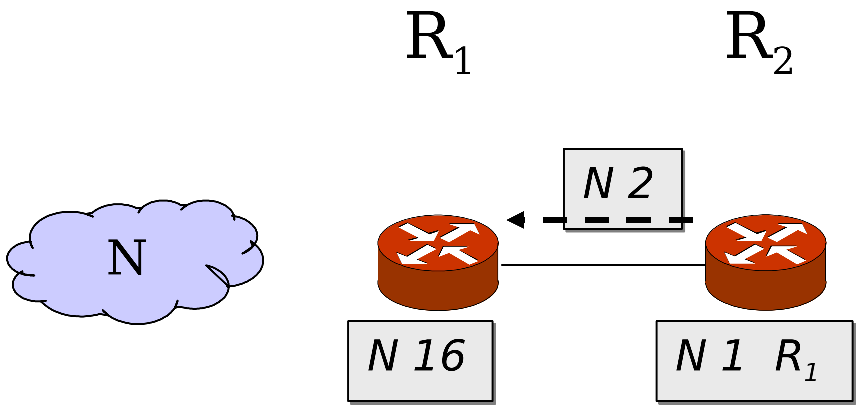


R_1 Immediately announces the broken link when it happens.

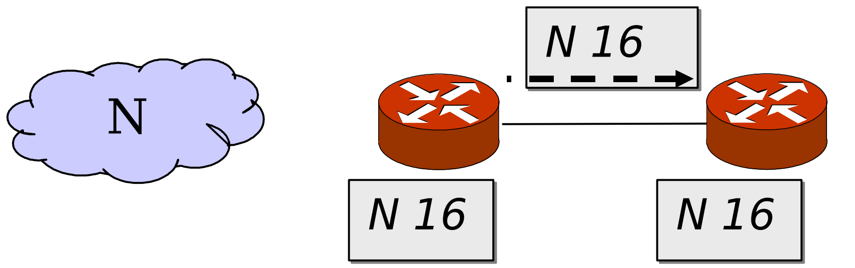
Hold Down

- When a route is removed, no update of this route is accepted for some period of time (hold-down time)- to give everyone a chance to remove the route.
- Hold-down is not in the RIP RFC, but CISCO implements it

R_1 ignores updates to N from R_2 for some period of time.



Eventually, R_1 sends the update to R_2.



RIP Timers

- Update
  - Time between each update: 30s with small random offset to avoid synchronization problems
- Timeout
  - If no updates are received, mark entry for deletion. It is then announced as unreachable: metric 16. Default: 180s.
- Garbage-collection
  - The entry is purged from the table – no longer announced. Default: 120s.
- Triggered-update timers
  - 1s – 5s (random)
  - Canceled by update timer

RIP protocol details

- RIP uses UDP on port 520
- RIPv1 uses UDP broadcast in neighbour communication
- RIPv2 uses UDP multicast (224.0.0.9) when sending periodic updates
- RIPv2 supports subnet masks (CIDR) and simple authentication
- A new RFC (RFC 4822) defines cryptographic authentication for RIPv2

RIPv2 header

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Command       | Version       |      Routing domain           |
+---------------+---------------+-------------------------------+
|       Address family          |      Route tag                |
+-------------------------------+-------------------------------+
|                          IP address                           |
+---------------------------------------------------------------+
|                          Netmask                              |
+---------------------------------------------------------------+
|                          Next Hop                             |
+---------------------------------------------------------------+
|                           Metric                              |
+---------------------------------------------------------------+
```

repeated entry

- If authentication is included, it must be the first entry.

RIPv2 fixed header fields

- Command: Request (1), Response (2)
- Version: RIPv2 (2)
- Routing domain: Routing process (if more than one instance)

RIPv2 advertisement fields

- Repeated for each destination:
  - Address family - IP (2) – Authentication (0xFFFF)
  - Route tag – AS number
  - IP address – Prefix (network address) of the destination
  - Netmask – destination netmask
  - Next hop – IP address of the nexthop router
  - Metric – The cost to the prefix

Disadvantages with RIP

- Slow convergence
  - Changes propagate slowly
  - Each neighbor only speaks ~every 30 seconds; information propagation time over several hops is long
- Instability
  - After a router or link failure RIP takes *minutes* to stabilize.
- Can only use hops count as metric.
- The maximum useful metric value is 15
  - Network diameter is limited to 15.
- RIP uses lots of bandwidth
  - It sends the whole routing table in updates.

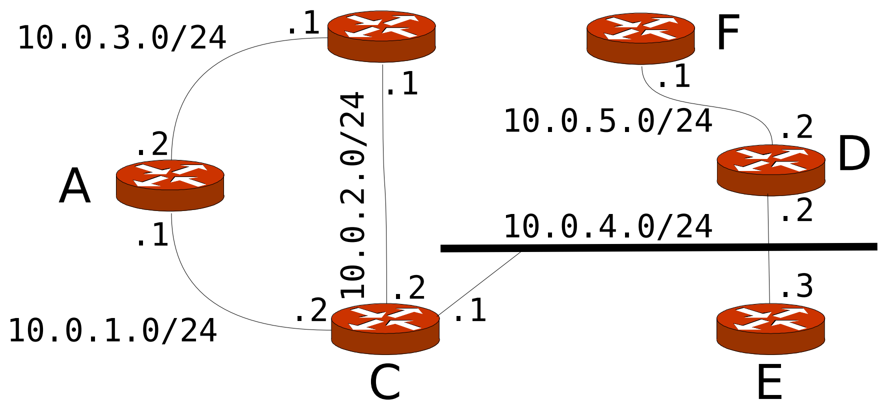
Why would anyone use RIP?

- Answer
  - It is easy to implement
  - It is generally available
  - Implementations have been rigorously tested
  - It is simple to configure.
  - It has little overhead (for small networks)
- RIP is also often used as a simple “glue” between two routing domains
  - Two different realms running different protocols may need a simple one-hop export/import protocol. RIP is ideal for that.

Other Distance-Vector protocols

- IGRP (CISCO proprietary)
  - Interior Gateway Routing Protocol
- EGP (Exterior Gateway Protocol) (RFC 827)
  - Predecessor to BGP
- DUAL algorithm
  - Diffusing Update Algorithm
  - Extends D-V with a distributed “diffusion” algorithm to avoid loops
- EIGRP (CISCO)
  - Extended IGRP
  - uses DUAL
- BGP uses path-vector – an extension to distance-vector

Homework 1



In the network above, all routers run RIPv2 and all link costs are 1. Assume an initial state for all routers, where only the directed connected networks are present in the router's routing tables. The destinations in the network are the /24 prefixes.

1. What is the initial routing table state of A? Express the routes as destination <prefix>, <cost> and <next-hop> (when applicable)
2. Assume C is the first router that sends a RIP update to all its neighbours. What is now the state of A's routing table?
3. Assume the routers run ECMP (Equal Cost MultiPath). After convergence, list the network(s) that is reachable via ECMP from A. Express the routes as prefix, cost, next-hop.
4. For an ECMP route in the previous exercise, assume A sends an IP packet to the .1 address. How many routers (hops) does the IP packet traverse before it reaches its destination using the different ECMP routes? Are there different number of hops for the different routes?
5. After convergence of RIP in the network, how many entries in the RIP update message does C send on 10.0.4.0/24 if RIP is configured to run:
- split horizon and poison reverse?
- only split horizon?
- neither split horizon nor poison reverse?