# The Network Layer

## Routing Algorithms

## Routing Algorithms

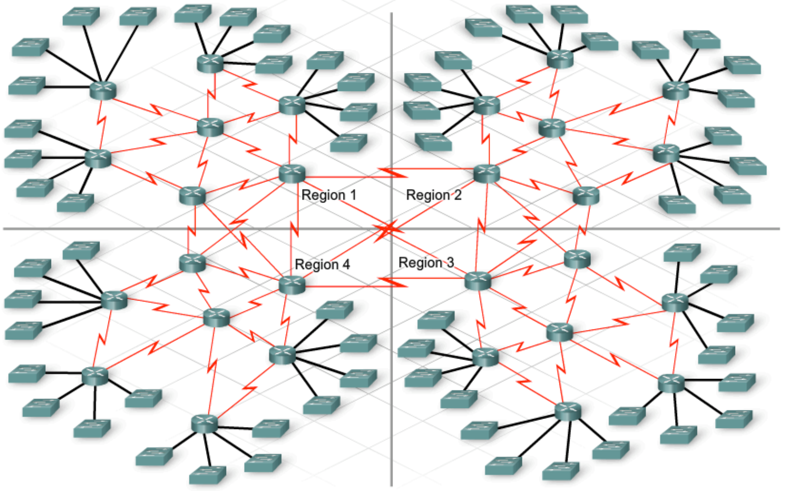
- Static vs. Dynamic Routing
- Flooding
- Distance Vector Routing
  - RIPv1 (“Routing Information Protocol”)
  - RIPv2 (“Routing Information Protocol”)

# Routing Algorithms

- Routing Algorithm
  - The part of the network layer responsible to decide which output line an incoming packet should be transmitted on. It can do this by dynamically updating the routing table.
  - It must handle link failures, changing topology, congestion...
- Forwarding
  - Handles each packet as it arrives and look for the output line to use in the routing tables.

# Routing Algorithms

Imagine maintaining static routing configurations for THIS network!



# Dynamic Routing Algorithms

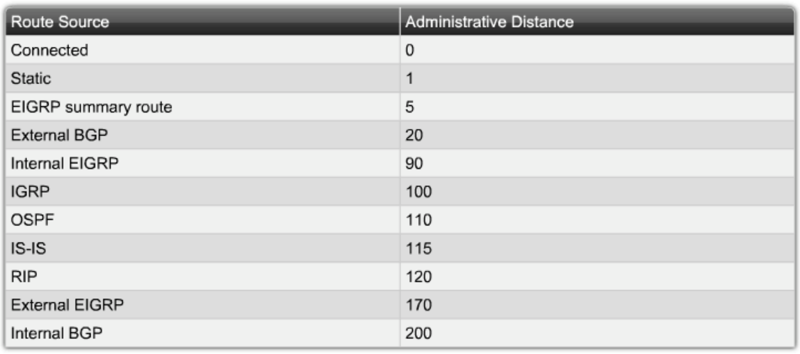
| | Interior Gateway Protocols | | | | Exterior Gateway Protocols |
|---|---|---|---|---|---|
| | Distance Vector Routing Protocols | | Link State Routing Protocols | | Path Vector |
| Classful | RIP | IGRP | | | EGP |
| Classless | RIPv2 | EIGRP | OSPFv2 | IS-IS | BGPv4 |
| IPv6 | RIPng | EIGRP for IPv6 | OSPFv3 | IS-IS for IPv6 | BGPv4 for IPv6 |

# Dynamic Routing Algorithms

| | Distance Vector | | | | Link State | |
|---|---|---|---|---|---|---|
| | RIPv1 | RIPv2 | IGRP | EIGRP | OSPF | IS-IS |
| Speed of Convergence | Slow | Slow | Slow | Fast | Fast | Fast |
| Scalability - Size of Network | Small | Small | Small | Large | Large | Large |
| Use of VLSM | No | Yes | No | Yes | Yes | Yes |
| Resource Usage | Low | Low | Low | Medium | High | High |
| Implementation and Maintenance | Simple | Simple | Simple | Complex | Complex | Complex |

# Dynamic Routing Algorithms

A router may learn about a network from several sources…
Which route should be used?
The one with LOWEST administrative distance!



| Route Source | Administrative Distance |
|---|---|
| Connected | 0 |
| Static | 1 |
| EIGRP summary route | 5 |
| External BGP | 20 |
| Internal EIGRP | 90 |
| IGRP | 100 |
| OSPF | 110 |
| IS-IS | 115 |
| RIP | 120 |
| External EIGRP | 170 |
| Internal BGP | 200 |

# Routing Algorithms

- Nonadaptive algorithms
  - The choice of the routes is computed in advance and downloaded to the routing tables when the network is booted.
  - STATIC ROUTING
- Adaptive algorithms
  - Change their routing decisions to reflect changes in the topology and traffic as well.
  - They differ from the place and time where they get their information and the metric used.

# Flooding

- A STATIC routing algorithm is flooding
  - Every incoming packet is sent on every outgoing line except the one it arrived on.
  - A hop-counter is decremented in each router, when it reaches 0 the packet is dropped.
  - Or each router manages a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list it is not flooded.
  - To prevent the list from growing without bound, it is augmented by a counter k, meaning that all sequence numbers below k have been seen.

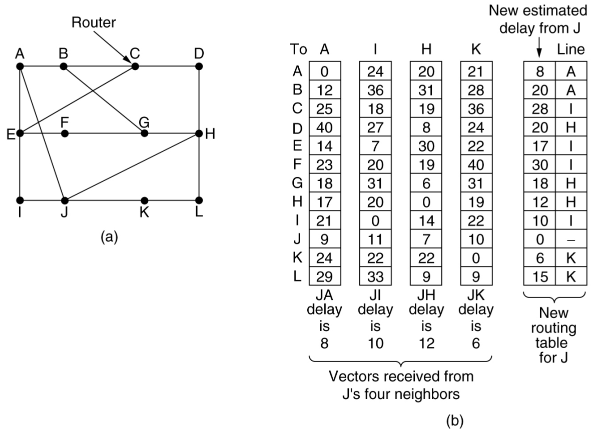
# Distance Vector Routing

- Modern networks use dynamic routing instead of static routing.
- One algorithm for dynamic routing is Distance Vector Routing (sometimes called Bellman-Ford Routing Algorithm).
- It is known in the Internet under the name RIP (Routing Information Protocol).
- Each router maintains a routing table indexed by, and containing one entry for, each router in the subnet.
- This table entry contain two parts: the preferred outgoing line to use for that destination and an estimation of time or distance to that destination.
- The metric used might be the number of hops, time delay in ms, load of the path, etc.

# Distance Vector Routing

- Once every T ms each router sends to each neighbour a list of costs to each destination and also receives a similar list from them.
- Imagine a table coming form **X** reporting $\mathbf{X_i}$ as the distance to reach router **i**. If the router know that the distance to **X** is **m** it can reach **i** via **X** with a distance of $\mathbf{X_i}$+**m**.
- By performing this calculation for each neighbour, a router can find out which estimate seems the best and use that estimate and the corresponding line in its new routing table.
- See example.

# Distance Vector Routing



| To | A | I | H | K | New estimated delay from J | Line |
|---|---|---|---|---|---|---|
| A | 0 | 24 | 20 | 21 | 8 | A |
| B | 12 | 36 | 31 | 28 | 20 | A |
| C | 25 | 18 | 19 | 36 | 28 | I |
| D | 40 | 27 | 8 | 24 | 20 | H |
| E | 14 | 7 | 30 | 22 | 17 | I |
| F | 23 | 20 | 19 | 40 | 30 | I |
| G | 18 | 31 | 6 | 31 | 18 | H |
| H | 17 | 20 | 0 | 19 | 12 | H |
| I | 21 | 0 | 14 | 22 | 10 | I |
| J | 9 | 11 | 7 | 10 | 0 | – |
| K | 24 | 22 | 22 | 0 | 6 | K |
| L | 29 | 33 | 9 | 9 | 15 | K |
| | JA delay is 8 | JI delay is 10 | JH delay is 12 | JK delay is 6 | New routing table for J | |

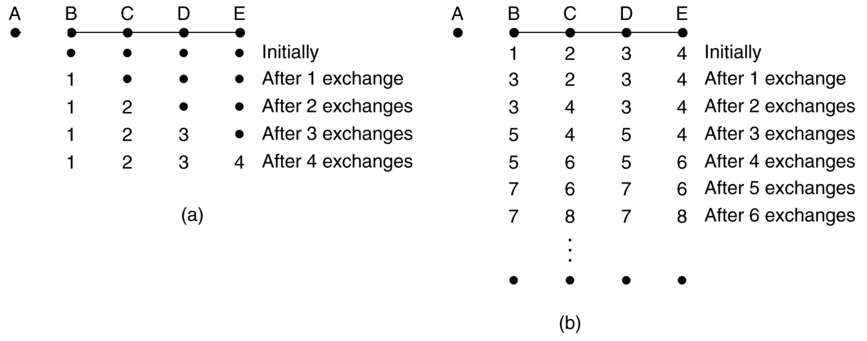
Vectors received from J's four neighbors

(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

# Distance Vector Routing

- Distance vector RA have a convergence problem.
- They react fast to good news but very slowly to bad news.
- Imagine a router that has a long delay to X. If on the next exchange *A* reports a short delay to X the router immediately relays its traffic through *A. In one exchange the good news is processed.*
- For bad news its not so simple...
- See example.

# Distance Vector Routing



| A | B | C | D | E | |
|---|---|---|---|---|---|
| | • | • | • | • | Initially |
| | 1 | • | • | • | After 1 exchange |
| | 1 | 2 | • | • | After 2 exchanges |
| | 1 | 2 | 3 | • | After 3 exchanges |
| | 1 | 2 | 3 | 4 | After 4 exchanges |

(a)

| A | B | C | D | E | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | Initially |
| | 3 | 2 | 3 | 4 | After 1 exchange |
| | 3 | 4 | 3 | 4 | After 2 exchanges |
| | 5 | 4 | 5 | 4 | After 3 exchanges |
| | 5 | 6 | 5 | 6 | After 4 exchanges |
| | 7 | 6 | 7 | 6 | After 5 exchanges |
| | 7 | 8 | 7 | 8 | After 6 exchanges |
| | | ⋮ | | | |
| | • | • | • | • | |

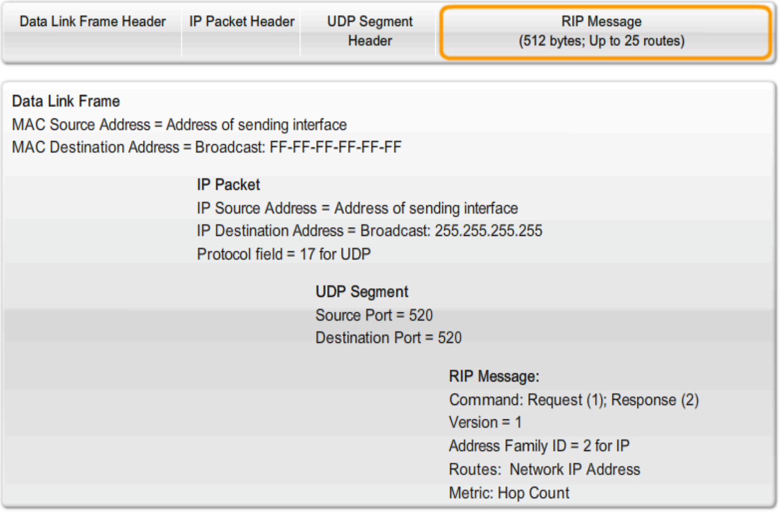
(b)

The count-to-infinity problem.

# RIP (by CISCO)

- **Periodic Updates** are sent at regular intervals (30 seconds default).
- **RIP Neighbors** are routers that share a link and are configured to use the same routing protocol. The router is only aware of the network addresses of its own interfaces and the remote network addresses it can reach through its neighbors.
- Routers using distance vector routing **are not aware of the network topology**.
- Broadcast Updates are sent to 255.255.255.255. Neighboring routers that are configured with the same routing protocol will process the updates.
- **Entire Routing Table Updates are sent**, with some exceptions to be discussed later, periodically to all neighbors

# Encapsulated RIPv1 Message



| Data Link Frame Header | IP Packet Header | UDP Segment Header | RIP Message (512 bytes; Up to 25 routes) |
|---|---|---|---|

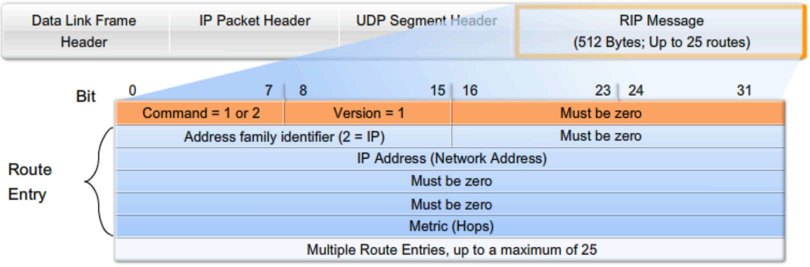
Data Link Frame
MAC Source Address = Address of sending interface
MAC Destination Address = Broadcast: FF-FF-FF-FF-FF-FF

IP Packet
IP Source Address = Address of sending interface
IP Destination Address = Broadcast: 255.255.255.255
Protocol field = 17 for UDP

UDP Segment
Source Port = 520
Destination Port = 520

RIP Message:
Command: Request (1); Response (2)
Version = 1
Address Family ID = 2 for IP
Routes: Network IP Address
Metric: Hop Count

# RIPv1 Message



Data Link Frame Header
IP Packet Header
UDP Segment Header
RIP Message (512 Bytes; Up to 25 routes)

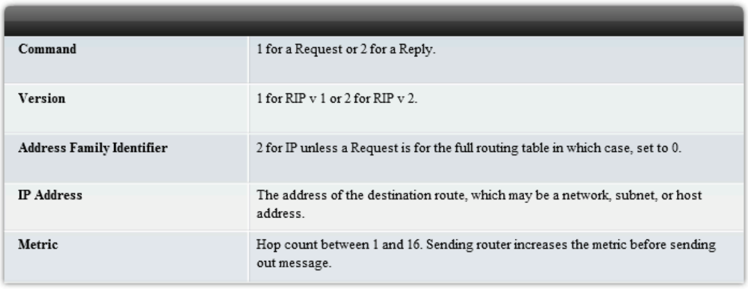
Bit 0 7 8 15 16 23 24 31

Command = 1 or 2
Version = 1
Must be zero

Route Entry

Address family identifier (2 = IP)
Must be zero
IP Address (Network Address)
Must be zero
Must be zero
Metric (Hops)

Multiple Route Entries, up to a maximum of 25



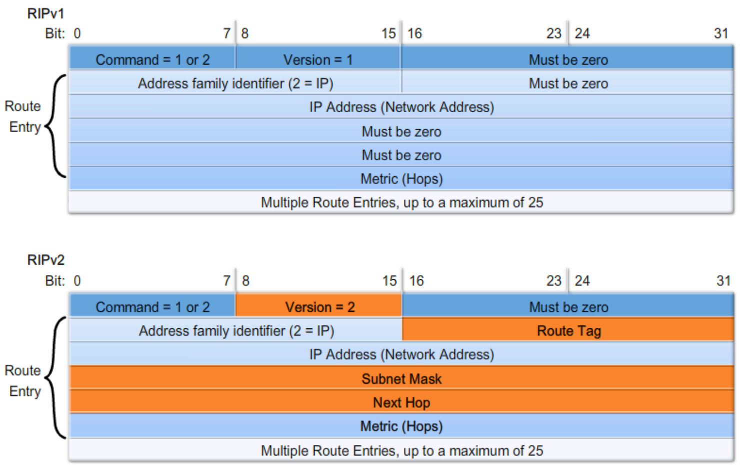
| | |
|---|---|
| **Command** | 1 for a Request or 2 for a Reply. |
| **Version** | 1 for RIP v 1 or 2 for RIP v 2. |
| **Address Family Identifier** | 2 for IP unless a Request is for the full routing table in which case, set to 0. |
| **IP Address** | The address of the destination route, which may be a network, subnet, or host address. |
| **Metric** | Hop count between 1 and 16. Sending router increases the metric before sending out message. |

# RIPv1 vs. RIPv2

- RIPv2
  - Fully compatible with RIPv1
  - Supports VLSM in advertised Routes
  - Faster convergence (next hop field present)
  - Security
  - Uses multicast (224.0.0.9) instead of broadcasts
  - Does not automatically summarizes routes on the network boundary

# RIPv1 vs. RIPv2

Comparing RIPv1 and RIPv2 Message Formats



RIPv1

Bit: 0 7 | 8 15 | 16 23 | 24 31

Command = 1 or 2 | Version = 1 | Must be zero

Route Entry:

Address family identifier (2 = IP) | Must be zero

IP Address (Network Address)

Must be zero

Must be zero

Metric (Hops)

Multiple Route Entries, up to a maximum of 25

RIPv2

Bit: 0 7 | 8 15 | 16 23 | 24 31

Command = 1 or 2 | Version = 2 | Must be zero

Route Entry:

Address family identifier (2 = IP) | Route Tag

IP Address (Network Address)

Subnet Mask

Next Hop

Metric (Hops)

Multiple Route Entries, up to a maximum of 25

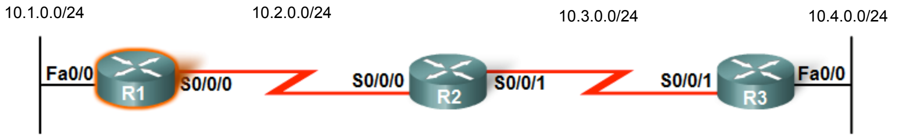
# Maintaining the Routing Table

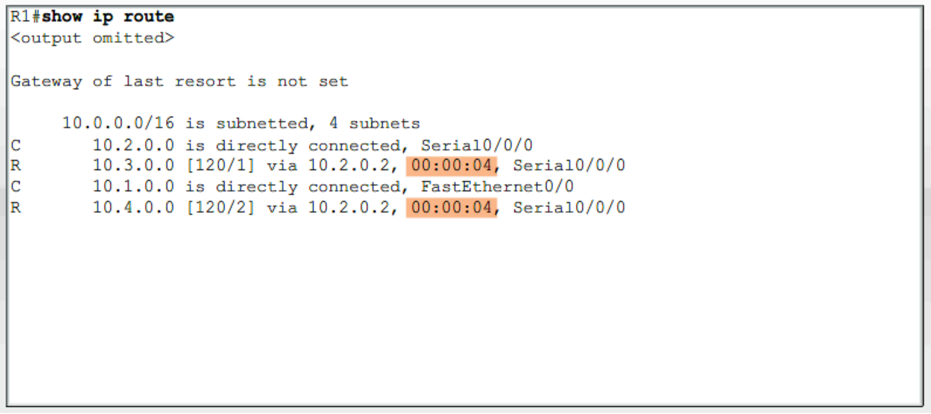
- The 30-second interval is a route update timer that also aids in tracking the age of routing information in the routing table.
- The age of routing information in a routing table is refreshed each time an update is received.
- Changes may occur for several reasons, including:
  - Failure of a link
  - Introduction of a new link
  - Failure of a router
  - Change of link parameters

# RIP Timers

- **Invalid Timer:**
  - If an update has not been received to refresh an existing route after 180 seconds (the default), the route is marked as invalid by setting the metric to 16. The route is retained in the routing table until the flush timer expires.
- **Flush Timer:**
  - By default, the flush timer is set for 240 seconds, which is 60 seconds longer than the invalid timer. When the flush timer expires, the route is removed from the routing table.
- **Holddown Timer:**
  - This timer stabilizes routing information and helps prevent routing loops during periods when the topology is converging on new information.
  - Once a route is marked as unreachable, it must stay in holddown long enough for all routers in the topology to learn about the unreachable network.
  - By default, the holddown timer is set for 180 seconds.

# RIP Example





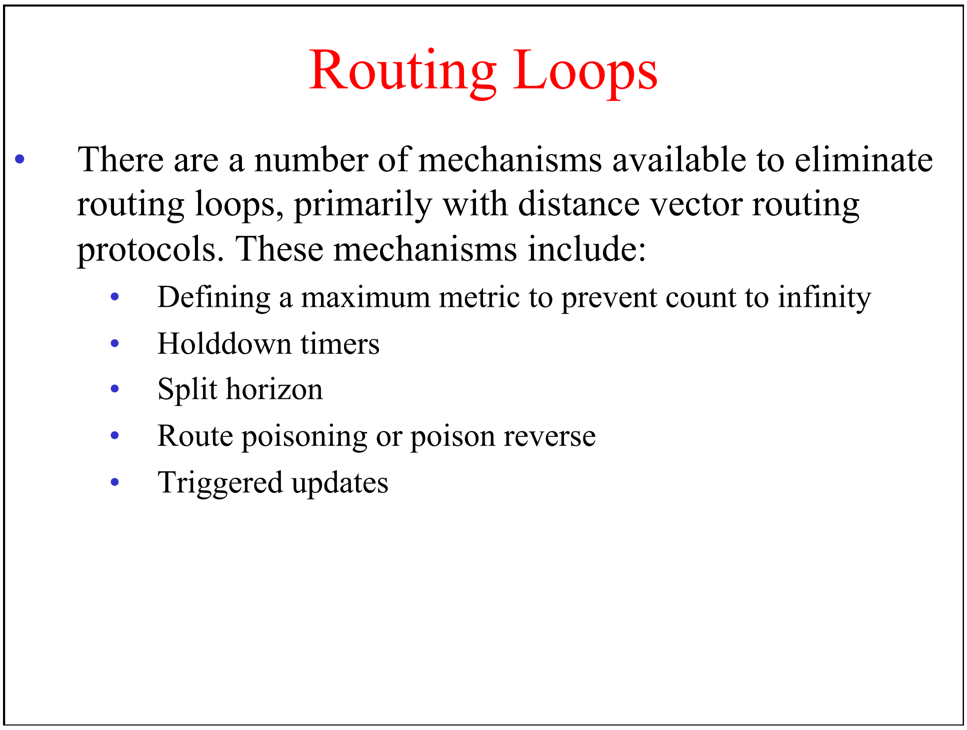
```
R1#show ip route
<output omitted>

Gateway of last resort is not set

     10.0.0.0/16 is subnetted, 4 subnets
C       10.2.0.0 is directly connected, Serial0/0/0
R       10.3.0.0 [120/1] via 10.2.0.2, 00:00:04, Serial0/0/0
C       10.1.0.0 is directly connected, FastEthernet0/0
R       10.4.0.0 [120/2] via 10.2.0.2, 00:00:04, Serial0/0/0
```

# Routing Loops

- The loop may be a result of:
  - Incorrectly configured static routes
  - Incorrectly configured route redistribution (redistribution is a process of handing the routing information from one routing protocol to another routing protocol)
  - Inconsistent routing tables not being updated due to slow convergence in a changing network
  - Incorrectly configured or installed discard routes

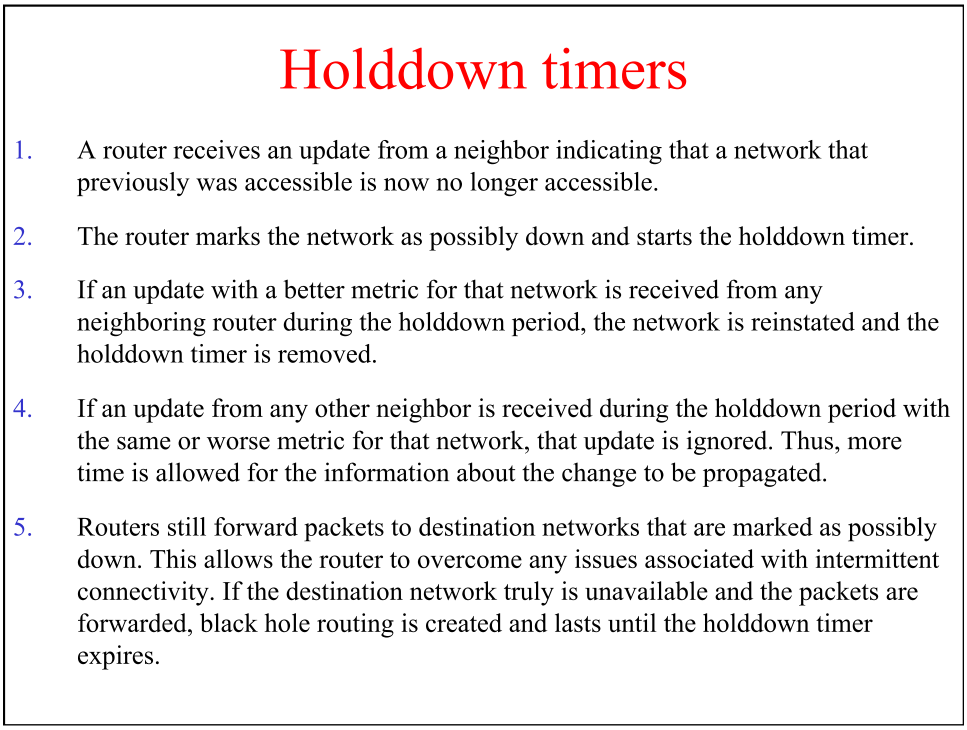
# Routing Loops

- A routing loop can create the following conditions:
  - Link bandwidth will be used for traffic looping back and forth between the routers in a loop.
  - A router's CPU will be strained due to looping packets.
  - A router's CPU will be burdened with useless packet forwarding that will negatively impact the convergence of the network.
  - Routing updates may get lost or not be processed in a timely manner. These conditions would introduce additional routing loops, making the situation even worse.
  - Packets may get lost in "black holes."



# Routing Loops

- There are a number of mechanisms available to eliminate routing loops, primarily with distance vector routing protocols. These mechanisms include:
  - Defining a maximum metric to prevent count to infinity
  - Holddown timers
  - Split horizon
  - Route poisoning or poison reverse
  - Triggered updates

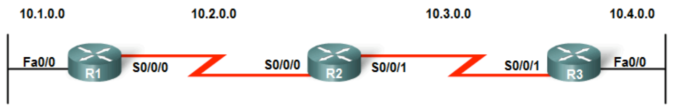


# Holddown timers

1. A router receives an update from a neighbor indicating that a network that previously was accessible is now no longer accessible.
2. The router marks the network as possibly down and starts the holddown timer.
3. If an update with a better metric for that network is received from any neighboring router during the holddown period, the network is reinstated and the holddown timer is removed.
4. If an update from any other neighbor is received during the holddown period with the same or worse metric for that network, that update is ignored. Thus, more time is allowed for the information about the change to be propagated.
5. Routers still forward packets to destination networks that are marked as possibly down. This allows the router to overcome any issues associated with intermittent connectivity. If the destination network truly is unavailable and the packets are forwarded, black hole routing is created and lasts until the holddown timer expires.

# Split Horizon

- Never advertise a route in the network interface where it was learned
- Applying split horizon to the previous example of route 10.4.0.0 produces the following actions:
  - R3 advertises the 10.4.0.0 network to R2.
  - R2 receives the information and updates its routing table.
  - R2 then advertises the 10.4.0.0 network to R1 out S0/0/0. R2 does not advertise 10.4.0.0 to R3 out S0/0/1, because the route originated from that interface.
  - R1 receives the information and updates its routing table.
  - Because of split horizon, R1 also does not advertise the information about network 10.4.0.0 back to R2.



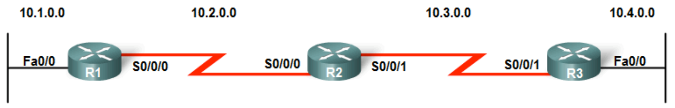
| Network | Interface | Hop |
|---|---|---|
| 10.1.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/0 | 2 |

| Network | Interface | Hop |
|---|---|---|
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/1 | 0 |
| 10.1.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/1 | 1 |

| Network | Interface | Hop |
|---|---|---|
| 10.3.0.0 | S0/0/1 | 0 |
| 10.4.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/1 | 1 |
| 10.1.0.0 | S0/0/1 | 2 |

# Split Horizon

- Complete routing updates are exchanged, with the exception of routes that violate the split horizon rule. The results look like this:
  - R2 advertises networks 10.3.0.0 and 10.4.0.0 to R1.
  - R2 advertises networks 10.1.0.0 and 10.2.0.0 to R3.
  - R1 advertises network 10.1.0.0 to R2.
  - R3 advertises network 10.4.0.0 to R2.



| Network | Interface | Hop |
|---|---|---|
| 10.1.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/0 | 2 |

| Network | Interface | Hop |
|---|---|---|
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/1 | 0 |
| 10.1.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/1 | 1 |

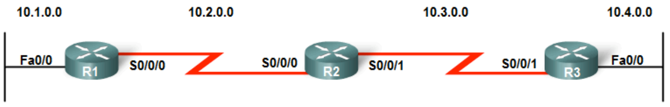
| Network | Interface | Hop |
|---|---|---|
| 10.3.0.0 | S0/0/1 | 0 |
| 10.4.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/1 | 1 |
| 10.1.0.0 | S0/0/1 | 2 |

# Route Poisoning

- Route poisoning is used to mark the route as unreachable in a routing update that is sent to other routers.
- Unreachable is interpreted as a metric that is set to the maximum. For RIP, a poisoned route has a metric of 16.
- Route poisoning speeds up the convergence process as the information about the unreachable route spreads through the network more quickly than waiting for the hop count to reach "infinity".

# Split Horizon with Poison Reverse

- The following process occurs:
  - Network 10.4.0.0 becomes unavailable due to a link failure.
  - R3 poisons the metric with a value of 16 and then sends out a triggered update stating that 10.4.0.0 is unavailable.
  - R2 processes that update, invalidates the routing entry in its routing table, and immediately sends a poison reverse back to R3.



| Network | Interface | Hop |
|---|---|---|
| 10.1.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/0 | 2 |

| Network | Interface | Hop |
|---|---|---|
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/1 | 0 |
| 10.1.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/1 | 1 |

| Network | Interface | Hop |
|---|---|---|
| 10.3.0.0 | S0/0/1 | 0 |
| 10.4.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/1 | 1 |
| 10.1.0.0 | S0/0/1 | 2 |

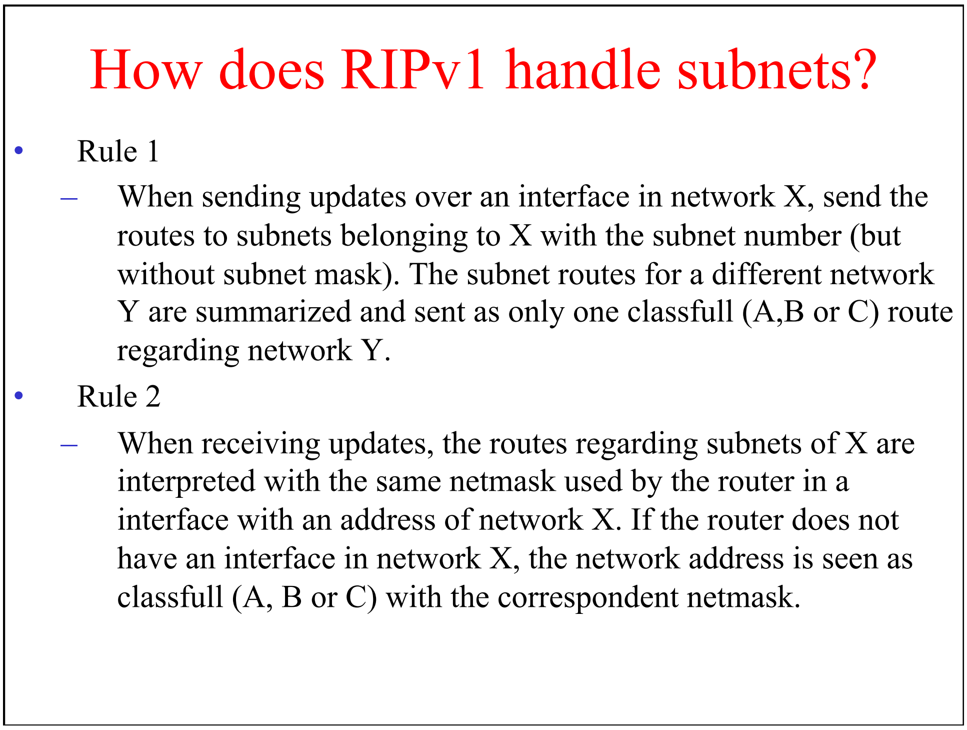
  - Poison reverse is a specific circumstance that overrides split horizon. It occurs to ensure that R3 is not susceptible to incorrect updates about network 10.4.0.0.

# Triggered updates

- With the previous techniques routers only send updates on the scheduled period (30s).
- To speed up the convergence process a new kind of updates is defined: Triggered Updates.
- Triggered updates are sent when one of the following topologic change occurs:
  - An interface changes state (up or down)
  - A route has entered (or exited) the "unreachable" state
  - A route is installed in the routing table

# Summary

- **Split horizon** -- Prevents loops between adjacent routers.
  - Rule: Never advertise a route out of the interface through which you learned it!
- **Poison reverse** -- Prevents larger loops.
  - Rule: Once you learn that a route is unreachable through an interface, advertise it as unreachable back through that same interface!
- **Holddown timer** -- Prevents incorrect route information from entering routing tables.
  - Rule: After a route is advertised as down, do not listen to routing updates with equal or higher metric on that route for a specific period of time!
- **Triggered Events** – Speed up network convergence.
  - Rule: Don’ t wait for a scheduled update on topologic changes!



# How does RIPv1 handle subnets?

- Rule 1
  - When sending updates over an interface in network X, send the routes to subnets belonging to X with the subnet number (but without subnet mask). The subnet routes for a different network Y are summarized and sent as only one classfull (A,B or C) route regarding network Y.
- Rule 2
  - When receiving updates, the routes regarding subnets of X are interpreted with the same netmask used by the router in a interface with an address of network X. If the router does not have an interface in network X, the network address is seen as classfull (A, B or C) with the correspondent netmask.