Distance vector protocols

Distance Vector Routing Principles
Routing loops and countermeasures to loops
Bellman-Ford route calculations
RIP

Distance Vector Routing Principles

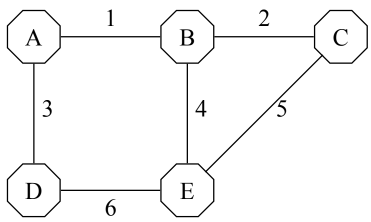
RIP – Routing Information Protocol is a basic protocol for interior routing

- RIP is a distance vector protocol
  - Based on the **Bellman-Ford** algorithm
- The **routing table** contains information about other known nodes
  - distance (cost) in hops
  - link (interface) identifier

E to	Link	Distance
E	-	0
B	4	1
A	4	2
D	6	1
C	5	1

- The nodes periodically send **distance vectors** based on the routing tables on all their links
- The nodes update their routing table with received distance vectors

Let us study the principles of DV protocols



Example network with nodes A, B, C, D, E and links 1, 2, 3, 4, 5, 6.

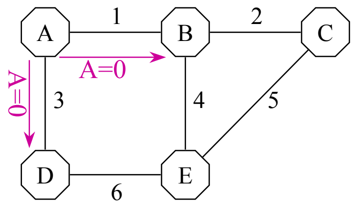
Initial state: Nodes know their own addresses and interfaces, nothing more.

Node A creates its routing table:

From A to node	Link	Distance
A	local	0

The corresponding distance vector (DV) is: A=0

Generation of routing tables starts when all routers send their DVs on all interfaces



Let's look at reception in Node B. First the table of B is:

From node B to	Link	Distance
B	local	0

1. B receives the distance vector A=0
2. B ***increments the DV with +1*** ⇒ A=1
3. B looks for the result in its routing table, no match
4. B adds the result to its RT, the result is

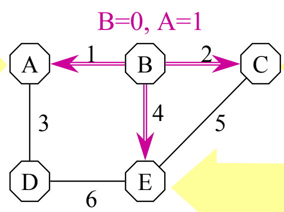
From node B to	Link	Distance
B	local	0
A	1	1

5. B generates its distance vector B=0, A=1

B creates its own DV and sends it to all neighbors

A to	Link	Distance
A	-	0
B	1	1

A=2 > A=0

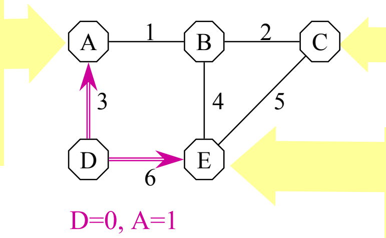


C to	Link	Distance
C	-	0
B	2	1
A	2	2

E to	Link	Distance
E	-	0
B	4	1
A	4	2

D sends its distance vector to all neighbors

A to	Link	Distance
A	-	0
B	1	1
D	3	1



C to	Link	Distance
C	-	0
B	2	1
A	2	2

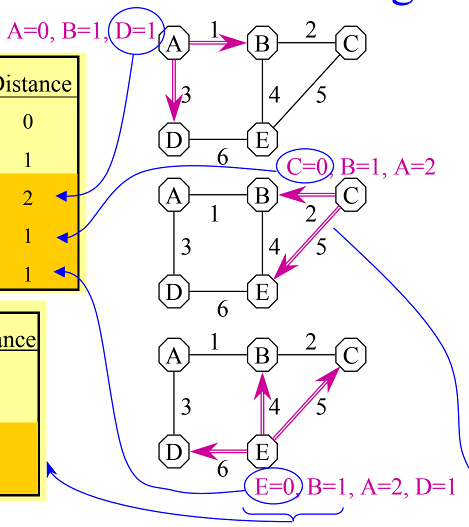
E to	Link	Distance
E	-	0
B	4	1
A	4	2
D	6	1

A=2 == A=2 ⇒ no change

The nodes whose RT changed create DVs and send them to neighbors

B to	Link	Distance
B	-	0
A	1	1
D	1	2
C	2	1
E	4	1

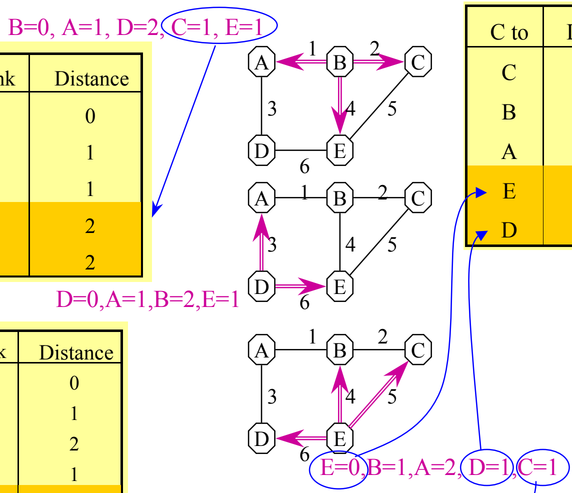
D to	Link	Distance
D	-	0
A	3	1
B	3	2
E	6	1



E to	Link	Distance
E	-	0
B	4	1
A	4	2
D	6	1
C	5	1

Again the changes are sent ...

A to	Link	Distance
A	-	0
B	1	1
D	3	1
C	1	2
E	1	2

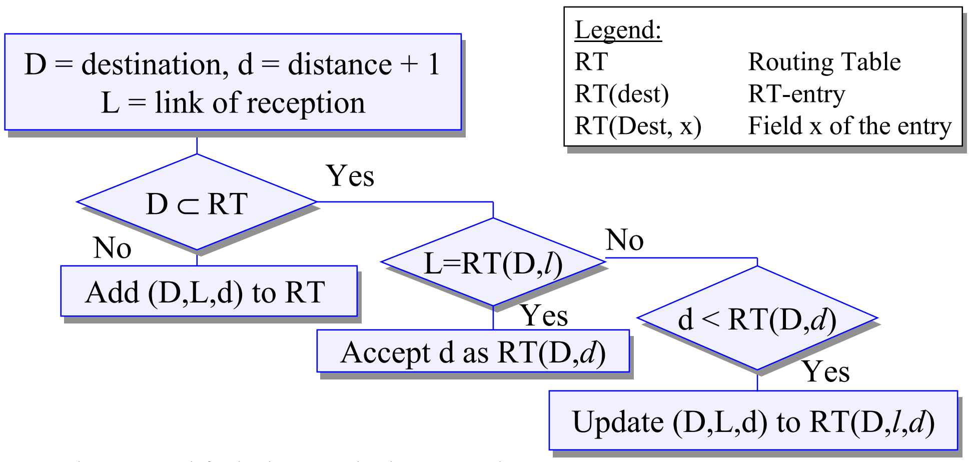


C to	Link	Distance
C	-	0
B	2	1
A	2	2
E	5	1
D	5	2

D to	Link	Distance
D	-	0
A	3	1
B	3	2
E	6	1
C	6	2

A, D, and C create new DVs, send them, but they have no impact.

Processing of Received Distance Vectors



Note: this is simplified, shows only the principle!

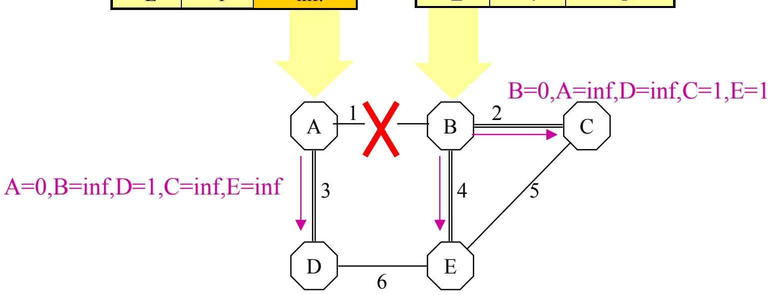
A link breaks...

A round of updates starts on link failure

A gives an infinite distance to the nodes reached through link 1

A to	Link	Distance
A	-	0
B	1	inf.
D	3	1
C	1	inf.
E	1	inf.

B to	Link	Distance
B	-	0
A	1	inf.
D	1	inf.
C	2	1
E	4	1



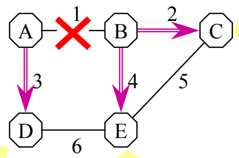
D, E and C update their routing tables

B=0,A=inf,D=inf,C=1,E=1

A=0,B=inf,D=1,C=inf,E=inf

+1

A=1,B=inf,D=2,C=inf,E=inf



D to	Link	Distance
D	-	0
A	3	1
B	3	inf
E	6	1
C	6	2

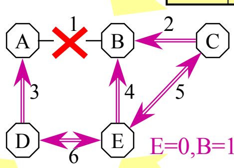
C to	Link	Distance
C	-	0
B	2	1
A	2	inf
E	5	1
D	5	2

E to	Link	Distance
E	-	0
B	4	1
A	4	inf
D	6	1
C	5	1

D, C, E generate their distance vectors...

A to	Link	Distance
A	-	0
B	1	inf.
D	3	1
C	3	3
E	3	2.

B to	Link	Distance
B	-	0
A	1	inf
D	4	2
C	2	1
E	4	1



C=0,B=1,A=inf,E=1,D=2

D=0,A=1,B=inf,E=1,C=2

E=0,B=1,A=inf,D=1,C=1

D to	Link	Distance
D	-	0
A	3	1
B	6	2
E	6	1
C	6	2

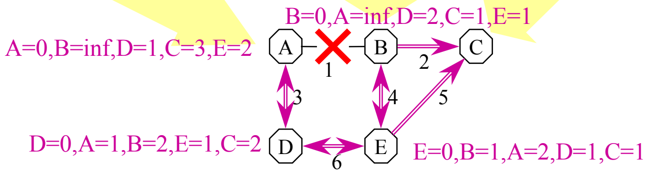
E to	Link	Distance
E	-	0
B	4	1
A	6	2
D	6	1
C	5	1

A, B, D, E generate their distance vectors

A to	Link	Distance
A	-	0
B	3	3
D	3	1
C	3	3
E	3	2.

B to	Link	Distance
B	-	0
A	4	3
D	4	2
C	2	1
E	4	1

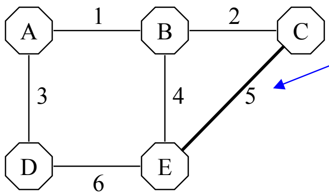
C to	Link	Distance
C	-	0
B	2	1
A	5	3
E	5	1
D	5	2



The result is that all nodes are able to communicate with all other nodes again.

Routing loops

The DV-protocol may create a transient routing loop

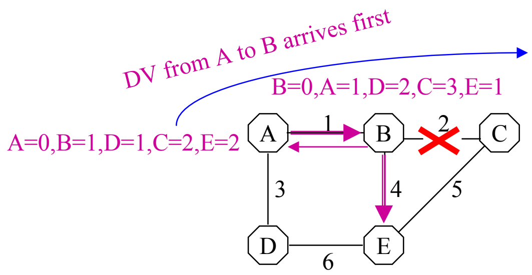


Let's assume that cost of link 5 is 8. A stable initial state for routes to C would be:

x to C	Link from x	Distance
A->C	1	2
B->C	2	1
C->C	-	0
D->C	3	3
E->C	4	2

Let's just look at the first link of each route.

Link 2 fails



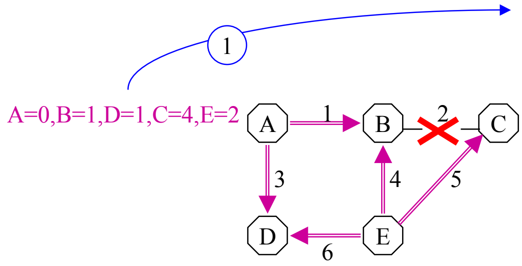
x to C	Link from x	Distance
A->C	1	2
B->C	2	inf
C->C	-	0
D->C	3	3
E->C	4	2

Intermediate state

x to C	Link from x	Distance
A->C	1	4
B->C	1	3
C->C	-	0
D->C	3	3
E->C	4	4

All packets to C are sent to B. B sends them to A. A sends them back to B… until TTL=0. (Bouncing effect)

A and E send their distance vectors



x to C	Link rom x	Distance
A->C	1	4
B->C	1	5
C->C	-	0
D->C	3	5
E->C	4	4

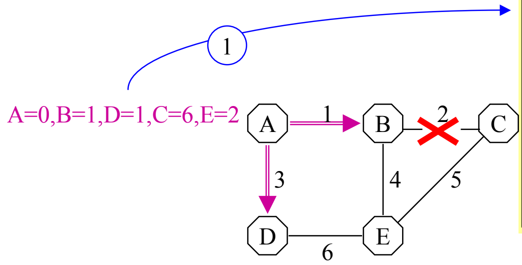
2

B generates a new DV:
B=0,A=1,D=2,C=5,E=1

⇒ Distance seen by A to C grows to 6

Distance vectors sent by C do not change anything because of high link cost

A sends a new distance vector



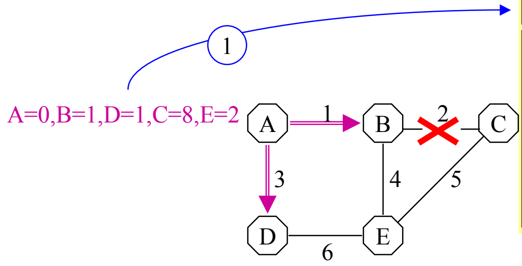
x to C	Link from x	Distance
A->C	1	6
B->C	1	7
C->C	-	0
D->C	3	7
E->C	4	6

2

B generates a new DV
B=0,A=1,D=2,C=7,E=1

⇒ Distance seen by A to C grows to 8

A sends a new distance vector



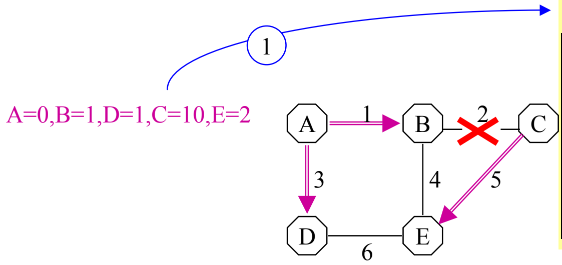
x to C	Link from x	Distance
A->C	1	8
B->C	1	9
C->C	-	0
D->C	3	9
E->C	4	8

2

B generates a new DV
B=0,A=1,D=2,C=9,E=1

⇒ Distance seen by A to C grows to 10

A sends a new distance vector



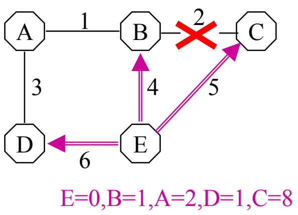
x to C	Link from x	Distance
A->C	1	10
B->C	1	11
C->C	-	0
D->C	3	11
E->C	5	8

2

B generates a new DV
B=0,A=1,D=2,C=11,E=1

E generates a new DV
E=0,B=1,A=2,D=1,C=8

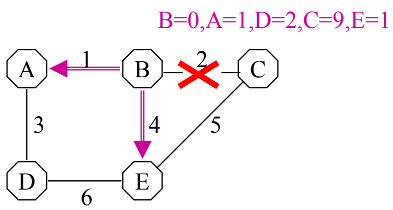
E sends a new distance vector



x to C	Link from x	Distance
A->C	1	10
B->C	4	9
C->C	-	0
D->C	6	9
E->C	5	8

B send its DV but the tables are already OK

x to C	Link from x	Distance
A->C	1	10
B->C	4	9
C->C	-	0
D->C	6	9
E->C	5	8

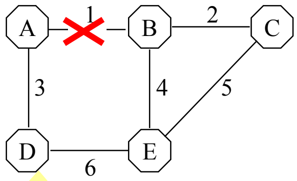


- Each update round improved the costs by 2
- The process progresses in a random order, because it is genuinely parallel in nature.
- During the process, the state of the network is bad. DV-packets may be lost due to the overload created by bouncing user messages

Counting to infinity occurs when failures break the network to isolated islands (1)

- Linkki 1 is broken, and the network has recovered.
- All link costs = 1

A to	Link	Distance
D	3	1
A	-	0
B	3	3
E	3	2
C	3	3

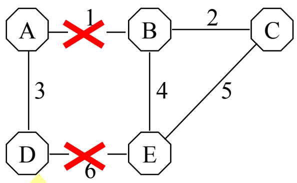


D to	Link	Distance
D	-	0
A	3	1
B	6	2
E	6	1
C	6	2

Counting to infinity occurs when failures break the network to isolated islands (2)

- Also link 6 breaks.
- D has not yet sent its distance vector.

A to	Link	Distance
D	3	1
A	-	0
B	3	3
E	3	2
C	3	3

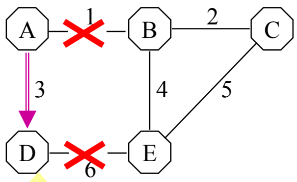


D to	Link	Distance
D	-	0
A	3	1
B	6	inf
E	6	inf
C	6	inf

Counting to infinity occurs when failures break the network to isolated islands (3)

- A sends its distance vector first.
  A=0,B=3,D=1,C=3,E=2
- D adds the information sent by A into its routing table.

A to	Link	Distance
D	3	1
A	-	0
B	3	3
E	3	2
C	3	3

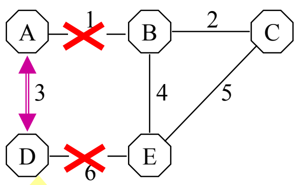


D to	Link	Distance
D	-	0
A	3	1
B	3	4
E	3	3
C	3	4

Counting to infinity occurs when failures break the network to isolated islands (4)

- The result is a loop. Costs are incremented by 2 on each round.
- An agreement is needed: Cost greater than any route cost is = inf.

A to	Link	Distance
D	3	1
A	-	0
B	3	5
E	3	4
C	3	5



D to	Link	Distance
D	-	0
A	3	1
B	3	4
E	3	3
C	3	4

The first method to avoid loops is to send less information

<u>The split horizon rule:</u>

If node A sends to node X through node B, it does not make sense for B to try to reach X through A

⇒ A should not advertise to B its short distance to X

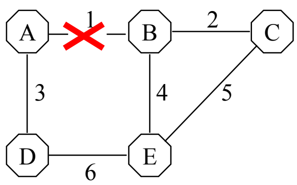
<u>Implementation choices:</u>

1. Split horizon
   - A does not advertise its distance to X towards B at all
   - ⇒ the loop of previous example can not occur

2. Split horizon with poisonous reverse
   - A advertises to B: X=inf.
   - ⇒ two node loops are killed immediately

Three node loops are still possible (1)

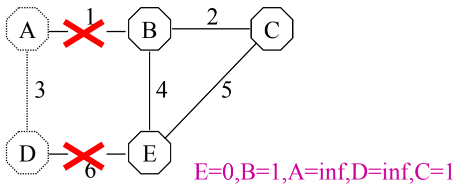
- Linkki 1 is broken, and the network has recovered.
- All link costs = 1



x to D	Link from x	Distance
B→D	4	2
C→D	5	2
E→D	6	1

Three node loops are still possible (2)

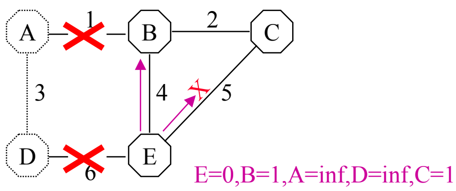
- Also link 6 fails.
- E sends its distance vector to B and C
  E=0,B=1,A=inf,D=inf,C=1



x to D	Link	Distance
B→D	4	2
C→D	5	2
E→D	6	inf

Three node loops are still possible (3)

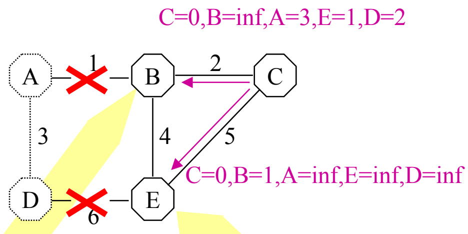
- Also link 6 fails.
- E sends its distance vector to B and C
  E=0,B=1,A=inf,D=inf,C=1
- ... But the DV sent to C is lost



x to D	Link from x	Distance
B→D	4	inf
C→D	5	2
E→D	6	inf

Three node loops are still possible (4)

- Now C sends its poisoned DV

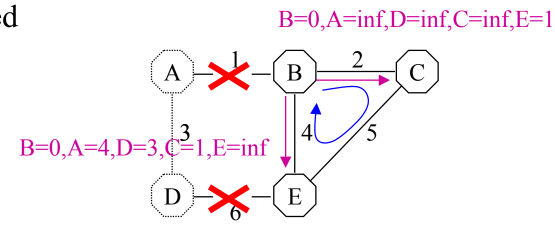


B to	Link	Distance
B	-	0
A	2	4
D	2	3
C	2	1
E	4	1

E to	Link	Distance
B	4	1
A	6	inf
D	6	inf
C	5	1
E	-	0

Three node loops are still possible (5)

- B generates its poisoned distance vectors
- The three node loop is ready
- Routes to D do not change except that the costs keep growing, nodes count to infinity.This finally breaks the loop: on link 5 cost=4 is advertised. C's knowledge about the distance to D grows ...



x to D	Link from x	Distance
B→D	2	3
C→D	5	2
E→D	4	4

When should a DV-protocol advertise

Time of advertisement is a compromise:

- \+ immediate delivery of change info
- \+ recovery from packet loss
- \+ need to monitor the neighbors
- \- sending all changes at the same time
- \- traffic load created by the protocol

\+ = Faster

\- = Slower

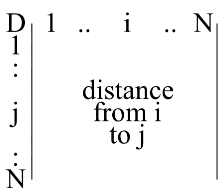
The second method to avoid loops is to use triggered updates

- Entries in the routing tables have refresh and obsolescence timeouts.
- RIP advertises when the refresh timer expires and when a change occurs in an entry (=triggered update).
- Triggered updates reduce the probability of loops
- Loops are still possible, e.g. because of packet loss
- Triggered updates speed up counting to infinity

The Bellman-Ford algorithm

Bellman-Ford algorithm (1)

- DV-protocols are based on the Bellman-Ford algorithm
- Centralized version:
  1. Let N be the number of nodes and M the number of links.
  2. L is the link table with M rows, $L[l].m$ - link cost
     $L[l].s$ - link source
     $L[l].d$ - link destination
  3. D is $N \times N$ matrix, such that $D[i,j]$ is the distance from i to j
  4. H on $N \times N$ matrix, such that $H[i,j]$ is the link i uses to send to j



Both directions are presented separately in the Link table!

A column ≡ DV of the corresponding node

Bellman-Ford algorithm (2)

- Initialized distance and link matrices

D	1	..	i	..	N
1	**0**	∞	∞	∞	∞
:	∞	**0**	∞	∞	∞
j	∞	∞	**0**	∞	∞
:	∞	∞	∞	**0**	∞
N	∞	∞	∞	∞	**0**

Distance matrix D

H	1	..	i	..	N
1	-1	-1	-1	-1	-1
:	-1	**-1**	-1	-1	-1
j	-1	-1	**-1**	-1	-1
:	-1	-1	-1	**-1**	-1
N	-1	-1	-1	-1	**-1**

Link matrix H

NB: *Link vector has both directions of a link separately.*
First in D-matrix appear one hop link distances, second two hop link distances etc.

Bellman-Ford algorithm (3)

1. Initialization: If $i=j$, then D[i,j] = 0, else D[i,j] = inf.
   Initialize ∀ H[i,j] = -1. *(previous slide)*
2. ∀ l and ∀ destinations k set i = L[l].s, j = L[l].d and calculate d = L[l].m + D[j,k]
3. If d < D[i,k], set D[i,k] = d; H[i,k] = l.
4. If at least one D[i,k] changed, go to step 2, else stop.

Bellman-Ford algorithm (4)

- Number of steps ≤ N
- Complexity $O(M \cdot N^2)$

RIP protocol

RIP-protocol properties (1)

- Simple protocol. Used before standardization.
- RIP version 1
  - RFC 1058 in 1988
- RIP is used inside an autonomous system
- RIP works both on shared media (Ethernet) and in point-to-point networks.
- RIP runs on top of UDP and IP.

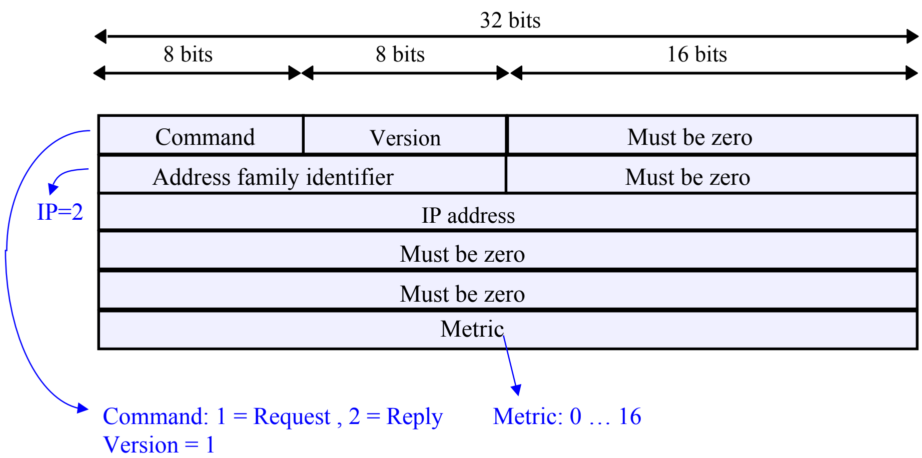
RIP-protocol properties (2)

- An entry in the routing table represents a host, a network or a sub-net
  - <netid,subnetid,host> represents a host (used only in exceptional cases)
  - <netid,subnetid,0> represents a sub-net
  - <netid,0,0> represents a network
  - <0.0.0.0> represents a route out from the autonomous system
- The mask must be manually configured.
- Information sent to the neighboring subnet is aggregated.

RIP-protocol properties (3)

- Distance = hop count = number of links on a path (route).
  - No other metrics
- Distance 16 = infinite.
- RIP advertises once in 30s.
  - If an entry is 180s old ⇒ distance is set to inf
  - Advertisements must be randomized to avoid bursts of RIP updates. 1-5 s.
- RIP also sends 1-5 s after an update (triggered updates).
- RIP uses poisoned vectors.

RIP message format

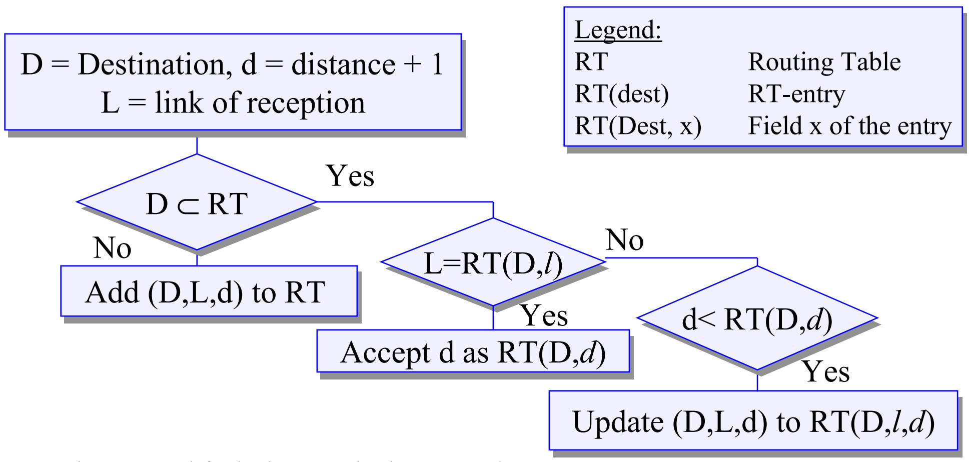


RIP routing table

A routing table entry contains

- Destination IP address
- Distance to destination
- Next hop IP address
- "Recently updated" flag
- Several timers (refresh, obsolescence...)

Processing of Received Distance Vectors



Note: this is simplified, shows only the principle!

RIP response messages

- Distance vectors are sent in reply messages
- 30 seconds period
  - All routing table entries
  - Different DV on different links because of poisoned vectors
  - More than 25 entries ⇒ several messages
- Triggered updates after changes
  - Contains changed entries
  - 1-5 seconds delay, so that the message contains all updates that are related to the same change
- Destinations with infinite distance can be omitted if the next hop is same as before.

RIP request messages

- The router can request routing tables from its neighbors at startup
  - Complete list
  - Response similar to normal update (+ poisoned vectors)
- Partial routing table
  - For debugging
  - No poisoned vectors

Silent nodes

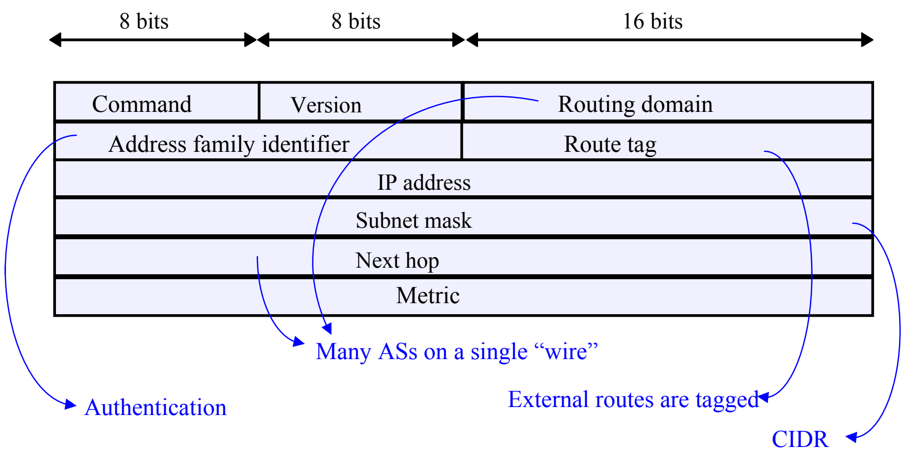
- When only RIP was used, hosts could listen to routing traffic and maintain their own routing tables
  - Which router is closest to the destination?
  - Which link, if several available?
- These where ”silent nodes”, that only listened to routing traffic without sending
- Nowadays there are too many routing protocols
  - RIP-2, OSPF, IGRP, ...

Historical

RIP version 2

- RFC-1388 (1387,1389)
- Why?
  - Simple and lightweight alternative to OSPF and IS-IS
- RIP-2 is a partially interoperable update with v1
  - RIP-1 router understand some of what a RIP-2 router is saying.
- Improvements
  - Authentication
  - Support for CIDR
  - Next hop –field
  - Subnet mask
  - External routes
  - Updates with multicast

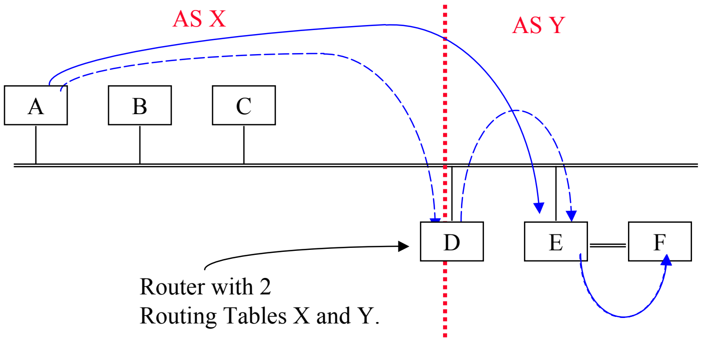
RIP version 2 - messages



Routing from one sub-net to another

- In RIP-1 the sub-net mask is not known outside the sub-net, only netid is sent in an advertisement out from a sub-net
  - ⇒ A host and a sub-net can not be distinguished
  - ⇒ All sub-nets must be interconnected with all other sub-nets and exterior traffic is received in the nearest router independent of the final destination inside our AS
- RIP-2 corrects the situation by advertising both the sub-net and the sub-net mask
  - Masks of different length within a network
  - CIDR
  - RIP-1 does not understand

Routing domain and next hop



Next hop ⇒ D advertises in X: the distance to F is f and the next hop is E!

Multicast support

- RIP-1 broadcasts advertisements to all addresses on the wire
  - Hosts must examine all broadcast packets
- RIP-2 uses a multicast address for advertisements
  - 244.0.0.9
  - No real multicast support needed, since packets are only sent on the local network
- Compatibility problems between RIP-1 and RIP-2

Observations about RIP

- Routers have a spontaneous tendency to synchronize their send times. This increases the probability of losses in the net. Therefore, send instants are randomized between 15s ... 45s.
  - Reason: send interval = constant + time of message packing + processing time of messages that are in the queue.
- When RIP is used on ISDN links a new call is established per 30s
  ⇒ Expensive.
- Slow network ⇒ queue length are restricted. RIP sends its DVs 25 entries/message in a row ⇒ RIP messages may be lost.
- A correction proposal: ack all DVs: no periodic updates
  ⇒ If there are no RIP message: assume that neighbor is alive and reachable
  ⇒ Info on all alternative routes is stored.