Routing Information Protocol

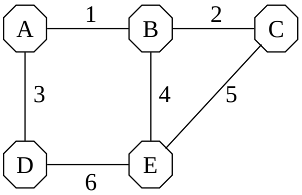
Distance Vector Routing Principles

Routing loops and countermeasures to loops

Bellman-Ford route calculations

RIP

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



Example network with nodes A, B, C …
Link numbers: 1, 2, 3 …
In a real network no global
link numbering exists.

RIP is a distance vector protocol.

Let us study DV protocol principles:

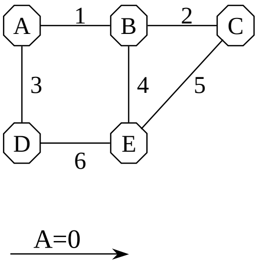
Initial state: Nodes know their own addresses and interfaces, no more.

Node A creates its routing table:

From A to Node	Link	Cost
A	local	0

The corresponding 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

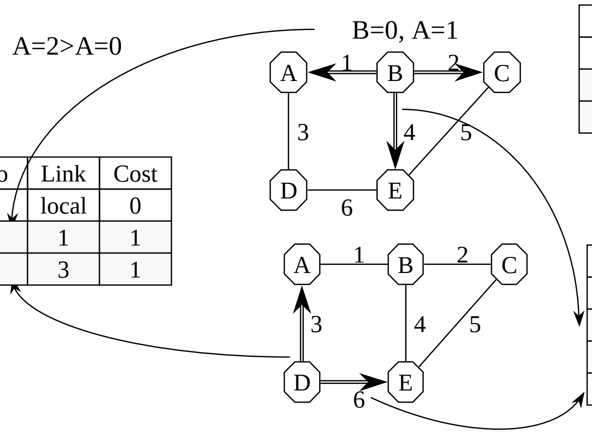
From B to Node	Link	Cost
B	local	0

1. B increments the DV + 1 ⇒ A=1 and
2. B looks for the result in its routing table, no match
3. B adds the result to its RT, result is

New entry→

From B to Node	Link	Cost
B	local	0
A	1	1

B creates its own DV and sends it to all neighbors



A=2>A=0

B=0, A=1

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

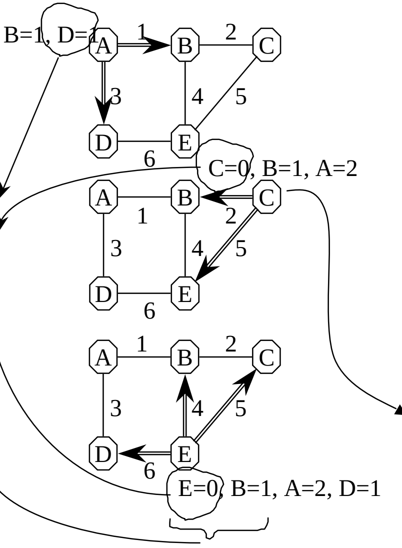
A to	Link	Cost
A	local	0
B	1	1
D	3	1

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

D=0, A=1

A=2 == A=2 ⇒ ei huomioida

Nodes with changed RTs create DVs and send them to neighbors

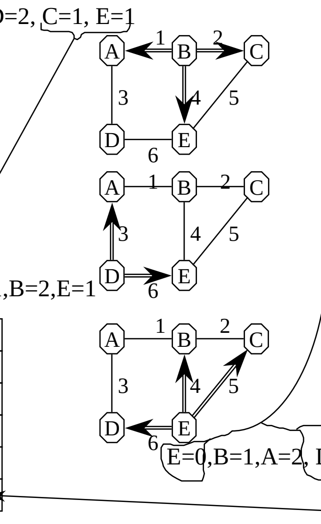


B to	Link	Cost
B	-	0
A	1	1
D	1	2
C	2	1
E	4	1

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

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

Again the changes are sent ...



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

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

D to	Link	Cost
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.

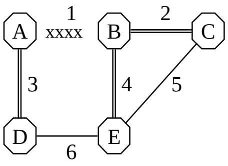
A round of updates starts on link failure

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

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

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

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

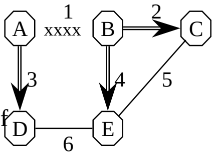


D, E ja C update their Routing Tables

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

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

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



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

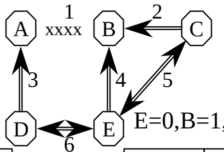
D to	Link	Cost
D	-	0
A	3	1
B	3	inf
E	6	1
C	6	2

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

D, C, E generate Distance Vectors...

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

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



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

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

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

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

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

A, B, D, E generate their Distance Vectors

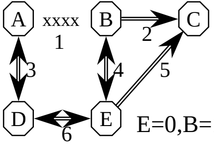
A to	Link	Cost
A	-	0
B	3	3
D	3	1
C	3	3
E	3	2

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

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

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

A=0,B=inf,D=1,C=3,E=2

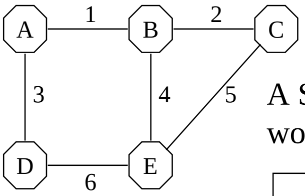


D=0,A=1,B=2,E=1,C=2

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

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

DV -protocol may create a transient routing loop



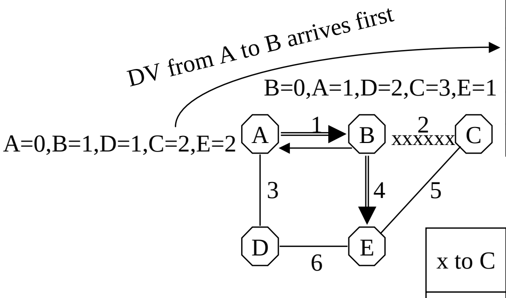
Let’s assume that cost of loop 5 is 8.

A Stable initial state for routes to C would be:

x to C	Link from x	Cost
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...



DV from A to B arrives first

B=0,A=1,D=2,C=3,E=1

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

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

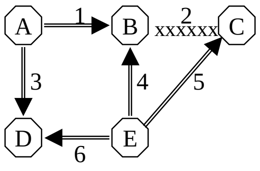
Intermediate state

x to C	Link from x	Cost
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 - pallottelu)

A and E send their Distance Vectors

A=0,B=1,D=1,C=4,E=2



① ②

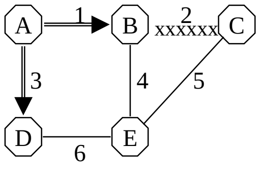
x to C	Link from x	Cost
A->C	1	4
B->C	1	5
C->C	-	0
D->C	3	5
E->C	4	4

B generates a new DV:

B=0,A=1,D=2,C=5,E=1
--> Distance seen by A to C grows to 6

A sends a new Distance Vector

A=0,B=1,D=1,C=6,E=2



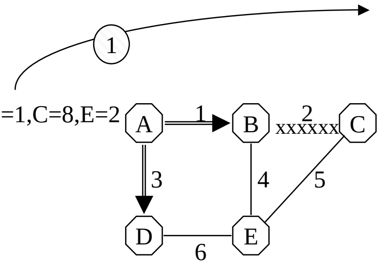
① ②

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

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 DV



A=0,B=1,D=1,C=8,E=2

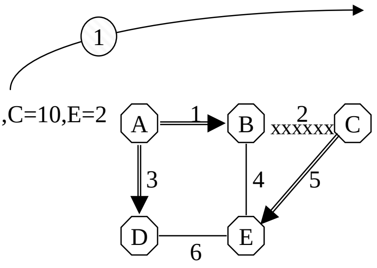
x to C	Link from x	Cost
A->C	1	8
B->C	1	9
C->C	-	0
D->C	3	9
E->C	4	8

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 DV again...



A=0,B=1,D=1,C=10,E=2

x to C	Link from x	Cost
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:

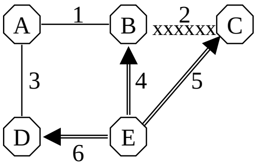
B=0,A=1,D=2,C=11,E=1

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	Cost
A->C	1	10
B->C	4	9
C->C	-	0
D->C	6	9
E->C	5	8

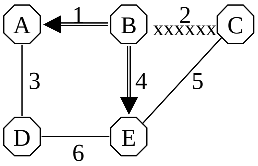


E=0,B=1,A=2,D=1,C=8

B send its DV but the Tables are already OK

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

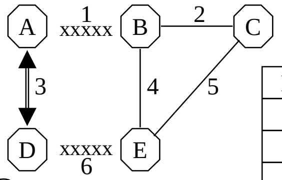
B=0,A=1,D=2,C=9,E=1



- 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 beak the network to isolated islands

When the network has recovered from failure of link 1, also link 6 breaks. All link costs =1.



② A=0,B=3,D=1,C=3,E=2

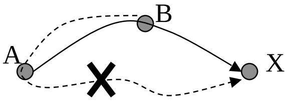
D to	Link	Cost
D	-	0
A	3	1
B	6	inf
E	6	inf
C	6	inf

① D is late in sending its DV

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

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

Loops can be often avoided if less info is sent and by generating DVs immediately on RT change



Split horizon rule = If node A sends to node X thru node B, it does not make sense for B to try to reach X thru A

⇒

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

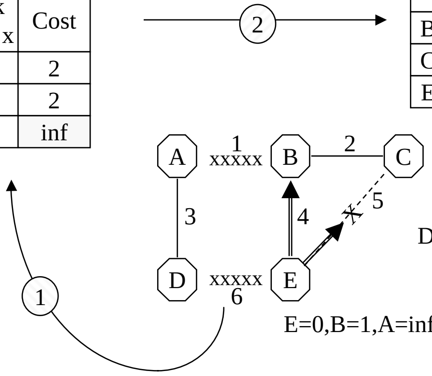
Implementation choices:

1. A does not advertise its distance to X towards B at all
   ⇒ the loop of previous example can not occur
2. A advertises to B: X=inf. (“split horizon with poisonous reverse”)
   ⇒ two node loops are killed immediately.

Three node loops are still possible

x to D	Link from x	Cost
B->D	4	2
C->D	5	2
E->D	6	inf

x to D	Link from x	Cost
B->D	4	inf
C->D	5	2
E->D	6	inf



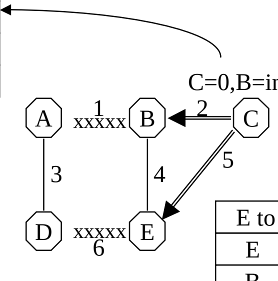
DV to C is lost

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

Now C sends its poisoned DV

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

C=0,B=inf,A=3,E=1,D=2



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

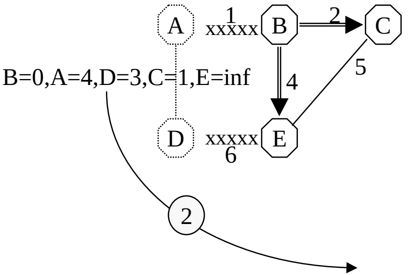
E to	Link	Cost
E	-	0
B	4	1
A	6	inf
D	6	inf
C	5	1

No change

Three node loop is ready...

① B generates its poisoned Distance Vectors

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



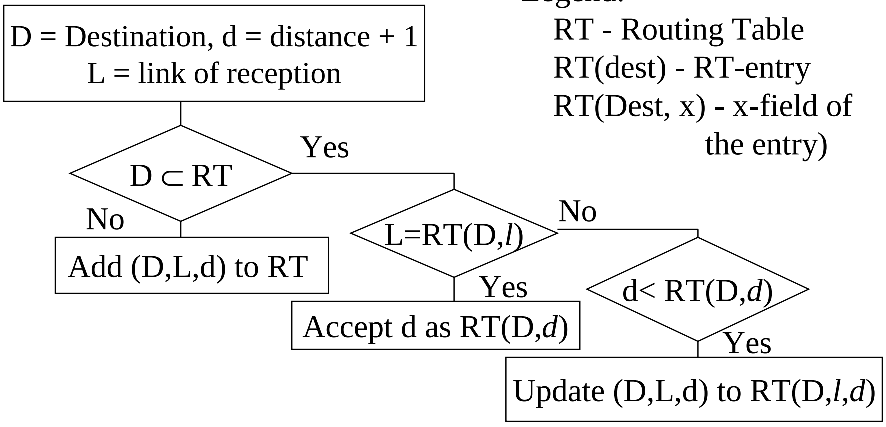
B=0,A=4,D=3,C=1,E=inf

x to D	Link from x	Cost
B->D	2	3
C->D	5	2
E->D	4	4

③ 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 ...

Received Distance Vectors are processed so

Legend:
RT - Routing Table
RT(dest) - RT-entry
RT(Dest, x) - x-field of the entry)



D = Destination, d = distance + 1
L = link of reception

D ⊂ RT

Yes

No

Add (D,L,d) to RT

L=RT(D,*l*)

No

Yes

Accept d as RT(D,*d*)

d< RT(D,*d*)

Yes

Update (D,L,d) to RT(D,*l*,*d*)

Note: this is simplified, shows only the principle!

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

Event triggered updates improve the functioning of RIP

- 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 updates speed up counting to infinity and reduce the probability of loops.

DV-protocols are based on the Bellman-Ford algorithm

Centralized version:
1. Let N be the nrof nodes and M the nrof links.
2. L is the link table with M rows, L[*l*].m - link cost,
   L[*l*].s - link source
   L[*l*].d - link sink
3. D is N × N matrix, such that D[*i*,*j*] is the distance from *i* to *j*
4. H on N × N matrix, such that H[*i*,*j*] is the link *i* uses to send to *j*

$$\begin{array}{c|ccccc|}
D & 1 & .. & i & .. & N \\
1 & & & & & \\
: & & & \text{Distance} & & \\
j & & & \text{from } i \text{ to } j & & \\
: & & & & & \\
N & & & & &
\end{array}$$

Both direction are presented separately in the Link table!
A Column ≡ DV of the corresponding node!

Initialized Distance and Link matrices are

$$\begin{array}{c|ccccc|}
D & 1 & .. & & .. & N \\
1 & \mathbf{0} & \infty & \infty & \infty & \infty \\
: & \infty & \mathbf{0} & \infty & \infty & \infty \\
 & \infty & \infty & \mathbf{0} & \infty & \infty \\
: & \infty & \infty & \infty & \mathbf{0} & \infty \\
N & \infty & \infty & \infty & \infty & \mathbf{0}
\end{array}
\qquad
\begin{array}{c|ccccc|}
H & 1 & .. & & .. & N \\
1 & \mathbf{-1} & -1 & -1 & -1 & -1 \\
: & -1 & \mathbf{-1} & -1 & -1 & -1 \\
 & -1 & -1 & \mathbf{-1} & -1 & -1 \\
: & -1 & -1 & -1 & \mathbf{-1} & -1 \\
N & -1 & -1 & -1 & -1 & \mathbf{-1}
\end{array}$$

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 is

1. Initialization: If $i=j$, then D[i,j] = 0, else D[i,j] = inf. Initialize $\forall$ H[i,j] = -1.
2. $\forall$ l and $\forall$ 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, GOTO 2, else END.

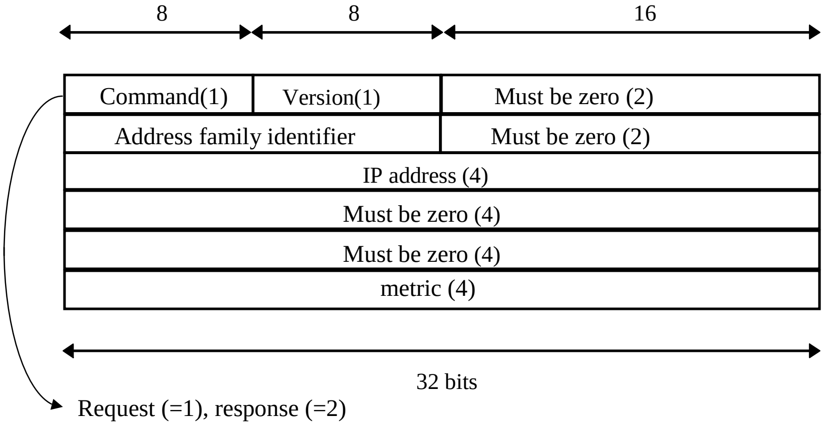
RIP protocol properties include

- RIP version 1 - RFC-1058
- RIP is used inside Autonomous Systems
- An Entry in the Routing Table represents a host, a network or a sub-net
  <netid,subnetid,host> represents a host
  <netid,subnetid,0> represents a sub-net
  <netid,0,0> represents a network
  <0.0.0.0> represents a route from the Autonomous System
- Distance = hop count = Nrof links on a path (route), 16=inf.

More RIP properties….

- RIP works both in shared media (Ethernet) and in point-to-point networks
- RIP runs on top of UDP and IP.
- RIP advertises once in 30s, if an entry is 180s old
  --> distance is set to inf
- Timer triggered advertisements must be randomnized to avoid bursts of RIP updates.
- RIP uses poisoned vectors
- Advertisement to neighbor sub-nets are aggregated

RIP message format is



8	8	16
Command(1)	Version(1)	Must be zero (2)
Address family identifier		Must be zero (2)
IP address (4)		
Must be zero (4)		
Must be zero (4)		
metric (4)		

32 bits

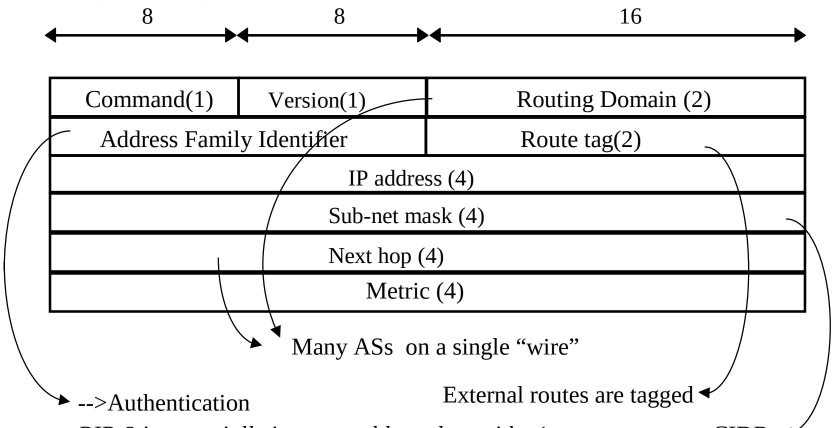
Request (=1), response (=2)

RIP Routing Table Entry will have

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

RIP version 2 - messages

RFC-1388 (1387,1389)



8

8

16

Command(1)

Version(1)

Routing Domain (2)

Address Family Identifier

Route tag(2)

IP address (4)

Sub-net mask (4)

Next hop (4)

Metric (4)

Many ASs on a single “wire”

-->Authentication

External routes are tagged

CIDR

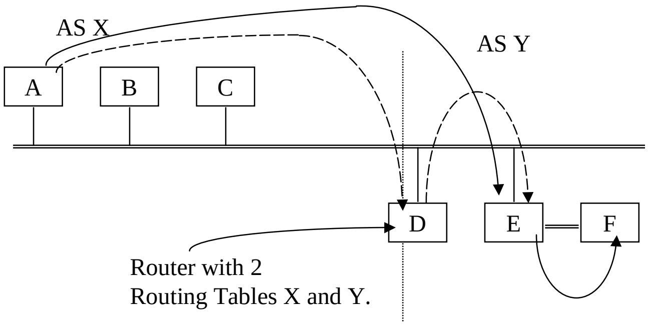
RIP-2 is a partially interoperable update with v1
(RIP-1 router understand some of what a RIP-2 router is saying.)

Routing from one sub-net to another

- In RIP-1 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

---

Routing Domain and Next hop



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

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/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.