Advanced Computer Networks

Internal routing - distance vector protocols

Prof. Andrzej Duda

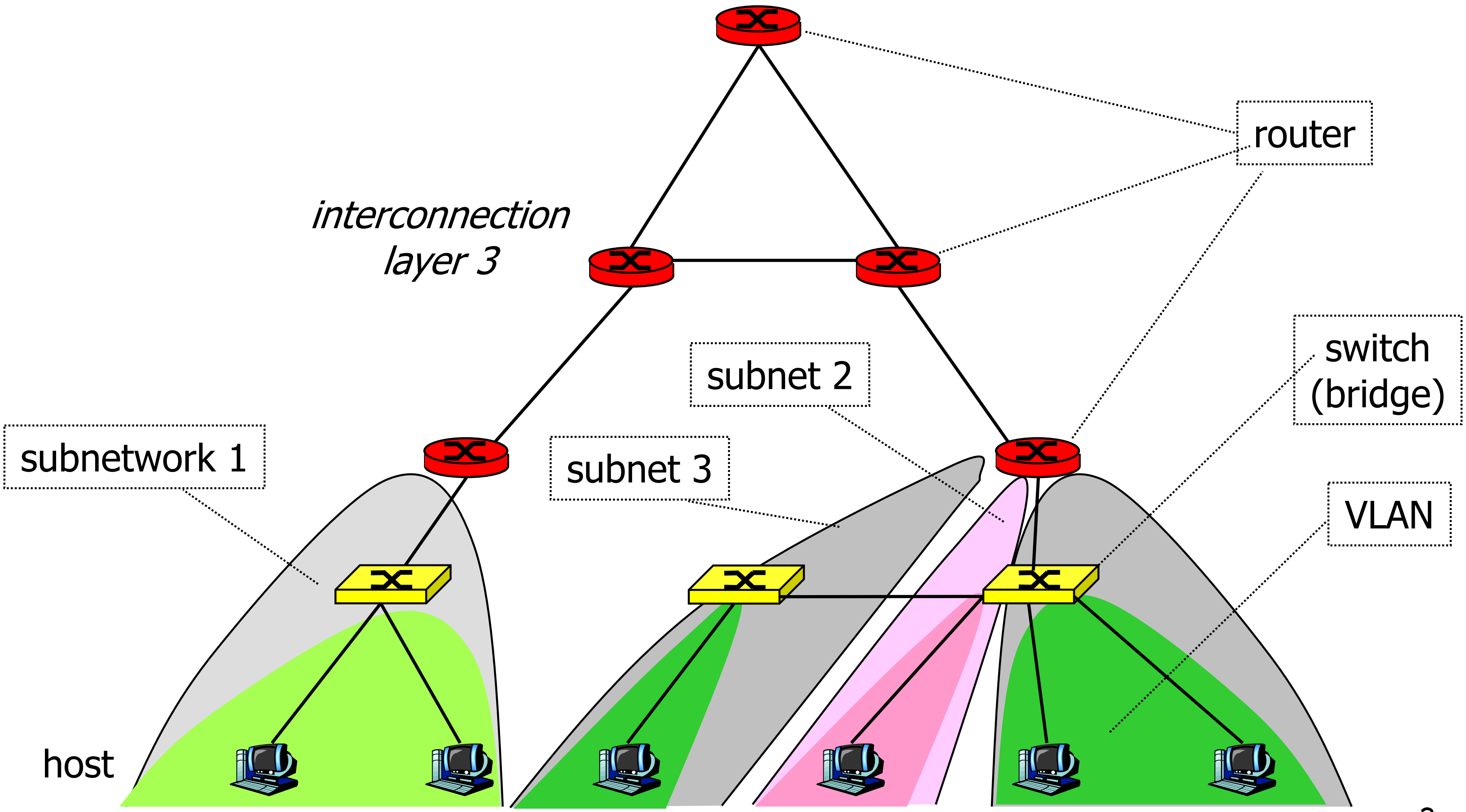
duda@imag.fr

http://duda.imag.fr

Contents

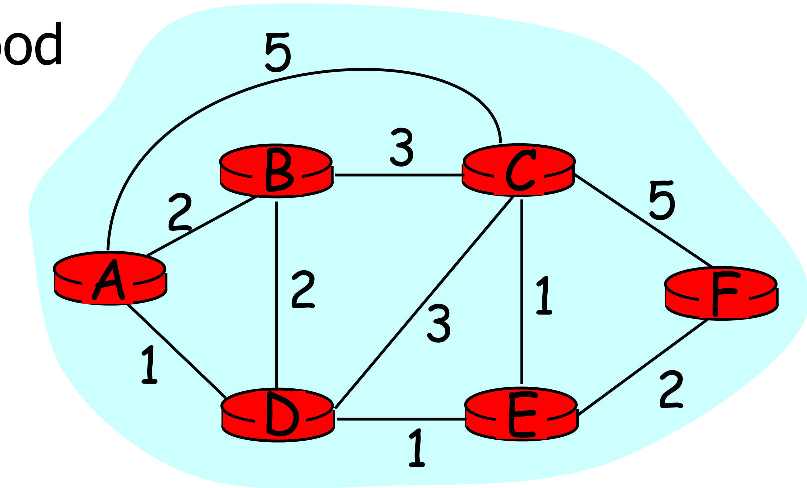
- Principles of internal routing
  - construct Routing Tables
- Distance vector (Bellman-Ford)
  - principles
  - case of link failures
  - count to infinity
  - split horizon
- RIP
- RIP v2
- IGRP

Interconnection structure



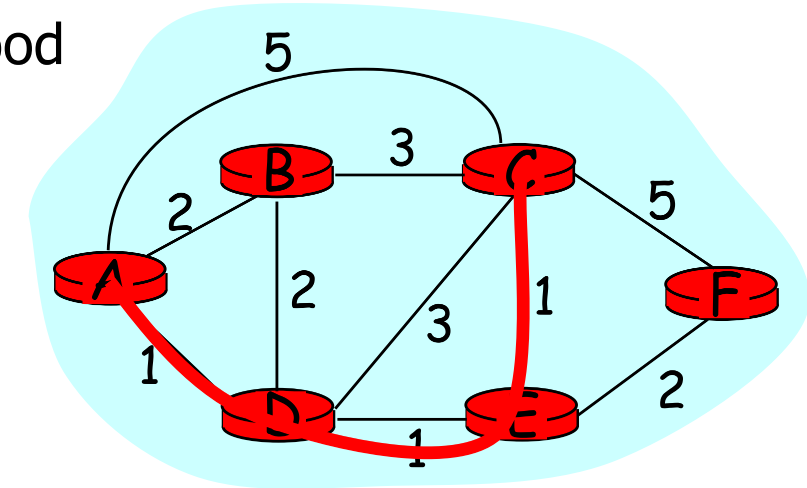
Routing algorithms

- Problem
  - find the **best** route to a destination
- What does it mean the best?
  - metric to measure how a route is good
  - hops
  - delay (inverse of link capacity)
  - performance measures:
    - link load, instantaneous delay
  - cost
- Graph optimization - Shortest Path
  - graph nodes are routers, graph edges are physical links
  - link cost: delay, $ cost, or congestion level
  - find the **shortest** path in a graph



Routing algorithms

- Problem
  - find the **best** route to a destination
- What does it mean the best?
  - metric to measure how a route is good
  - hops
  - delay (inverse of link capacity)
  - performance measures:
    - link load, instantaneous delay
  - cost
- Graph optimization - Shortest Path
  - graph nodes are routers, graph edges are physical links
  - link cost: delay, $ cost, or congestion level
  - find the **shortest** path in a graph



Main algorithms

- Distance Vector (Bellman-Ford)
  - routers only know their local state
    - link metric and neighbor estimates
  - internal routing protocols (RIP, IGRP)
- Link State
  - knowledge of the global state
    - metrics of all links
    - global optimization (Shortest Path First - Dijkstra)
  - internal routing protocols (OSPF)
- Path Vector
  - similar to DV – announce prefixes
    - path: sequence of AS with attributes
    - coarse-grain optimization and policy routing
  - external routing protocols (BGP)

Metrics

- Static - do not depend on the network state
  - number of hops
  - static delay (inverse of capacity)
  - cost
- Dynamic - depend on the network state
  - link load
  - current delay

Flooding

- Simple and robust routing
  - no need for routing tables
  - each packet duplicated on each outgoing link
  - packet duplication
  - duplicated packets destroyed at destination
  - robust - tolerates link or router failures
  - optimal in some sense
    - the first packet has found the shortest path to the destination
    - cannot be compared to the shortest path calculated by Link State - no packet duplication
- Problem
  - increased load due to packet duplication
- Used in OSPF to distribute link state information and in ad hoc routing protocols (AODV, OLSR)

Distance vector

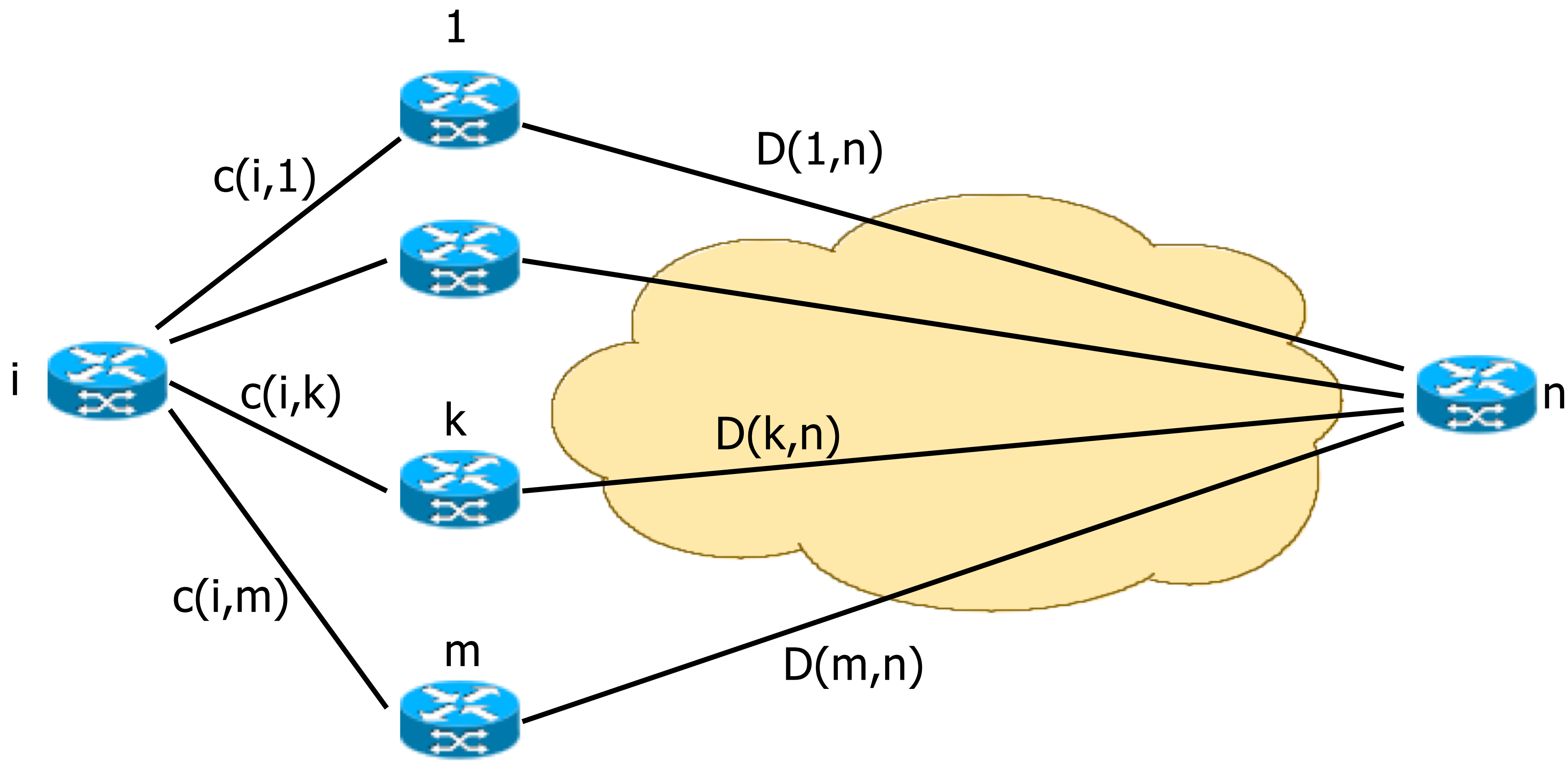
- Dynamic routing based on distributed estimation of the distance to the destination
  - uses the distributed algorithm by Bellman-Ford (dynamic programming)
  - each router receives aggregated information from its neighbors
  - estimates the local cost to its neighbors
  - computes the best routes
  - no global network states
- Distance
  - number of hops
  - delay

Bellman-Ford algorithm

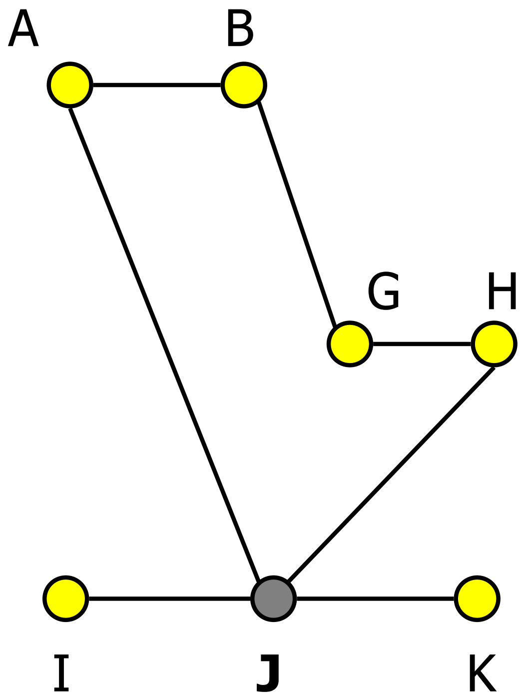
- Bellman-Ford algorithm
  - node *i* knows cost *c(i,k)* to its immediate neighbours (+∞ for most values of *k*)
  - distance *D(i,n)* is given by: $D(i,n) = \min_k (c(i,k) + D(k,n))$
  - in the worst case, convergence after *N*-1 iterations
- Distributed Bellman-Ford algorithm
  - initially: *D(i,n)* = 0 if *i* directly connected to *n* and *D(i,n)* = +∞ otherwise
  - node *i* receives from neighbour *k* latest values of *D(k,n)* for all *n* (distance vector)
  - node *i* computes the best estimates

$$D(i,n) = \min_k (c(i,k) + D(k,n))$$

Bellman-Ford algorithm



Example of Bellman-Ford



	A	I	H	K	Table of J	
A	0	24	20	21	8	A
B	12	36	31	28	20	A
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
J:	8	10	12	6		

computation of G : 18+8=26, 31+10=41, 6+12=18, 6+31=37

→ choice of 18, H

Distance vector example

- Simple network
  - routers connected by links
  - destinations = subnetworks connected to routers
  - symmetric links
  - cost = number of hops

Initialization

A

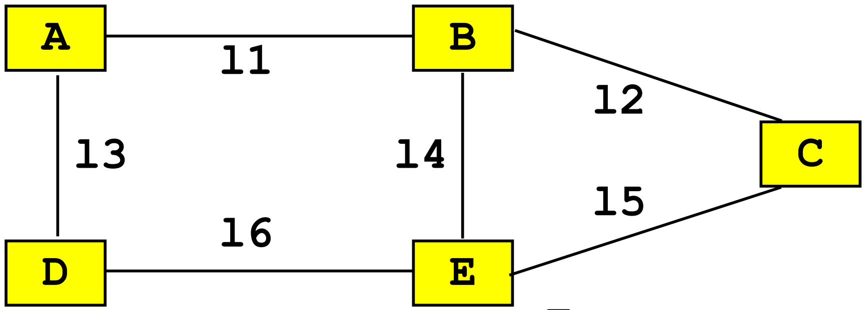
dest	link	cost
A	local	0

B

dest	link	cost
B	local	0

C

dest	link	cost
C	local	0



D

dest	link	cost
D	local	0

E

dest	link	cost
E	local	0

Distance vector announcement

A

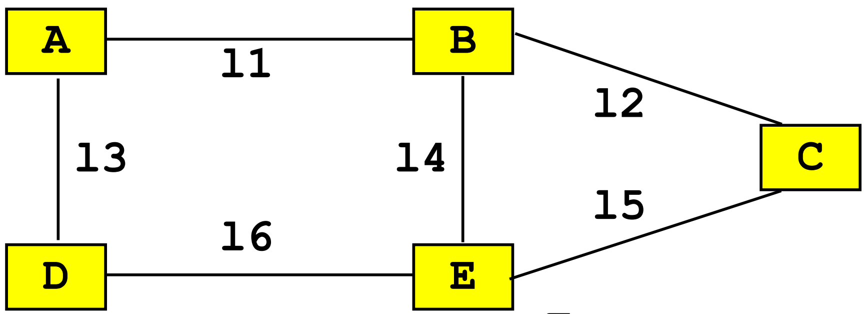
dest	link	cost
A	local	0

from A:
dest cost
A 0

B

dest	link	cost
B	local	0
A	11	1

from A:
dest cost
A 0



C

dest	link	cost
C	local	0

D

dest	link	cost
D	local	0
A	13	1

E

dest	link	cost
E	local	0

Distance vector announcement

A

dest	link	cost
A	local	0
B	11	1

from B:

dest	cost
A	1
B	0

B

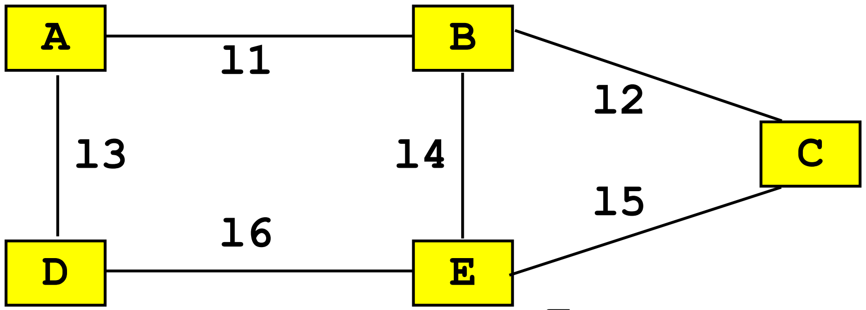
dest	link	cost
B	local	0
A	11	1

from B:

dest	cost
A	1
B	0

C

dest	link	cost
C	local	0
A	12	2
B	12	1



D

dest	link	cost
D	local	0
A	13	1

from B:

dest	cost
B	0
A	1

E

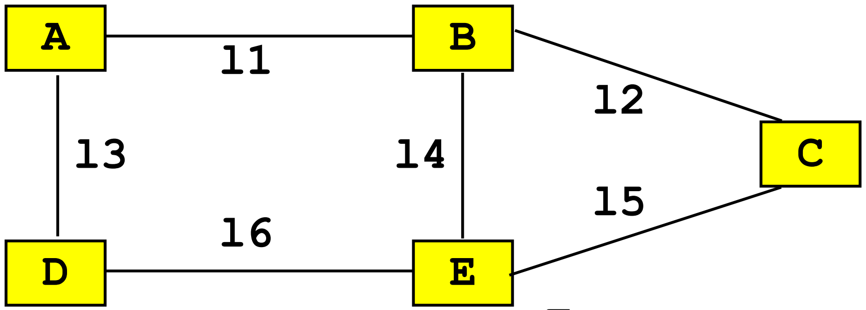
dest	link	cost
E	local	0
A	14	2
B	14	1

Distance vector announcement

A

dest	link	cost
A	local	0
B	11	1
D	13	1

from D:
dest cost
D 0
A 1



B

dest	link	cost
B	local	0
A	11	1

C

dest	link	cost
C	local	0
A	12	2
B	12	1

D

dest	link	cost
D	local	0
A	13	1

from D:
dest cost
D 0
A 1

E

dest	link	cost
E	local	0
A	14	2
B	14	1
D	16	1

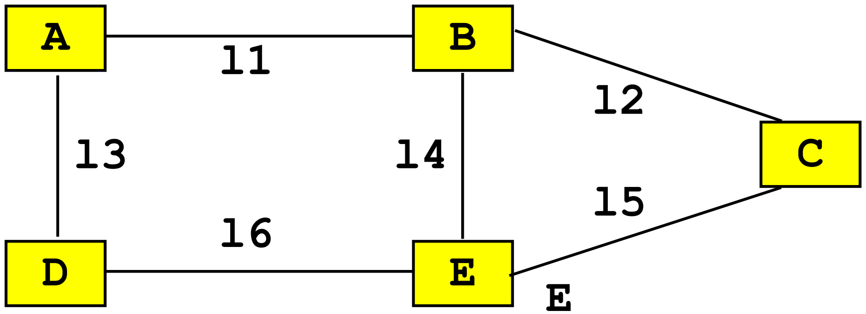
Final

A

dest	link	cost
A	local	0
B	l1	1
D	l3	1
C	l1	2
E	l1	2

B

dest	link	cost
B	local	0
A	l1	1
C	l2	1
E	l4	1
D	l1	2



C

dest	link	cost
C	local	0
A	l2	2
B	l2	1
D	l5	2
E	l5	1

D

dest	link	cost
D	local	0
A	l3	1
B	l3	2
C	l6	2
E	l6	1

E

dest	link	cost
E	local	0
A	l4	2
B	l4	1
D	l6	1
C	l5	1

Link failure

A

dest	link	cost
A	local	0
B	11	∞
D	13	1
C	11	∞
E	11	∞

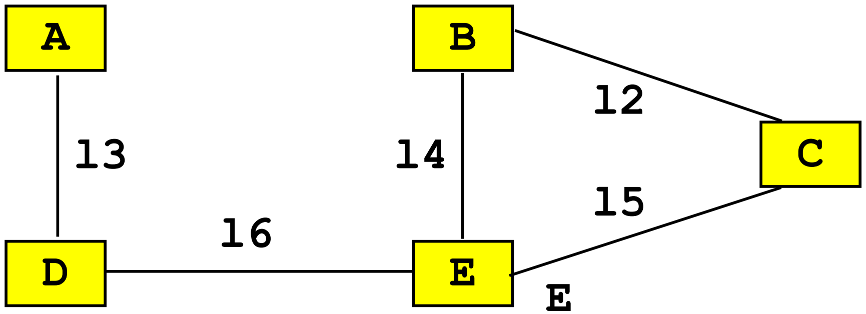
B

dest	link	cost
B	local	0
A	11	∞
C	12	1
E	14	1
D	11	∞

C

dest	link	cost
C	local	0
A	12	∞
B	12	1
D	15	2
E	15	1

from A:
dest cost
A 0
B,C,E ∞
D 1



D

dest	link	cost
D	local	0
A	13	1
B	13	∞
C	16	2
E	16	1

E

dest	link	cost
E	local	0
A	14	∞
B	14	1
D	16	1
C	15	1

Link failure

A

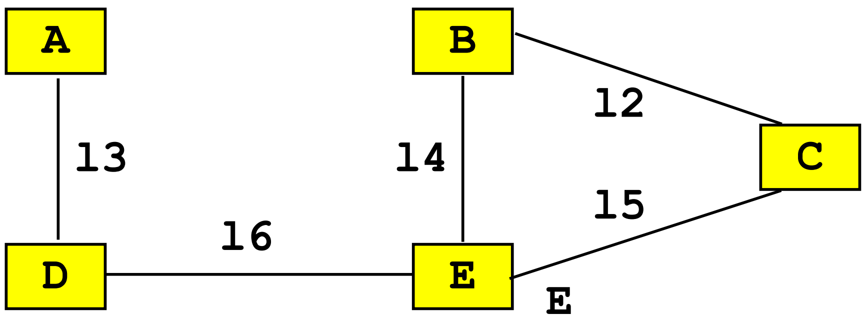
dest	link	cost
A	local	0
B	l1	∞
D	l3	1
C	l1	∞
E	l1	∞

B

dest	link	cost
B	local	0
A	l1	∞
C	l2	1
E	l4	1
D	l1	∞

C

dest	link	cost
C	local	0
A	l2	∞
B	l2	1
D	l5	2
E	l5	1



D

dest	link	cost
D	local	0
A	l3	1
B	l6	2
C	l6	2
E	l6	1

from B:

dest	cost
E	0
A	∞
B,D,C	1

E

dest	link	cost
E	local	0
A	l4	∞
B	l4	1
D	l6	1
C	l5	1

Final state after failure

A

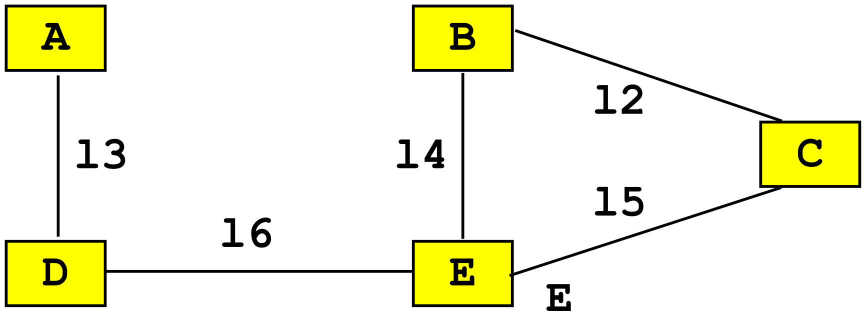
dest	link	cost
A	local	0
B	13	3
D	13	1
C	13	3
E	13	2

B

dest	link	cost
B	local	0
A	14	3
C	12	1
E	14	1
D	14	2

C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1



D

dest	link	cost
D	local	0
A	13	1
B	16	2
C	16	2
E	16	1

E

dest	link	cost
E	local	0
A	16	2
B	14	1
D	16	1
C	15	1

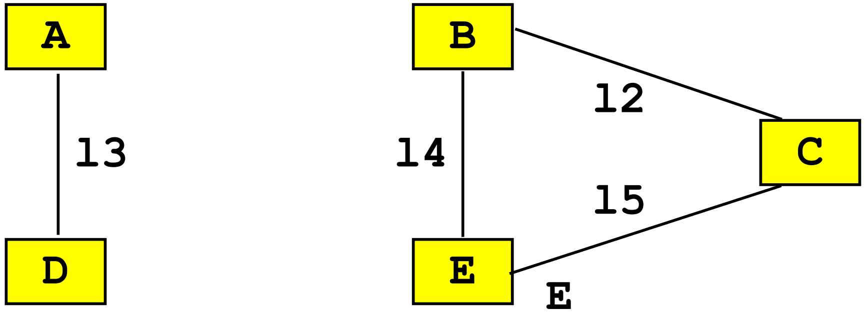
Equal link costs - link failures

A

dest	link	cost
A	local	0
B	13	3
D	13	1
C	13	3
E	13	2

B

dest	link	cost
B	local	0
A	14	3
C	12	1
E	14	1
D	14	2



C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

D

dest	link	cost
D	local	0
A	13	1
B	16	∞
C	16	∞
E	16	∞

E

dest	link	cost
E	local	0
A	16	2
B	14	1
D	16	1
C	15	1

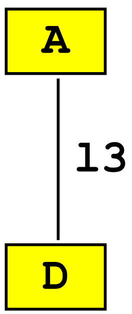
Counting to infinity

A

dest	link	cost
A	local	0
B	13	3
D	13	1
C	13	3
E	13	2

from A:

dest	cost
A	0
B,C	3
D	1
E	2



D

dest	link	cost
D	local	0
A	13	1
B	13	4
C	13	4
E	13	3

- Loop between A and D
- Exchange of routes, costs increase by 2 each cycle
- Convergence to a stable state
  - ∞ = large number
  - e.g. RIP: ∞ = 16

Split horizon

- Minimize the effects of bouncing and counting to infinity
- Rule
  - if A routes packets to X via B, it does not announce this route to B

Example of split horizon

A

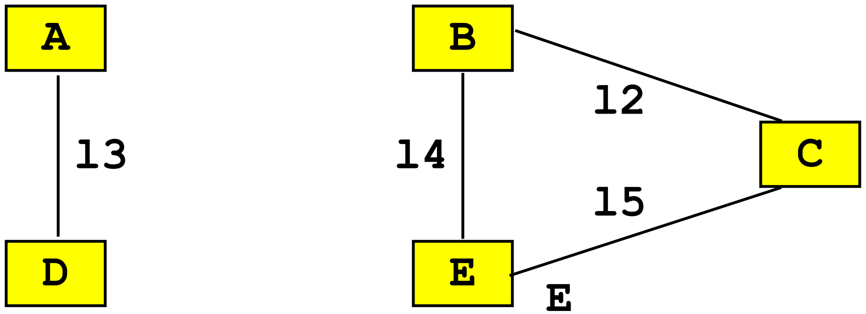
dest	link	cost
A	local	0
B	13	3
D	13	1
C	13	3
E	13	2

B

dest	link	cost
B	local	0
A	14	3
C	12	1
E	14	1
D	14	2

C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1



D

dest	link	cost
D	local	0
A	13	1
B	16	∞
C	16	∞
E	16	∞

E

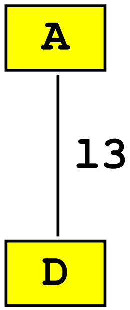
dest	link	cost
E	local	0
A	16	2
B	14	1
D	16	1
C	15	1

Split horizon

A

dest	link	cost
A	local	0
B	13	3
D	13	1
C	13	3
E	13	2

from A:
dest cost
A 0



D

dest	link	cost
D	local	0
A	13	1
B	16	∞
C	16	∞
E	16	∞

- Split horizon cuts the process of counting to infinity

Split horizon

A

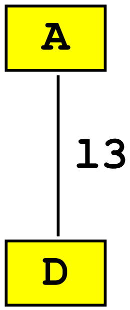
dest	link	cost
A	local	0
B	13	∞
D	13	1
C	13	∞
E	13	∞

from D:

dest cost

D 0

B,C,E ∞



D

dest	link	cost
D	local	0
A	13	1
B	16	∞
C	16	∞
E	16	∞

- Split horizon cuts the process of counting to infinity

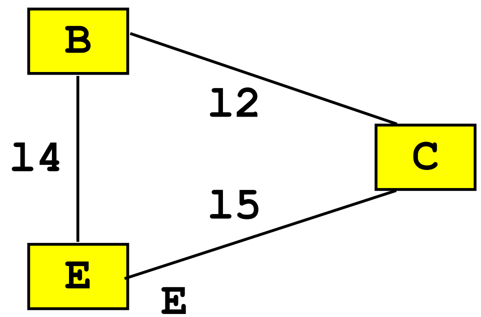
Split horizon may fail

from E:

dest	cost
E	0
A	∞
C	1
D	∞

B

dest	link	cost
B	local	0
A	14	∞
C	12	1
E	14	1
D	14	∞



C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

E

dest	link	cost
E	local	0
A	16	∞
B	14	1
D	16	∞
C	15	1

Split horizon may fail

B

dest	link	cost
B	local	0
A	12	4
C	12	1
E	14	1
D	12	3

from C:

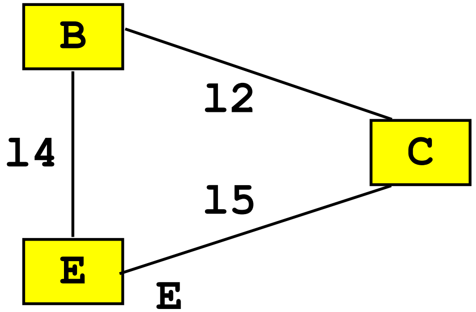
dest	cost
C	0
A	3
D	2
E	1

C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

from C:

dest	cost
C	0
B	1



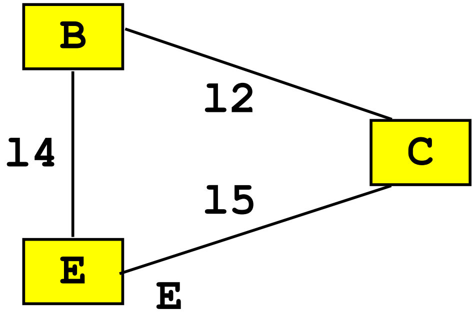
E

dest	link	cost
E	local	0
A	16	∞
B	14	1
D	16	∞
C	15	1

Split horizon may fail

B

dest	link	cost
B	local	0
A	12	4
C	12	1
E	14	1
D	12	3



C

dest	link	cost
C	local	0
A	15	3
B	12	1
D	15	2
E	15	1

from B:

dest	cost
A	4
B	0
C	1
D	3

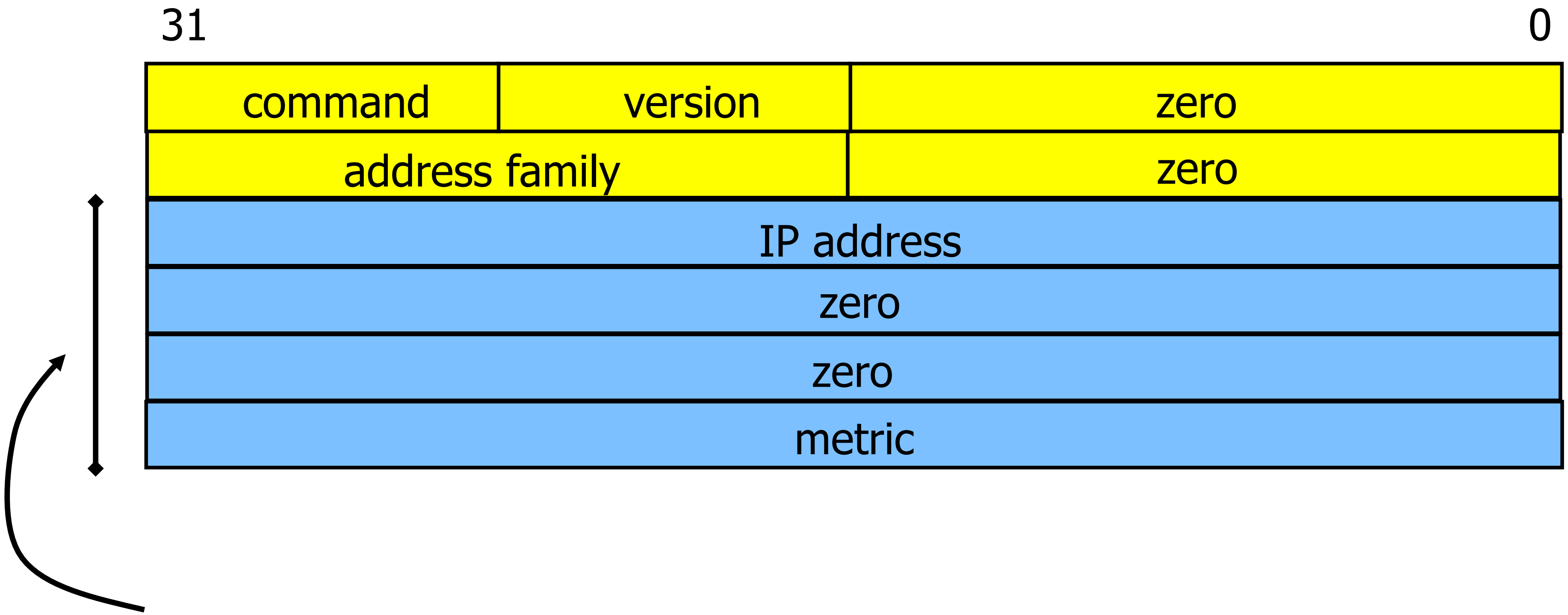
E

dest	link	cost
E	local	0
A	14	5
B	14	1
D	14	4
C	15	1

RIP v1

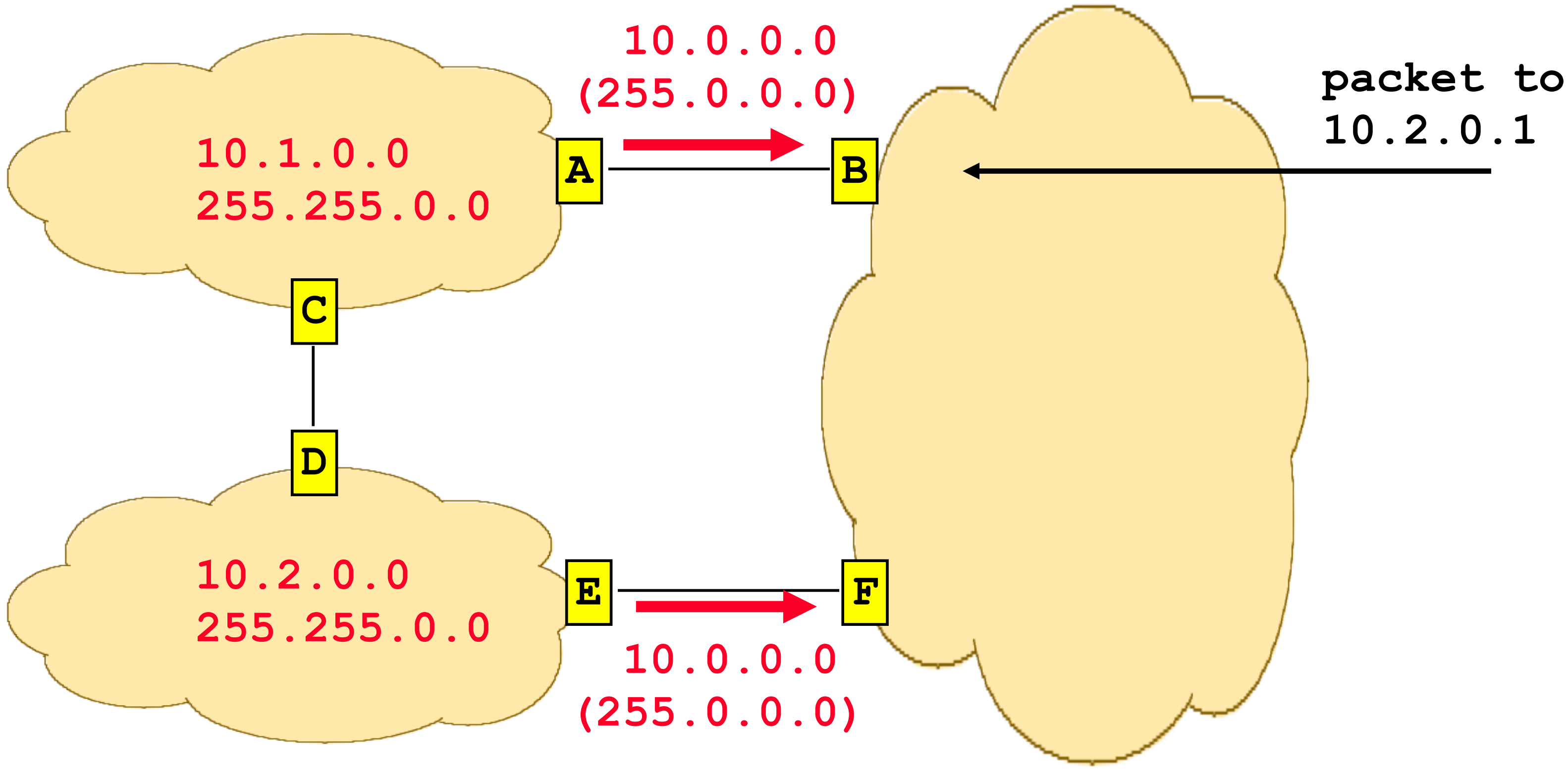
- Distance vector protocol
- Metric - hops
- Network span limited to 15
  - ∞ = 16
- Split horizon
- Destination network identified by IP address
  - no prefix/subnet information - derived from address class
- Encapsulated as UDP packets, port 520
- Largely implemented (**`routed`** on Unix)
- Broadcast every 30 seconds or when update detected
- Route not announced during 3 minutes
  - cost becomes ∞

Message format



- May be repeated 25 times
- Command
  - **REQUEST** - 1 (sent at boot to initialize)
  - **RESPONSE** - 2 (broadcast each 30 sec)

Missing netmask



- A and E can forward to `10.0.0.0`
- Packet to `10.2.0.1` can go through F or B
  - if sent to B, it goes through A and C
- If link C-D broken, no route to destination

RIP v2 (RFC 2453)

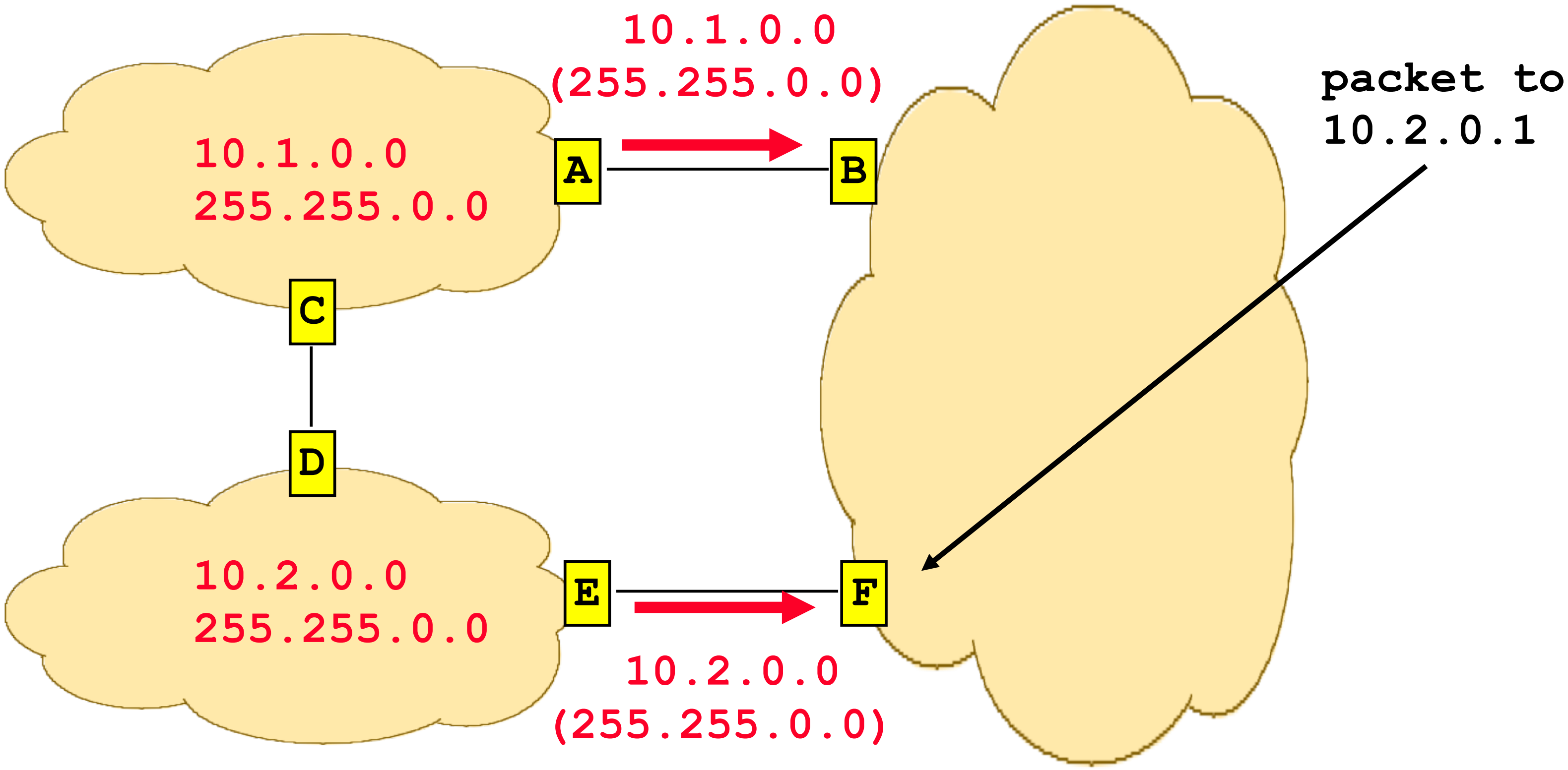
- Subnetworks
  - take into account CIDR prefixes and netmasks
- Authentication
- Multicast
  - **`224.0.0.9`** mapped to **`MAC 01-00-5E-00-00-09`**
  - on LAN only, no need for IGMP

Message format

<table>
<tr><td colspan="2">31</td><td colspan="2" align="right">0</td></tr>
<tr><td>command</td><td>version</td><td colspan="2">unused</td></tr>
<tr><td colspan="2">address family</td><td colspan="2">route tag</td></tr>
<tr><td colspan="4">IP address</td></tr>
<tr><td colspan="4">netmask</td></tr>
<tr><td colspan="4">next router</td></tr>
<tr><td colspan="4">metric</td></tr>
</table>

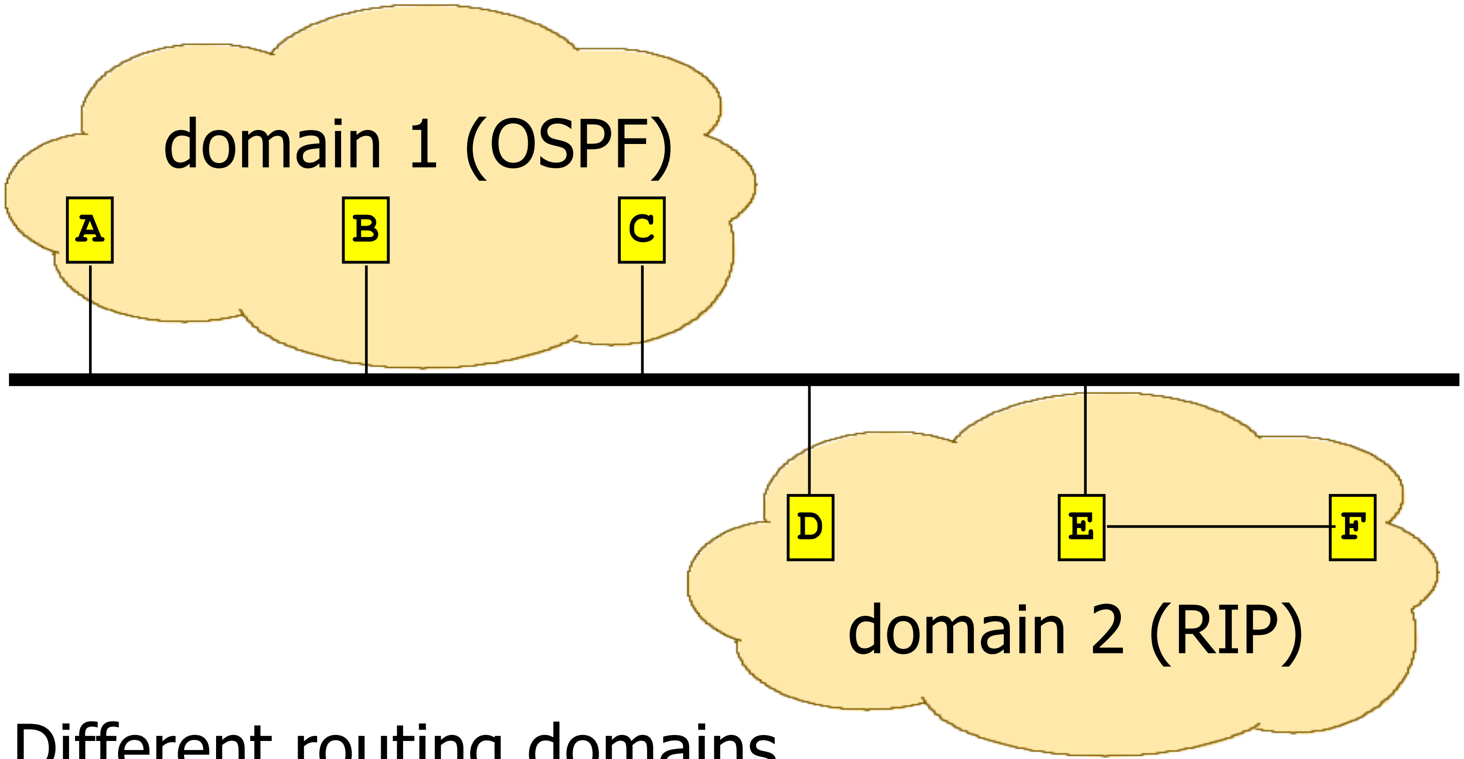
- Command, version unchanged
- One address family - authentication
- Next router
  - used at the border of different routing domains (e.g. RIP and OSPF)
- Route tag
  - for external routes (used by BGP)

Announcing netmasks



- E can forward to **`10.2.0.0`**
- Packet to **`10.2.0.1`** can go through F

Routing domains



- Different routing domains
  - e.g. routers under different administrations that run different routing protocols (RIP, OSPF)
- If A wants to send a packet to F, it goes through D and E
- When announcing F, D adds E as **next router**

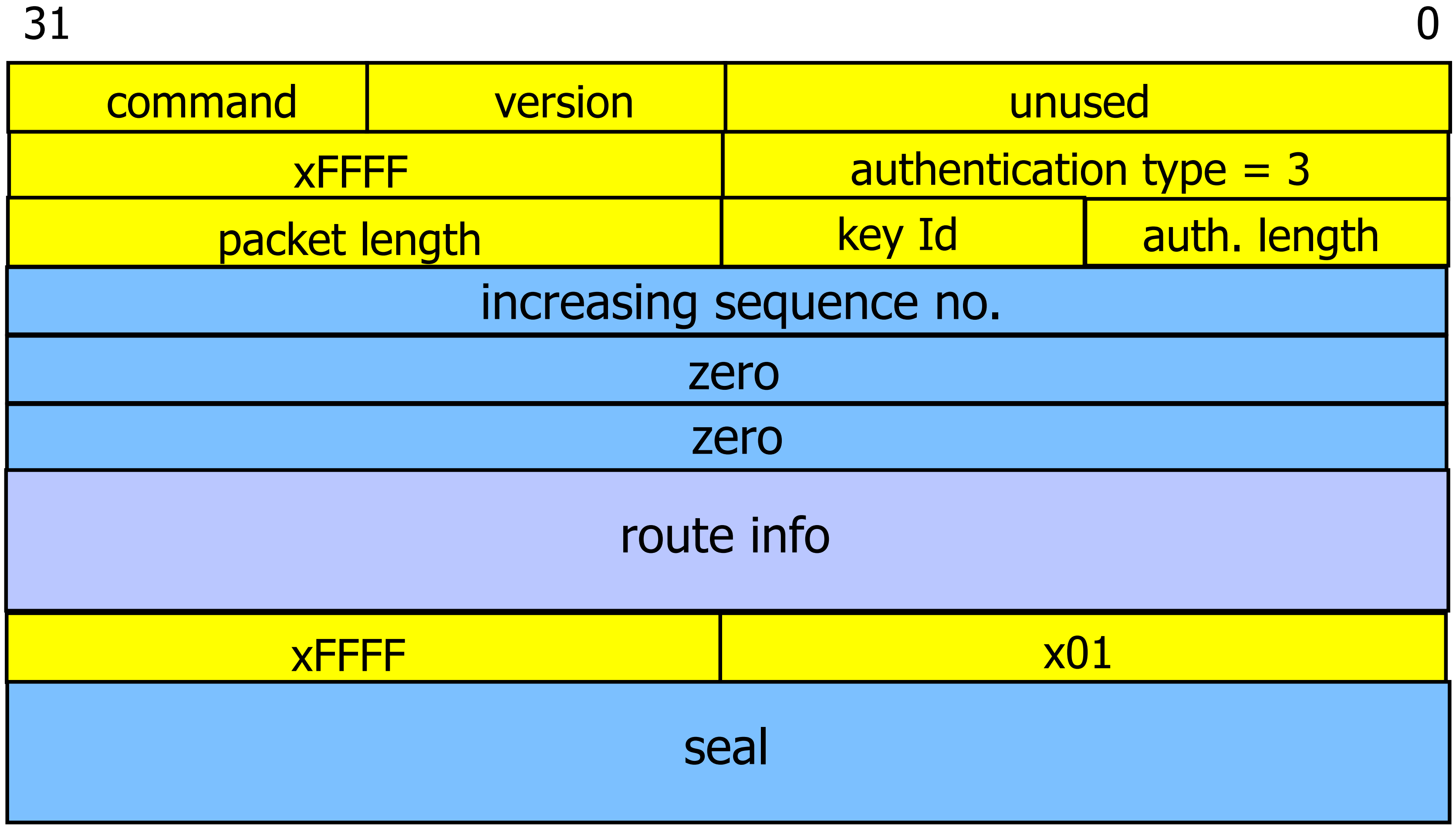
Simple authentication

31		0
command	version	unused
xFFFF		authentication type = 2
password on 16 bytes		

- Configuration of gated (**`/etc/gated.conf`**)

```
rip yes {
        interface all
        version 2 multicast
        authentication simple "qptszwmz"
}
```

MD5 authentication



MD5 authentication

- Seal
  - MD5 digest on the message using a shared secret
  - sequence number avoids replay attacks
- Configuration of gated (**`/etc/gated.conf`**)

```
rip yes {
         interface all
         version 2 multicast
         authentication md5 "qptszwmz"
}
```

IGRP (Interior Gateway Routing Protocol)

- Proprietary protocol by CISCO
- Metric that estimates the global delay
- Maintains several routes of similar cost
  - load sharing
- Takes into account netmasks
- No limit of 15
  - number of routers included in messages
- Broadcast every 90 sec

Metric example

- Metric
  - Trans = 10000000/Bandwidth (time to send 10 Kb)
  - delay = (sum of Delay)/10
  - m = [K_1*Trans + (K_2*Trans)/(256-load) + K_3*delay]
  - default: K1=1, K2=0, K3=1, K4=0, K5=0
  - if K5 ≠ 0, m = m * [K_5/(Reliability + K_4)]
- Bandwidth in Kb/s, Delay in μs
  - At Venus: Route for 172.17/16: Metric = 10000000/784 + (20000+1000)/10 = 14855
  - At Saturn: Route for 12./8: Metric = 10000000/224 + (20000 + 1000)/10 = 46742

Conclusion

- Main distance vector protocols
- Largely deployed (Unix BSD **`routed`**)
- Simplicity
- Slow convergence
- Not suited for large and complex networks
  - Link State protocols should be used instead