



# Routing in the Internet

Jean-Yves Le Boudec

1

## Contents

- A. Introduction
- B. Unicast Routing: Distance Vector
- C. Unicast Routing: Link State
- D. Multicast Routing
- E. The Spanning Tree Algorithm for Bridges
- F. Interdomain Routing: BGP
- G. Load dependent routing

2

## Concepts You Should Know...

after this lecture

- distance vector
- link state
- reverse path multicast
- interior, exterior protocol
- topology database
- count to infinity
- what is OSPF, IGRP, RIP, BGP
- the spanning tree algorithm for bridges

3

## A. Introduction

- Connectionless Network Layer assumes routing tables are maintained at hosts and routers
  - used by **Packet Forwarding**
- Routing = control method
  - maintain routing tables automatically
  - in routers
- At host
  - normally done by default rules
  - plus ICMP redirect
  - in old times: was done also by a routing protocol (RIP)
- LANs connected by bridges operate at layer 2 like connectionless packet forwarders
  - how do they maintain routing information ?

4

## Internal Routing

- ❑ Used inside an administrative domain
  - other methods used to find routes between domains
  
- ❑ **Problem** solved by a routing protocol
  - find reachable destinations
  - find best paths towards destinations
    - best in the sense of some metric
    - in this chapter, best means along shortest path, for some additive metric (number of hops, delay)

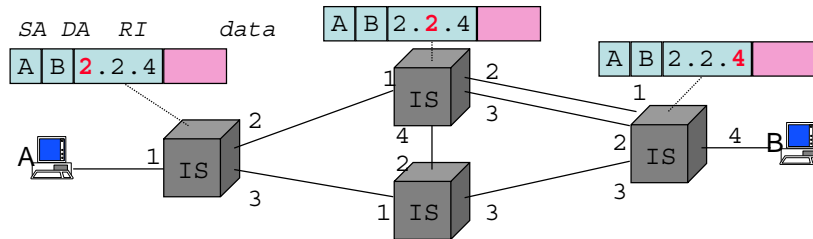
5

## Simple Routing Methods

- ❑ **static configuration**
  - for toy networks only
- ❑ **flooding**
  - each packet duplicated on each outgoing link; loops prevented by packet id or other mechanism ; duplicated packets destroyed at destination
  - simple and robust
    - no need for routing tables
    - robust - tolerates link or router failures
    - optimal in some sense
      - the first packet has found the shortest path to the destination
  - costly
    - many duplicated packets - little useful traffic
  - used as an ingredient by mobile ad-hoc routing methods (AODV, OLSR)
- ❑ **source routing**
  - source writes route into packet header
  - router reads next hop from packet header, moves pointer
  - route discovered by flooding

6

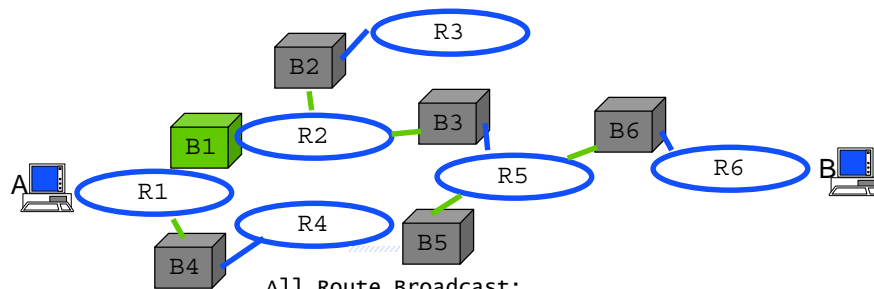
## Source Routing



Loop free routes from A to B: \_\_\_\_\_

7

## Route Discovery in Token Rings



All Route Broadcast:

A-R1-B1-R2-B2-R3

B3-R5-B6-R6

B5-R4

B4-R4-B5-R5-B3-R2-B2-R3

B6-R6

5 frames generated

2 copies reach B route<sub>1</sub> = R1.B1.R2.B3.R5.B6.R6

route<sub>2</sub> = R1.B4.R4.B5.R5.B6.R6

- explorer packet reaches B and accumulates route
- B sends response to A along reverse path
- A now knows a route to B

8

## Other Methods

- ❑ **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
    - topology database
    - global optimization (Shortest Path First - Dijkstra)
  - internal routing protocols (OSPF, PNNI (ATM))
- ❑ **Path vector**
  - no knowledge of the global state
    - path: sequence of AS with attributes
    - global optimization and policy routing
  - external routing protocols (BGP)

9

## Metrics

- ❑ Distance vector and link state find paths that minimize a **metric**
  - Static metric - does not depend on the network state; for example:
    - number of hops
    - link capacity and static delay
    - cost
  - Dynamic metric- depend on the network state
    - link load
    - current delay
    - see end of section

10

## B. Distance Vector

- Computes best paths to all destinations
  - uses distributed Bellman-Ford
    - classical setting:
      - concatenation = addition
      - best = smallest in usual sense
  - each router receives aggregated information from its neighbors
  - individual link cost is setup by network management

11

## The Bellman-Ford Centralized Algorithm

- The original Bellman-Ford theorem computes the best path from  $i$  to  $j$  for any couple  $(i,j)$ .
- Let  $A(i,j)$  be the cost of going from  $i$  to  $j$ , defined for  $i \neq j$ . We assume  $A(i,j) > 0$  and  $A(i,j) = \infty$  for  $i \neq 1$  when  $i$  and  $j$  are not connected.
- Take for example  $j=1$  and define  $p^k(i)$  as the cost of the best path from  $i$  to 1 in at most  $k$  hops. Let  $p^0(1) = 0$ ,  $p^0(i) = \infty$  for  $i \neq 1$ .

### Theorem 1 BF (Bellman Ford)

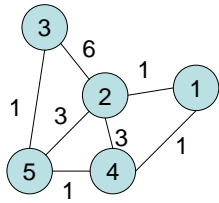
1. For  $k=1,2, \dots$ :  $p^k(i) = \min_{j \neq i} [A(i,j) + p^{k-1}(j)]$  for  $i \neq 1$  and  $p^k(1)=0$
2. If the network is fully connected, for  $k \geq n-1$ :  $p^k(i) = p(i)$  where  $n$  is the number of nodes
3. The shortest path from  $i \neq 1$  to 1 is defined by  $\text{pred}(i) = \text{Argmin}_{j \neq i} [A(i,j) + p(j)]$ .

Proof:  $p^k(i)$  is the distance from  $i$  to 1 in at most  $k$  hops.

12

## Example

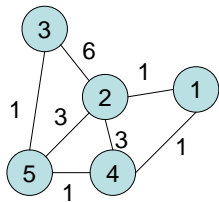
- Apply the theorem: write  $p^k(i)$ ,  $\text{pred}(i)$  and draw the shortest paths to node 1.



13

## Impact of Initial Conditions

- Example: does the algorithm converge to the shortest path with initial condition as shown ?



$i \backslash k$	0	1	2	3	4
1	0				
2	0				
3	0				
4	0				
5	0				

$i \backslash k$	0	1	2	3
1	0			
2	6			
3	1			
4	1			
5	0			

14

## Impact of Initial Condition

### Theorem 2

- The algorithm converges in a finite number of steps to the correct values for all initial conditions such that  $p^0(1)=0$  and for every node  $i$  that is connected to 1
- If there is no path from  $i$  to 1, the algorithm tends to infinity

15

### Proof of Theorem 2

We do the proof assuming all nodes are connected.

1. Let  $p^k$  be the vector  $p^k[i]$ ,  $i=2, \dots$ . Let  $B$  be the mapping that transforms an array  $x[i]_{i=2, \dots}$  into the array  $Bx$  defined for  $i \neq 1$  by

$$Bx[i] = \min_{j \neq i, j \neq 1} [A(i,j) + x(j)]$$

Let  $b$  be the array defined for  $i \neq 1$  by

$$b[i] = A(i,1)$$

The algorithm can be rewritten in vector form as

$$(1) p^k = B p^{k-1} \wedge b$$

where  $\wedge$  is the pointwise minimum

2. Eq (1) is a min-plus linear equation and the operator  $B$  satisfies  $B(x \wedge y) = Bx \wedge By$ .

Thus, Eq(1) can be solved using min-plus algebra into

$$(2) p^k = B^k p^0 \wedge B^{k-1} b \wedge \dots \wedge Bb \wedge b$$

3. Define the array  $e$  for  $i \neq 1$  by  $e[i] = \infty$ . Let  $p^0 = e$ . Eq (2) becomes

$$(3) p^k = B^{k-1} b \wedge \dots \wedge Bb \wedge b. \text{ Now we have the Bellman Ford algorithm with classical initial conditions, thus, by Theorem 1:}$$

$$(4) \text{ for } k \geq n-1: B^{k-1} b \wedge \dots \wedge Bb \wedge b = q$$

where  $q[i]$  is the distance from  $i$  to 1.

4. We can rewrite Eq(2) for  $k \geq n-1$  as

$$(5) p^k = B^k p^0 \wedge q$$

5.  $B^k p^0[i]$  can be written as  $A[i, i_1] + A[i_1, i_2] + \dots + A[i_{k-1}, i_k] + p[i_k]$  thus

(6)  $B^k p^0[i] \geq k a$ , where  $a$  is the minimum of all  $A[i, j]$ . Thus  $B^k p^0[i]$  tends to  $\infty$  when  $k$  grows. Thus for  $k$  large enough,  $B^k p^0$  is larger than and can be ignored in Eq(5). In other words, for  $k$  large enough :

$$(6) p^k = q$$

□

16

## Distributed Bellman Ford

- BF can be used in a centralized algorithm to compute  $p(i)$  i.e. find the spanning tree. However, this is not its main interest, because there is a better algorithm (Dijkstra) that can be used in a centralized method
- But: it can be distributed, as follows.

### Distributed Bellman-Ford Algorithm

- every node, say  $i$ , maintains an estimate  $q(i)$  of  $p(i)$ ; initially,  $q(i) = p^0(i)$ ; also  $\text{pred}(i)=i$  initially;
- whenever  $q(i)$  is modified (or initialized),  $i$  sends the new value  $q(i)$  to all its neighbours
- when node  $i$  receives a new value  $q(j)$  from a neighbour  $j$  it updates  $q(i)$  by  $q(i) := \min(q(i), A(i,j)+q(j))$   
i.e. node  $i$  sees whether  $q(j)$  can be used to provide a better estimate.
- if the received new value improves  $q(i)$ ,  $\text{pred}(i):=j$

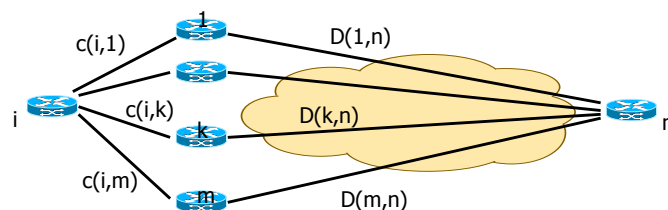
- **Theorem:** if the time to reliably send a message is bounded by  $T$ , the algo converges to the same result as the centralized version in at most  $nT$  time units (if the network is fully connected)

17

## Distributed Bellman-Ford

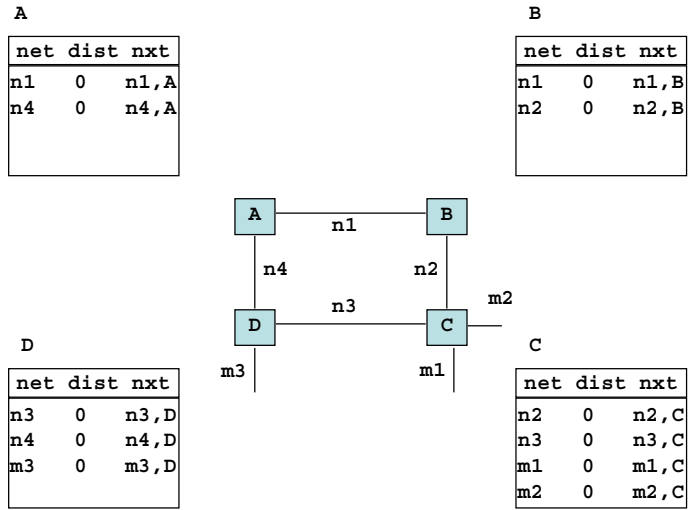
### □ Distributed Bellman-Ford algorithm

- initially:  $D(i,n) = 0$  if  $i$  directly connected to  $n$  and  $D(i,n) = +\infty$  otherwise
- node  $i$  receives from neighbour  $k$  latest values of  $D(k,n)$  for all  $n$  (distance vector)
- node  $i$  computes the best estimates  
$$D(i,n) = \min_k (c(i,k) + D(k,n))$$
- converges if network is stable
  - hello mechanism to reset computation after changes



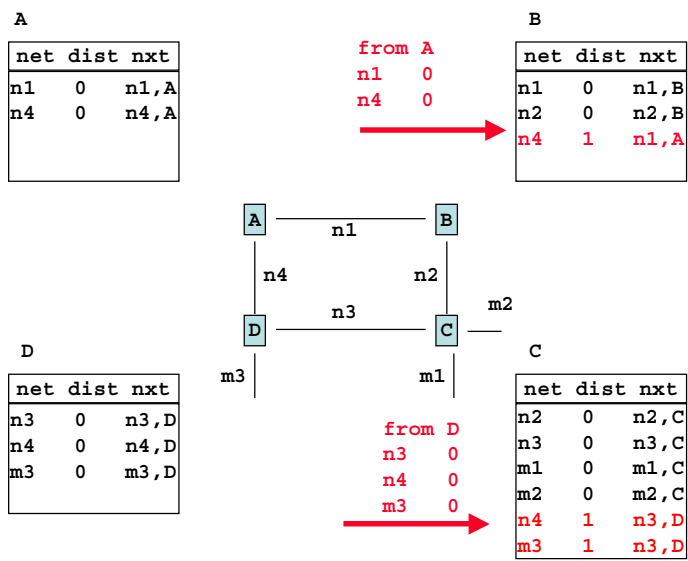
18

# Example 1



19

# Example 1



20

# Example 1

A

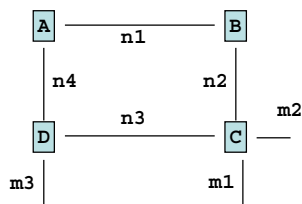
net	dist	nxt
n1	0	n1,A
n4	0	n4,A

B

net	dist	nxt
n1	0	n1,B
n2	0	n2,B
n3	1	n2,C
n4	1	n1,A
m1	1	n2,C
m2	1	n2,C
m3	2	n2,C

D

net	dist	nxt
n3	0	n3,D
n4	0	n4,D
m3	0	m3,D



C

net	dist	nxt
n2	0	n2,C
n3	0	n3,C
m1	0	m1,C
m2	0	m2,C
n4	1	n3,D
m3	1	n3,D

from C  
 ↑  
 n2 0  
 n3 0  
 m1 0  
 m2 0  
 n4 1  
 m3 1

21

# Example 1 - Final

A

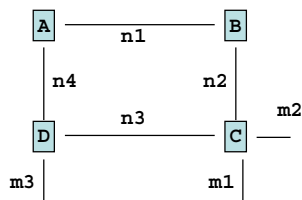
net	dist	nxt
n1	0	n1,A
n2	1	n1,B
n3	1	n4,D
n4	0	n4,A
m1	2	n4,D
m2	2	n4,D
m3	1	n4,D

B

net	dist	nxt
n1	0	n1,B
n2	0	n2,B
n3	1	n2,C
n4	1	n1,A
m1	1	n2,C
m2	1	n2,C
m3	2	n2,C

D

net	dist	nxt
n1	1	n4,A
n2	1	n3,C
n3	0	n3,D
n4	0	n4,D
m1	1	n3,C
m2	1	n3,C
m3	0	m3,D



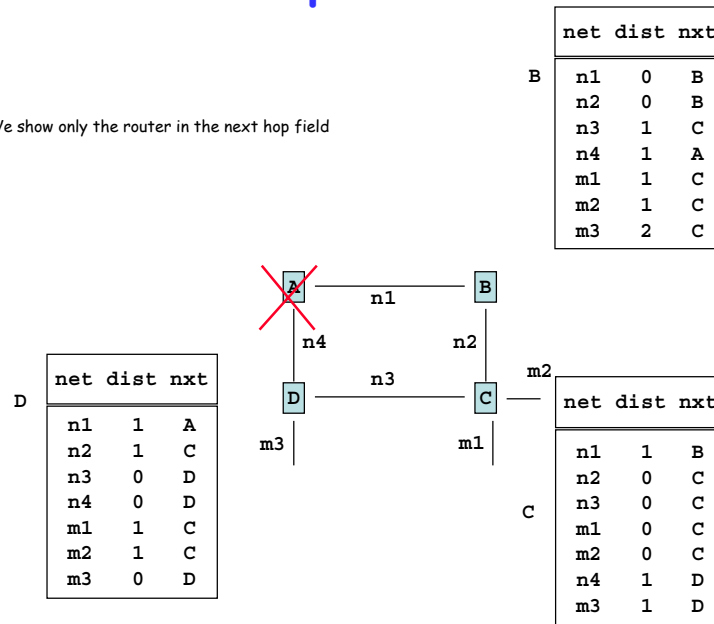
C

net	dist	nxt
n1	1	n2,B
n2	0	n2,C
n3	0	n3,C
m1	0	m1,C
m2	0	m2,C
n4	1	n3,D
m3	1	n3,D

22

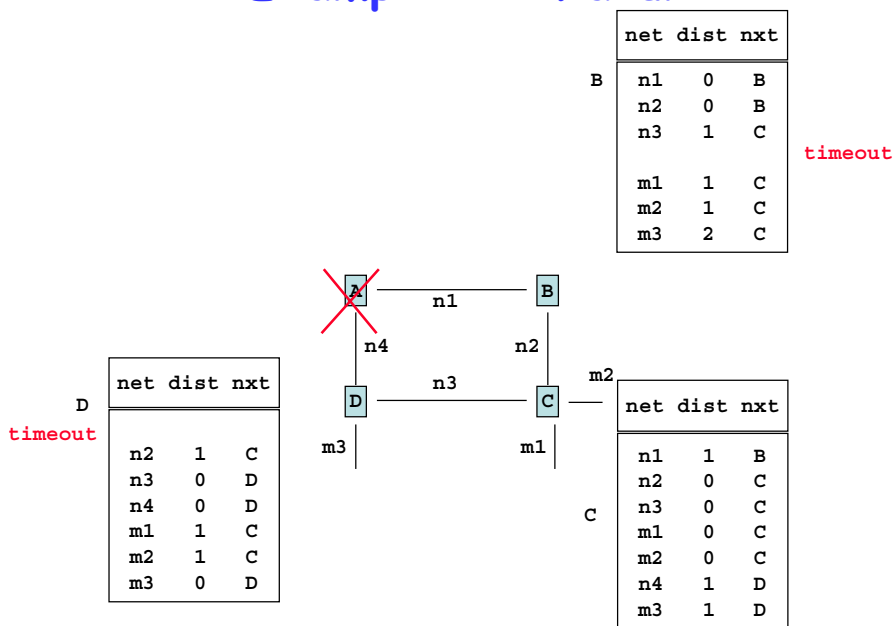
## Example 1 - Failure

We show only the router in the next hop field



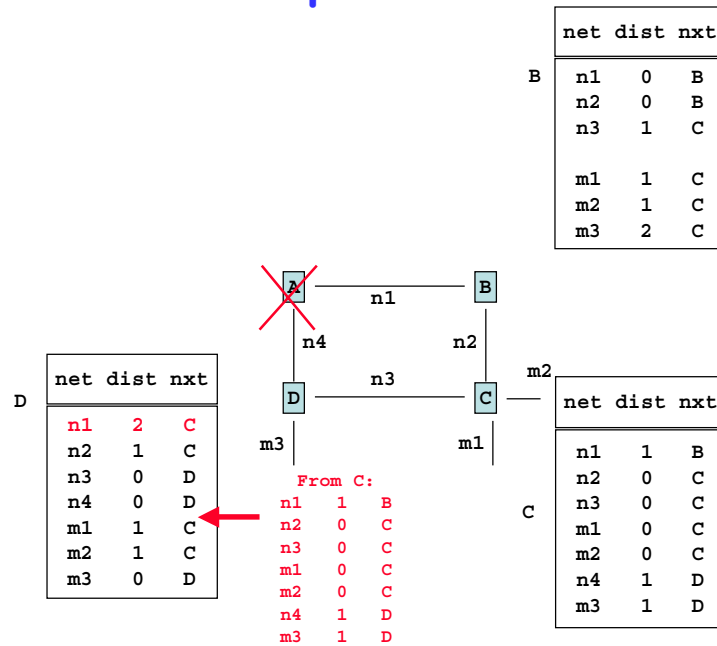
23

## Example 1 - Failure



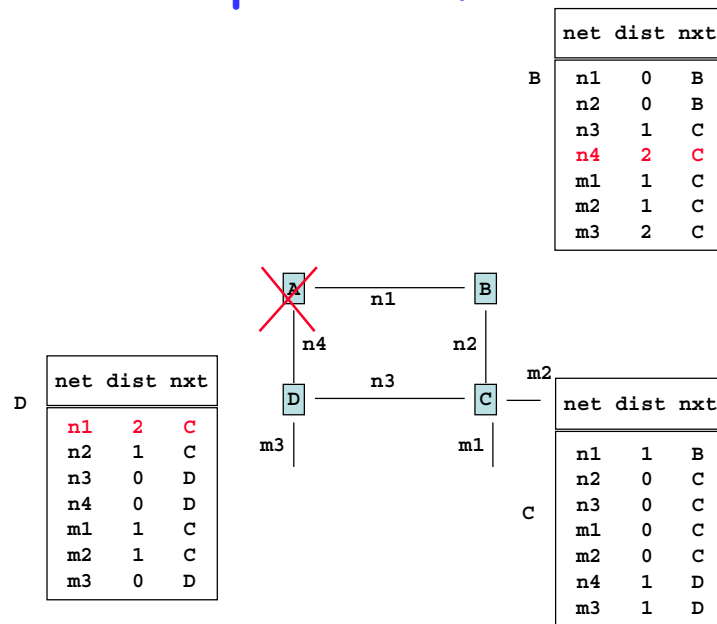
24

## Example 1 - Failure



25

## Example 1 - After Failure



26

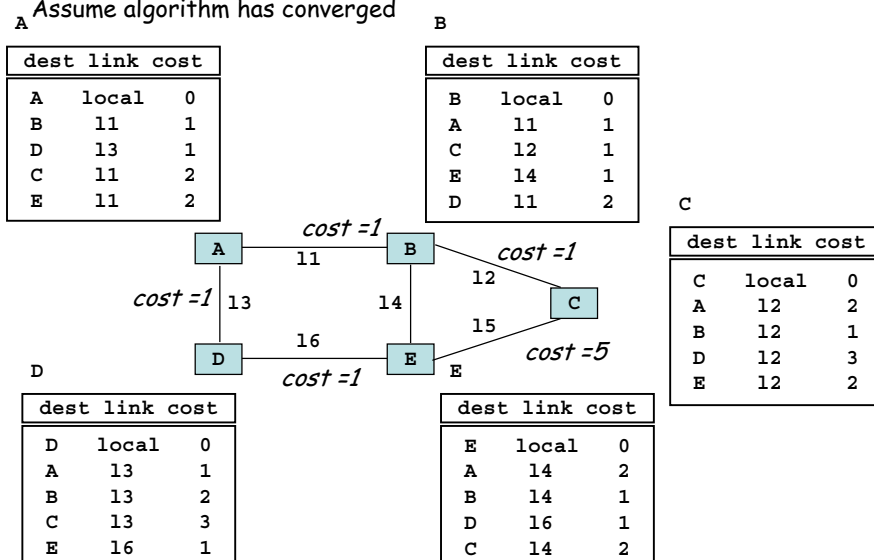
## Example 1: conclusions

- Example 1 illustrates
  - how Bellman Ford is mapped to the network concepts
  - how topology changes are taken into account
    - most recent announcement replaces previous ones
    - non refreshed announcements become obsolete
  - how distance vector carries reachability information

27

## Example 2

To simplify, we identify destination with router  
 Assume algorithm has converged



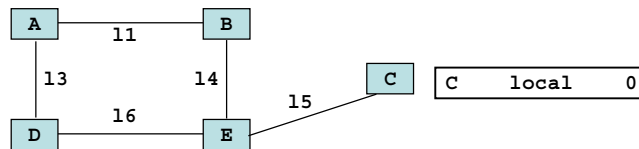
28

## Example 2

- ❑ we now show only table entries: to C
- ❑ link 2 fails
- ❑ B updates its table

C	11	2
---	----	---

C	12	$\infty$
---	----	----------



C	13	3
---	----	---

C	14	2
---	----	---

29

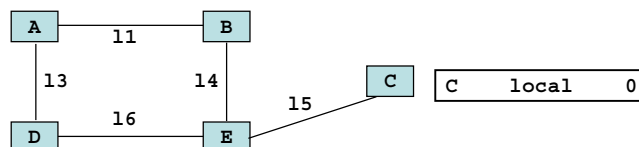
## Example 2: Link failure

- Just before B updates its table, A broadcasts its table with cost 2 to C
- B accepts

C	11	2
---	----	---

C	11	3
---	----	---

→  
from A: C | 1 2



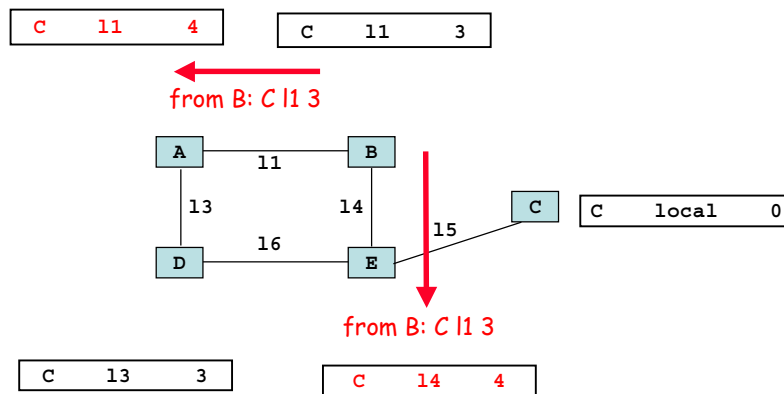
C	13	3
---	----	---

C	14	2
---	----	---

30

## Example 2: Link failure

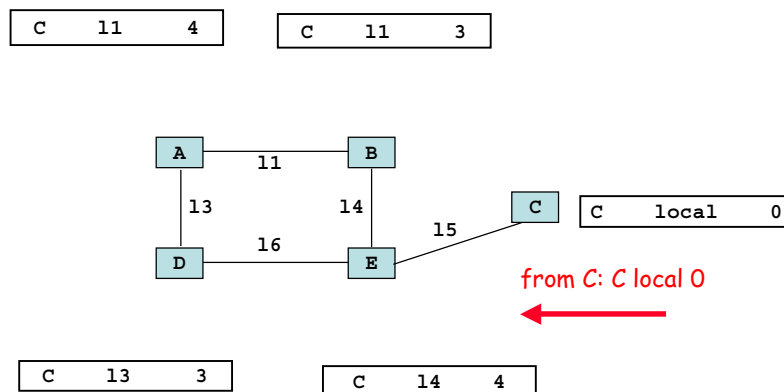
- B sends update to A and E
- A and E accept



31

## Example 2: Link failure

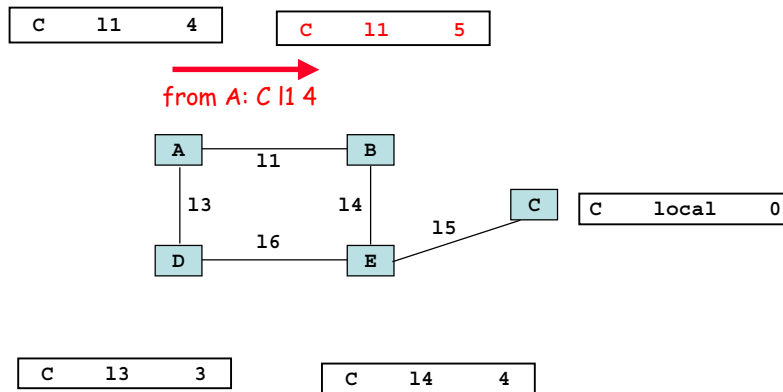
- C sends update
- it is ignored by E because it is less good



32

## Example 2: Link failure

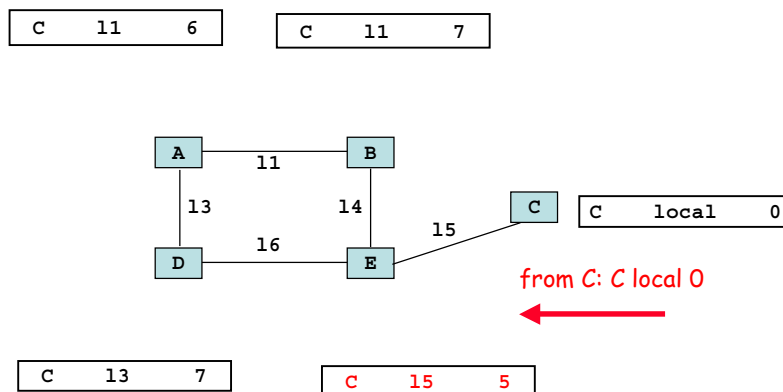
- A broadcasts its table with cost 4 to C
- B accepts ... we have a loop between A and C
- cost is increase by 2 at every iteration



33

## Example 2: Link failure

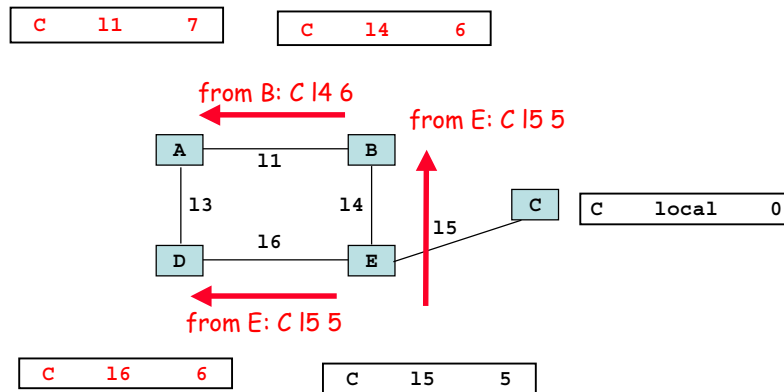
- E now accepts announcement from C



34

## Example 2: Link failure

- E sends announcements to D and B
- B and D send announcements to A
- the algorithm has converged - stable state



35

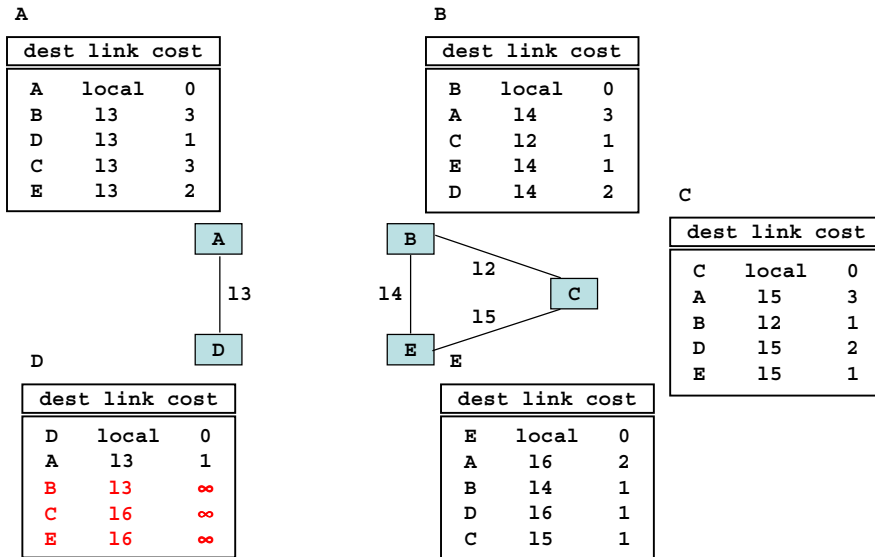
## Conclusions from Example 2

- ❑ the algorithm converges after modification of the topology, but the convergence may be very slow
  - bounce effect
- ❑ Q: during convergence time, how are routing tables ?

36

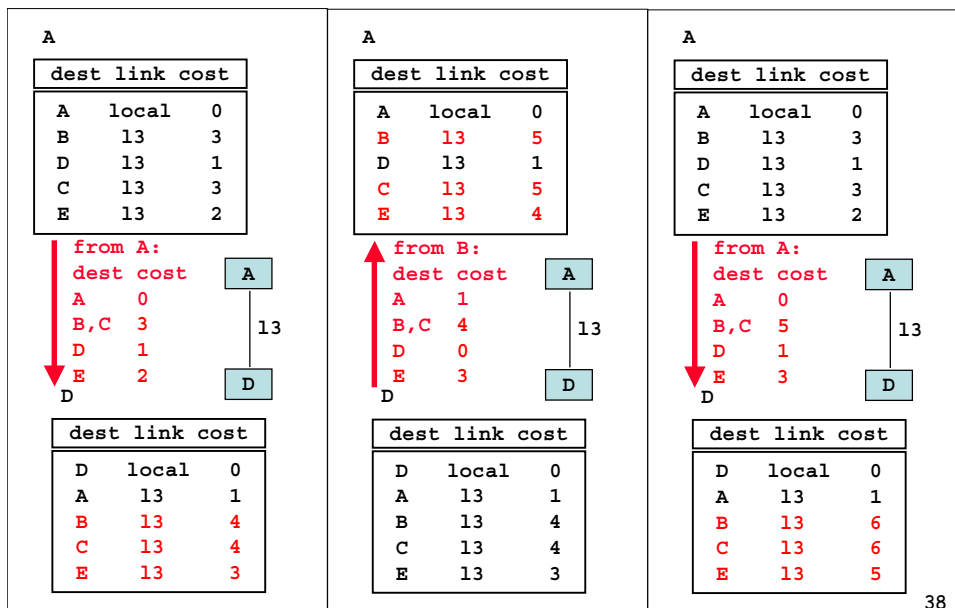
# Example 3

Assume now all link costs are equal to 1  
 Links l1 and l6 fail  
 D detects failure and sets costs to infity



37

# Example 3



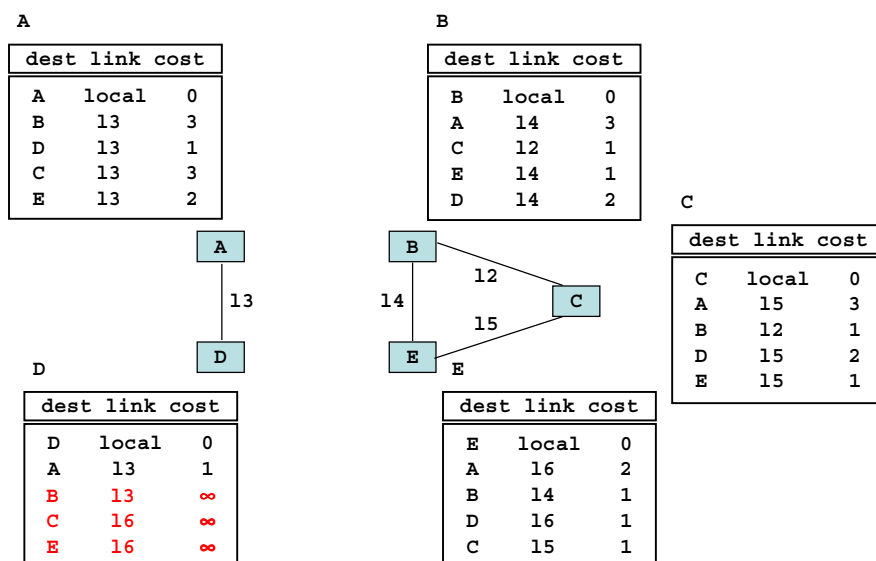
38

## Conclusion from Example 3

- ❑ The costs to C, B, E grow unbounded "Count to Infinity"
  - the true costs are infinite
- ❑ Convergence to a stable state if we set
  - $\infty$  = large number
  - e.g. RIP:  $\infty$  = 16
- ❑ "Split Horizon"
  - a heuristic to prevent this
  - if A routes packets to X via B, it does not announce this route to B

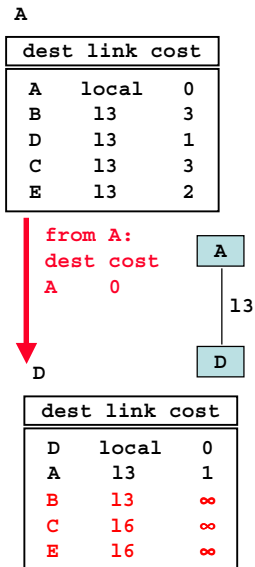
39

## Example 3: with Split Horizon



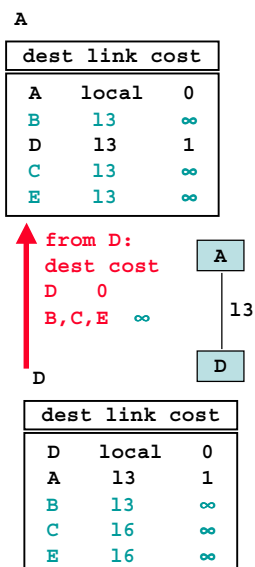
40

## Example 3: with Split Horizon



41

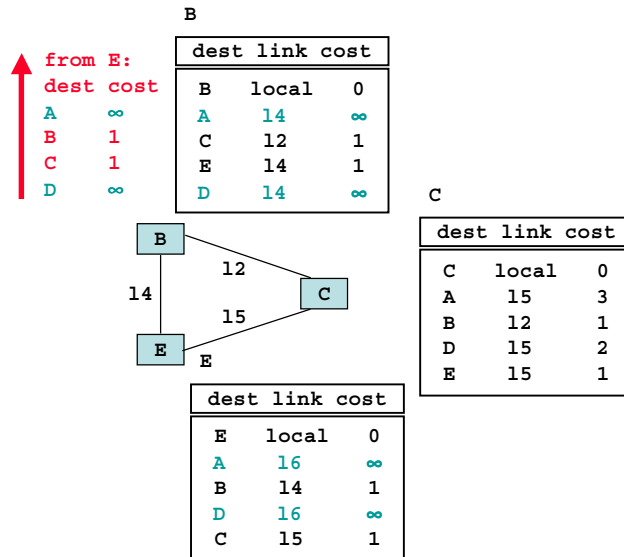
## Split horizon



- ❑ Split horizon cuts the process of counting to infinity

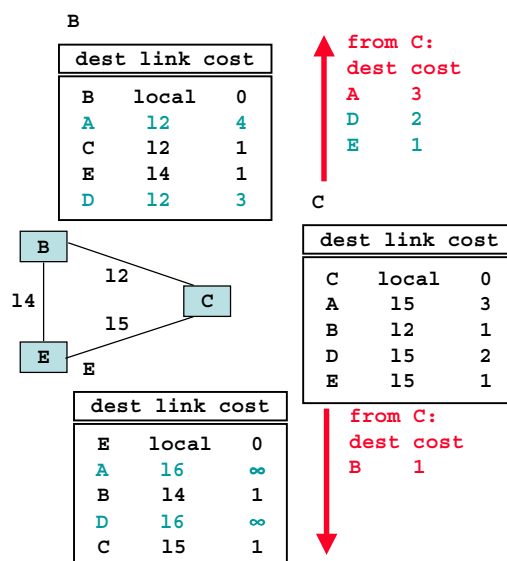
42

## Split horizon may fail



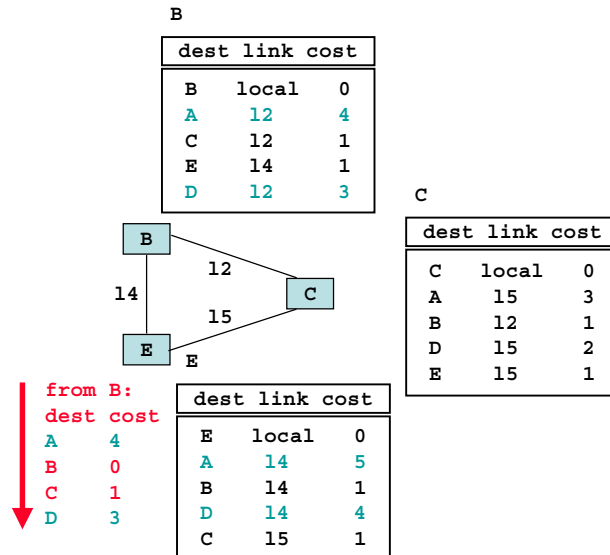
43

## Split horizon may fail



44

## Split horizon may fail



45

## Conclusion: Distance Vector

- ❑ convergence to stable state may be slow after changes
- ❑ count to infinity must be prevented by setting a maximum distance

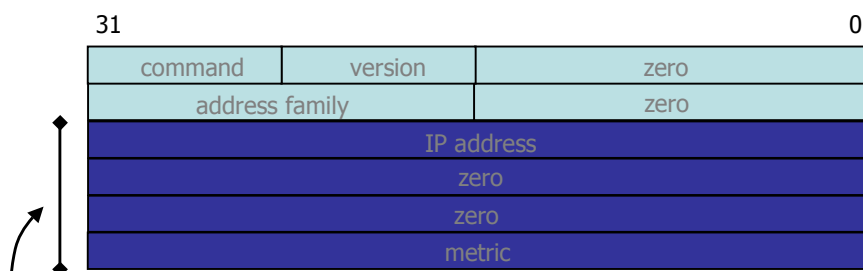
46

## Distance Vector Protocols RIP v1

- Distance vector protocol
- Metric - hops
- Network span limited to 15
  - $\infty = 16$
- 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  $\infty$

47

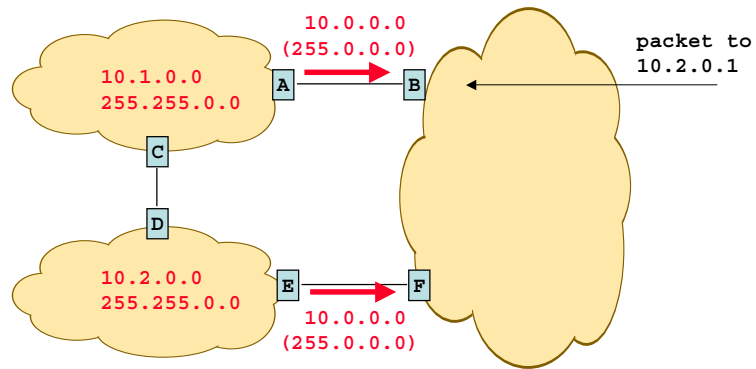
## Message format



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

48

## 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

49

## RIP v2

- Subnetworks
  - take into account CIDR prefixes and netmasks
- Authentication

50

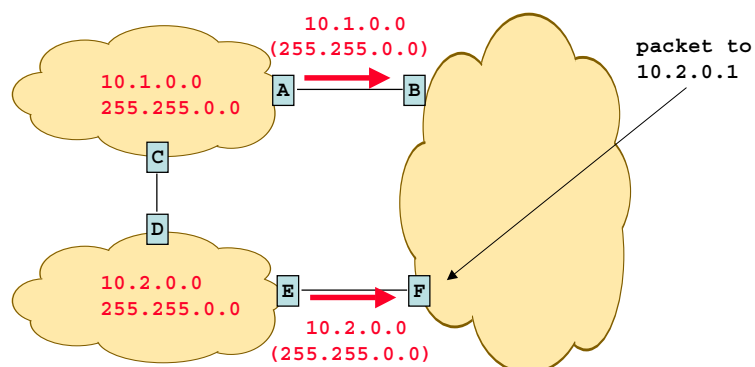
## Message format

31			0
command	version	routing domain	
address family		route tag	
IP address			
netmask			
next router			
metric			

- Command, version unchanged
- One address family - authentication
- Routing domain and next router
  - distinguish different addressing domains (e.g. AS)
  - used at the border of AS
- Route tag
  - for external routes (used by BGP)

51

## Announcing netmasks

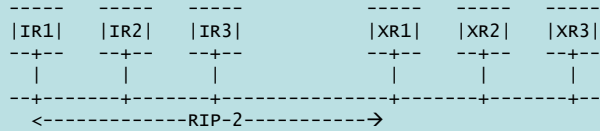


- E can forward to 10.2.0.0
- Packet to 10.2.0.1 can go through F

52

## Use of Next-Hop Field

This is a simple example of the use of the next hop field in a rip entry.

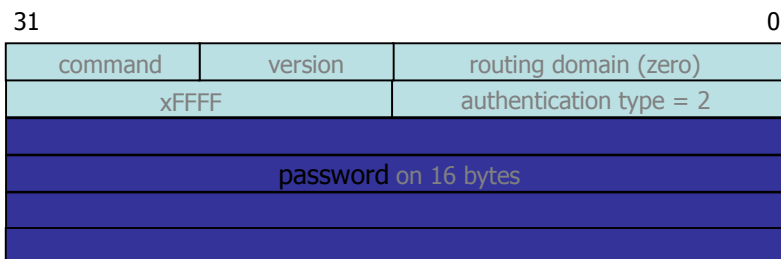


Assume that IR1, IR2, and IR3 are all "internal" routers which are under one administration (e.g. a campus) which has elected to use RIP-2 as its IGP. XR1, XR2, and XR3, on the other hand, are under separate administration (e.g. a regional network, of which the campus is a member) and are using some other routing protocol (e.g. OSPF). XR1, XR2, and XR3 exchange routing information among themselves such that they know that the best routes to networks N1 and N2 are via XR1, to N3, N4, and N5 are via XR2, and to N6 and N7 are via XR3. By setting the Next Hop field correctly (to XR2 for N3/N4/N5, to XR3 for N6/N7), only XR1 need exchange RIP-2 routes with IR1/IR2/IR3 for routing to occur without additional hops through XR1. Without the Next Hop (for example, if RIP-1 were used) it would be necessary for XR2 and XR3 to also participate in the RIP-2 protocol to eliminate extra hops.

RFC 2453

53

## Simple authentication

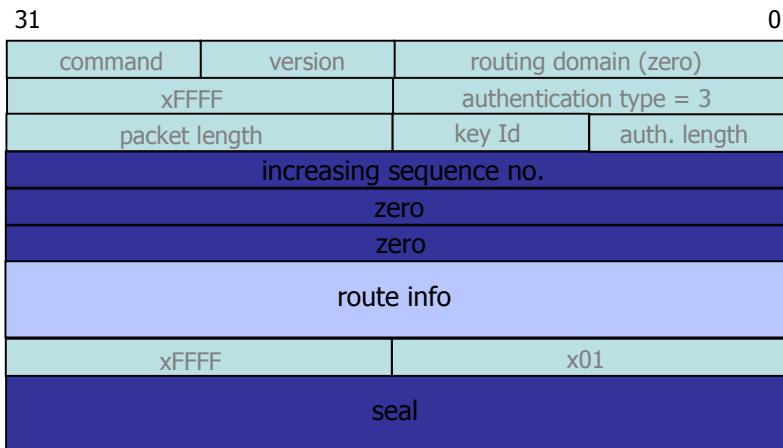


```

❑ Configuration of gated (/etc/gated.conf)
  rip yes {
      interface all
      version 2 multicast
      authentication simple "qptszwmz"
  }
  
```

54

## MD5 authentication



55

## 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"
}
```

56

## 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

57

## Metric example



- ❑ Metric
  - $Trans = 10000000 / \text{Bandwidth}$  (time to send 10 Kb)
  - $delay = (\text{sum of Delay}) / 10$
  - $m = [K_1 * Trans + (K_2 * Trans) / (256 - \text{load}) + K_3 * \text{delay}]$
  - default:  $K_1=1, K_2=0, K_3=1, K_4=0, K_5=0$
  - if  $K_5 \neq 0, m = m * [K_5 / (\text{Reliability} + K_4)]$
- ❑ Bandwidth in Kb/s, Delay in  $\mu s$ 
  - At Venus: Route for 172.17/16: Metric =  $10000000 / 784 + (20000 + 1000) / 10 = 14855$
  - At Saturn: Route for 12./8: Metric =  $10000000 / 224 + (20000 + 1000) / 10 = 46742$

58

## 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

59

## Review Questions

Explain the following terms:

- distance vector
- bounce effect
- count to infinity
- split horizon
- Bellman Ford
- RIP, IGMP
- source routing
- Explain why shortest path routing is not necessarily a globally optimum
- What is the Braess paradox ?

60

## C. Link State Routing

- ❑ **Principle** of link state routing
  - each router keeps a topology database of whole network
  - link state updates flooded, or multicast to all network
  - routers compute their routing tables based on topology  
often uses Dijkstra's shortest path algorithm
- ❑ Used in OSPF (Open Shortest Path First) and PNNI (ATM routing protocol)

61

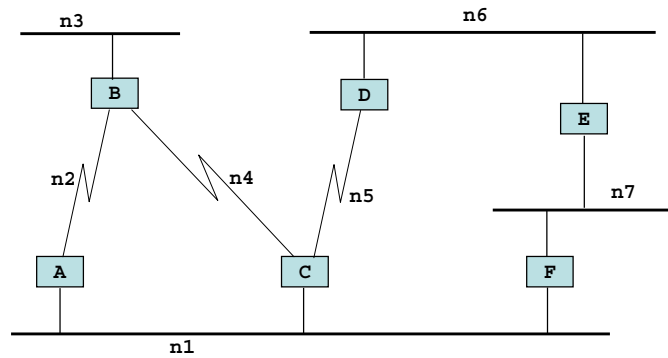
### (a) Topology Database Synchronization

- ❑ Neighbouring nodes synchronize before starting any relationship
  - Hello protocol; keep alive
  - initial synchronization of database
  - description of all links (no information yet)
- ❑ Once synchronized, a node accepts link state advertisements
  - contain a sequence number, stored with record in the database
  - only messages with new sequence number are accepted
  - accepted messages are flooded to all neighbours
  - sequence number prevents anomalies (loops or blackholes)

62

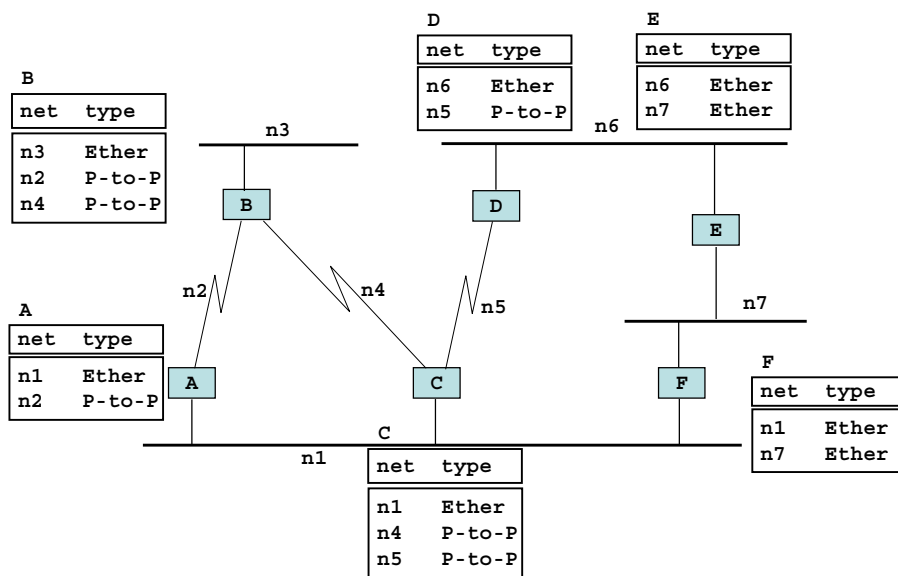
## Example network

- Each router knows directly connected networks



63

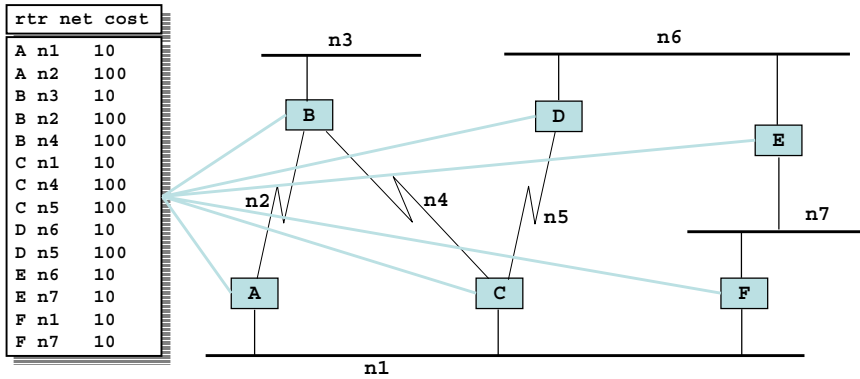
## Initial routing tables



64

## After Flooding

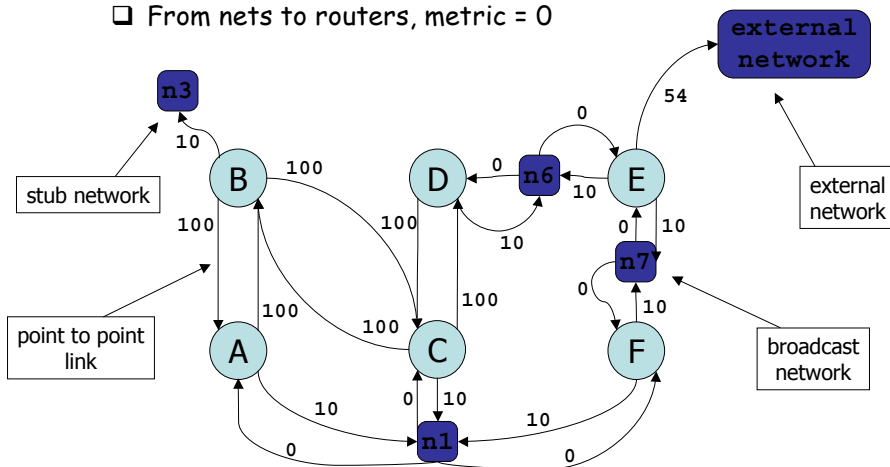
- ❑ The local metric information is flooded to all routers
- ❑ After convergence, all routers have the same information



65

## (b) Topology graph

- ❑ Arrows routers-to-nets with a given metric
  - except P-to-P, stub, and external networks
- ❑ From nets to routers, metric = 0



66

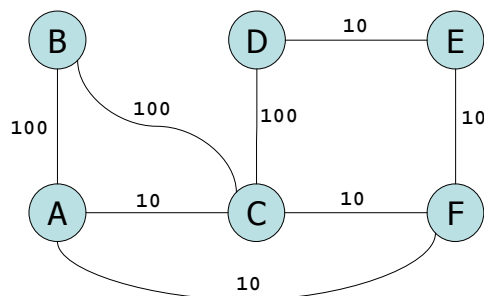
## (b) Path Computation

- ❑ Performed locally, based on topology database
- ❑ Computes one or several best paths to every destination from this node
- ❑ Best Path = shortest for OSPF
- ❑ OSPF uses Dijkstra's shortest path
  - the best known algorithm for centralized operation
- ❑ Paths are computed independently at every node
  - synchronization of databases guarantees absence of persistent loops
  - every node computes a shortest path tree *rooted at self*

67

## Simplified graph

- ❑ Only arrows with metrics between routers
- ❑ Every node executes the shortest path computation on the graph - same graph, but different sources



68

## Dijkstra's Shortest Path Algorithm

- The nodes are  $0 \dots N$  and the algorithm computes best paths from node  $0$
- $c(i,j)$  is the cost of  $(i,j)$ ,
- $\text{pred}(i)$  is the predecessor of node  $i$  on the tree  $M$  being built
- $m(j)$  is the distance from node  $0$  to node  $j$ .

```

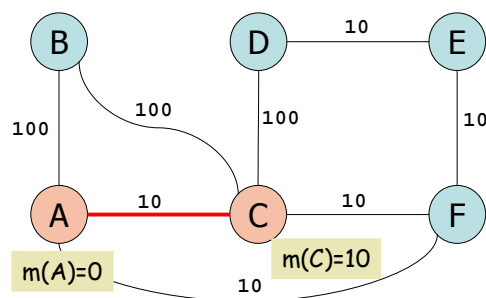
m(0) = 0; M = {0};
for k=1 to N {
  find (i0, j0) that minimizes m(i) + c(i,j),
                    with i in M, j not in M
  m(j0) = m(i0) + c(i0, j0)
  pred(j0) = i0
  M = M ∪ {j0}
}

```

□ like Be

69

## Example: Dijkstra at A

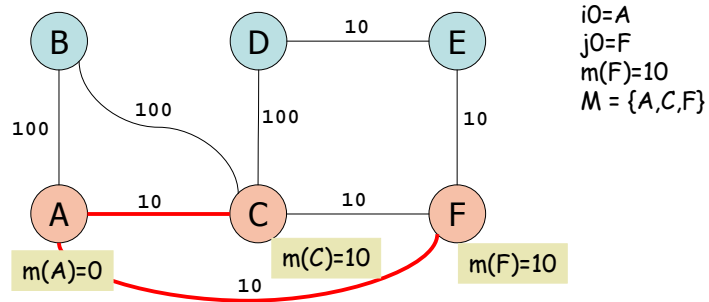


init:  $M = \{ A \}$

step 1:  
 $i0=A$   
 $j0=C$   
 $m(C)=10$   
 $M = \{ A, C \}$

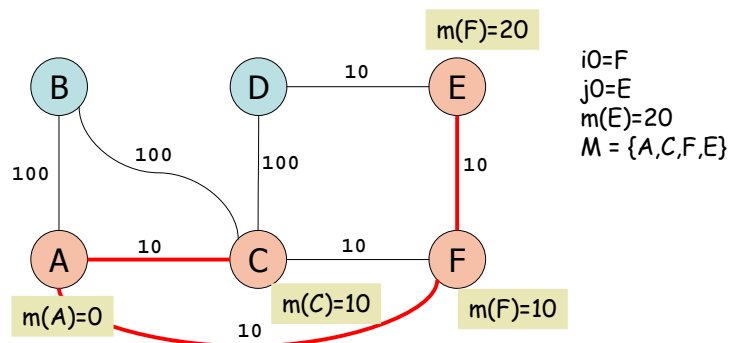
70

## Example: Dijkstra at A



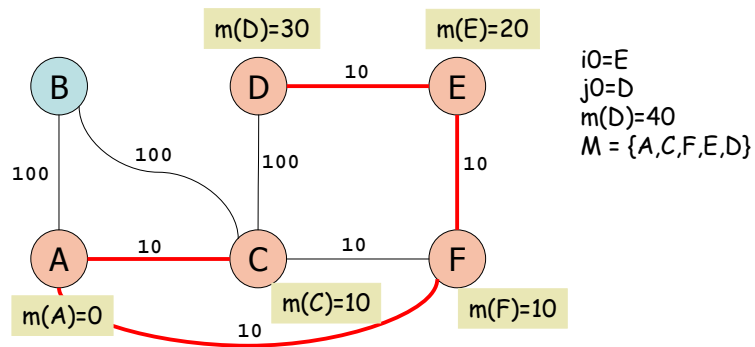
71

## Example: Dijkstra at A



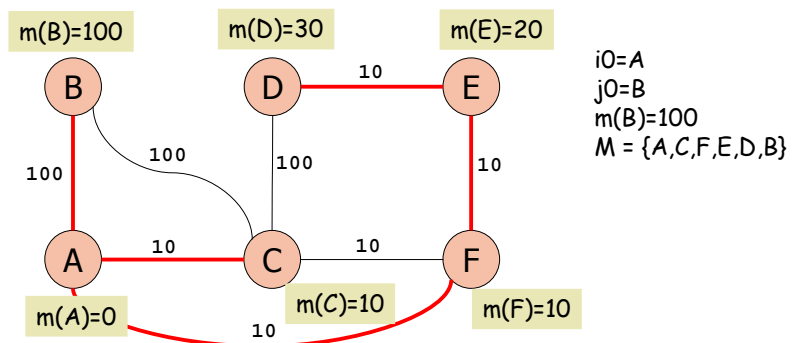
72

## Example: Dijkstra at A



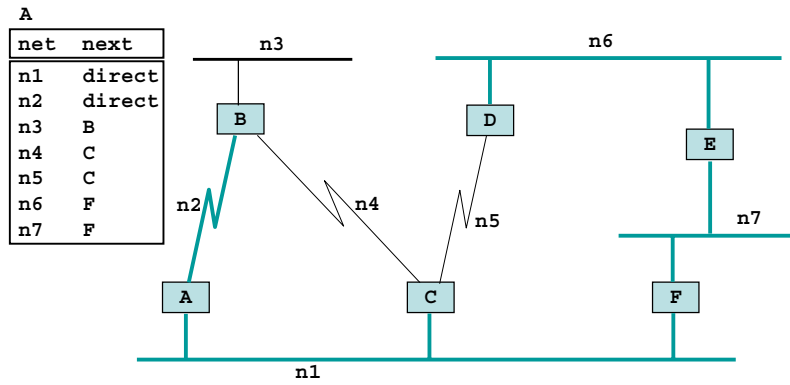
73

## Example: Dijkstra at A



74

## Routing table of A



75

## Test Your Understanding

- Q1: Run Dijkstra at C
- Q2: What are the routing tables at C

76

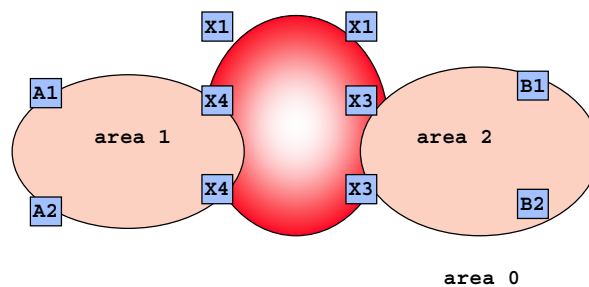
## Divide large networks

- ❑ Why divide large networks?
- ❑ Cost of computing routing tables
  - update when topology changes
  - SPF algorithm
    - $n$  routers,  $k$  links
    - complexity  $O(n*k)$
  - size of DB, update messages grows with the network size
- ❑ Limit the scope of updates and computational overhead
  - divide the network into several areas
  - independent route computing in each area
  - inject aggregated information on routes into other areas

77

## Hierarchical Routing

- ❑ A large OSPF domain can be configured into *areas*
  - one *backbone area* (area 0)
  - non backbone areas (areas numbered other than 0)
- ❑ All inter-area traffic goes through area 0
  - strict hierarchy
- ❑ Inside one area: link state routing as seen earlier
  - one topology database per area



78

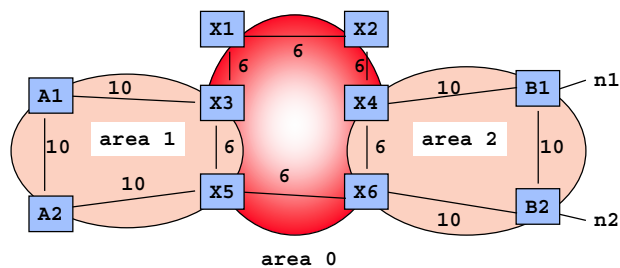
## Principles

- ❑ Routing method used in the higher level:
  - *distance vector*
  - no problem with loops - one backbone area
- ❑ Mapping of higher level nodes to lower level nodes
  - area border routers (inter-area routers) belong to both areas
- ❑ Inter-level routing information
  - summary link state advertisements (LSA) from other areas are injected into the local topology databases

79

## Example

- ❑ Assume networks *n1* and *n2* become visible at time 0. Show the topology databases at all routers



80

## Hints

- All routers in area 2 propagate the existence of n1 and n2, directly attached to B1 (resp. B2). Draw the topology database in area 2.
- Area border routers X4 and X6 belong to area 2, thus they can compute their distances to n1 and n2
- Area border routers X4 and X6 inject their distances to n1 and n2 into the area 0 topology database (item 3 of the principle). The corresponding summary link state record is propagated to all routers of area 0. Draw now the topology database in area 0.
- All routers in area 0 can now compute their distance to n1 and n2, using their distances to X4 and X6, and using the principle of distance vector (item 1 of the principle). Do the computation for X3 and X5.
- Area border routers X3 and X5 inject their distances to n1 and n2 into the area 1 topology database (item 3 of the principle). Draw now the topology database in area 1.

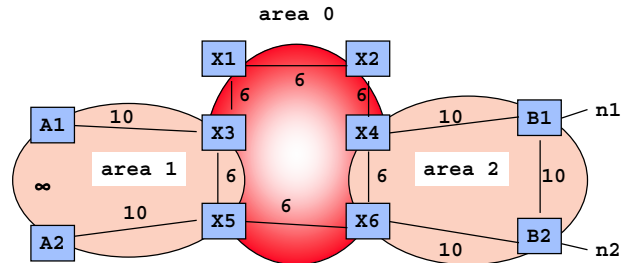
81

## Comments

- ❑ Distance vector computation causes none of the RIP problems
  - strict hierarchy: no loop between areas
- ❑ External and summary LSA for all reachable networks are present in all topology databases of all areas
  - most LSAs are external
  - can be avoided in configuring some areas as terminal: use default entry to the backbone
- ❑ Area partitions require specific support
  - partition of non-backbone area is handled by having the area 0 topology database keep a map of all area connected components
  - partition of backbone cannot be repaired; it must be avoided; can be handled by backup virtual area 0 links through non backbone area

82

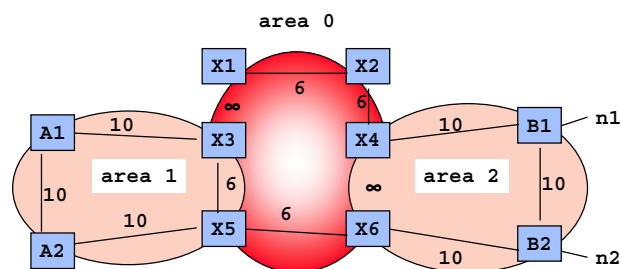
## Problems - link failure



- ❑ Link A1-A2 fails, link X3-X6 is not in area 1. Area 1 is partitioned
  - X3 has a route to A1, X5 to A2
  - cannot pass to X5 a packet to A1 and to X3 a packet to A2
- ❑ Solution
  - X3 and X5 will advertise only distances to connected networks - as though there were two separated areas

83

## Problems - partitioned backbone

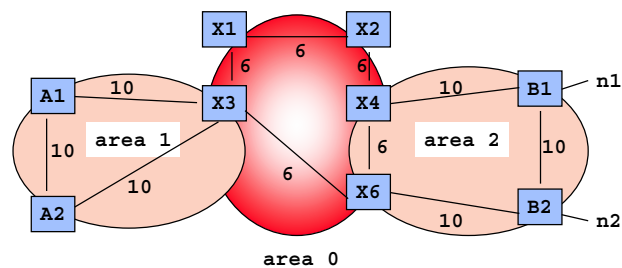


- ❑ No connectivity between areas via backbone
- ❑ There is a route through Area 2
- ❑ Virtual link
  - X4 and X6 configure a virtual link through Area 2
  - virtual link entered into the database, metric = sum of links

84

## Stub area

- ❑ Many networks are connected only via one router
- ❑ Stub area
  - all external networks aggregated into default route
  - this reduces routing table sizes
  - e.g. route to n1, n2 or any other network in Area 0 and 2 goes through X3



85

## The OSPF Protocol

- ❑ OSPF (Open Shortest Path First)
  - IETF standard for internal routing
  - used in large networks (ISPs)
- ❑ Link State protocol + Hierarchical
- ❑ The network is represented using the following principles
  - separate hosts and routers
  - consider different types of networks
    - broadcast (Ethernet), NBMA (ATM, X.25), point-to-point (PPP)
  - divide large networks into several areas
  - independent route computing in each area

86

## LS (comments)

- ❑ Multiple paths are possible
  - modification of Dijkstra's algorithm to keep lists of paths instead of just a spanning tree
  - even non-shortest paths are possible. For routing to be loop free, at node  $i$ , for a packet sent to  $n$ , we require that the next hop  $k$  satisfies  $d(k,n) < d(i,n)$
- ❑ LS can use other algorithms than Dijkstra's, but complexity of D. is generally less
  - Dijkstra:  $O(N \ln M)$ ,  $N$ =number of nodes,  $M$ =number of links
  - BF:  $O(MN)$

87

## LS: Summary

- ❑ All nodes compute their own topology database
  - represents the whole network
  - strongly synchronized
- ❑ All nodes compute their best path tree to all destinations
- ❑ Routing tables are built from the tree
  - used for next hop routing only
- ❑ LS versus DV
  - LS avoids convergence problems of DV
  - supports flexible cost definitions; can be used for routing ATM connections
  - LS is much more complex

88

## D. Multicast Routing

- ❑ Multicast packet forwarding
  - given source address and multicast destination address, send packet to all *relevant* interfaces
  - *relevant* is indicated by the multicast routing method
- ❑ End-system group membership (as receiver) is known to routers via IGMP
- ❑ LS method: Multicast-OSPF (M-OSPF)
  - topology database contains info about group membership
  - OSPF computes a tree for one source to all destinations (standard LS routing)
  - parts of tree leading to no destination are removed
  - routing table built accordingly
  - works only in one area using one LS routing algorithm
- ❑ Other methods exist for global multicast routing
  - RPF (Reverse Path Forwarding) with pruning
  - core based tree

89

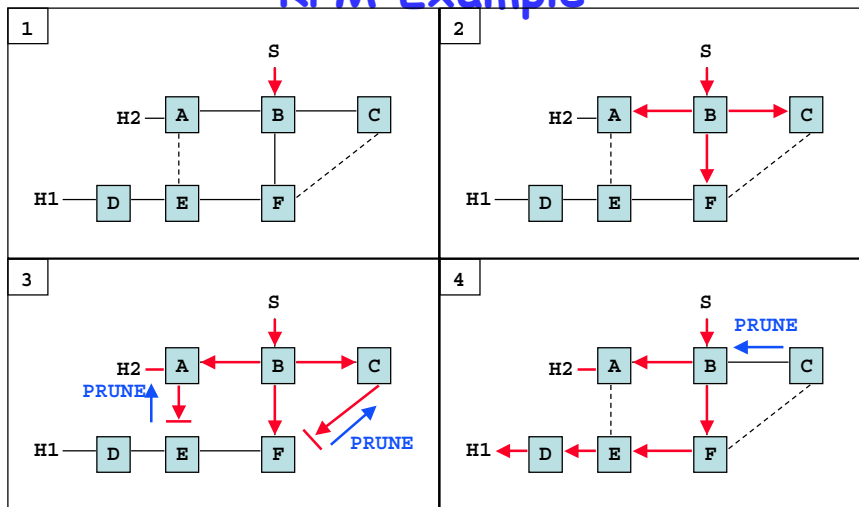
## Reverse Path Multicasting (RPM)

- ❑ RPM = Reverse Path with pruning
  - compute one tree per source  $S$  using reverse of direct path tree to  $S$
  - then suppress (*prune*) branches that are not needed
- ❑ Router  $R$  receives packet  $\text{src addr} = S$ ,  $\text{dest addr} = M$  on interface  $i$ 

```
if  $i$  is not the next hop towards  $S$  {
    discard packet; send PRUNE ( $S$ ;  $M$ ) to  $i$ ;
}
else {
    if there is a relevant interface send packet to all relevant
    interfaces
    else send PRUNE ( $S$ ;  $M$ ) to  $i$ 
}
relevant interface: leaf: at least one member exists
member:             host with IGMP
                   routers that did not send PRUNE ( $S$ ;  $M$ )
```
- ❑ Implemented with DVMRP (Mbone) or PIM-dense (global Internet)
  - DVMRP uses its own DV algo for path computation
  - PIM-dense uses existing point-to-point routing

90

## RPM Example



Showing the Shortest Path Tree for traffic to S, as given by local tables (links not on tree : - - - )  
H1 and H2 have subscribed to the multicast address

91

- The figure shows the Shortest Path Tree for traffic to S, as given by local tables (links not on the shortest path tree to S are dashed).
- H1 and H2 have subscribed to the multicast address M
- (1) S sends IP packets to M.
- (2) The first packet is forwarded by B to all its ports
- (3) E and F do not accept packets arriving via ports not on the spanning tree. The corresponding links are removed from the multicast distribution tree
- F does not send to C if it can compute shortest paths one step ahead: F knows that C will not accept the packet from F because, for C, F is not on the path to S
- (4) C suppresses its link to B because it has no host listening and the only connected routers has sent a PRUNE message for (S, M). All other packets follow the marked line. The effect of PRUNE messages disappears after a timeout ( for example 1 mn), which causes steps (2) and (3) to be repeated at regular intervals.

92

## PIM Sparse Mode

- ❑ PIM sparse mode
  - designate one router as rendez-vous point (RP)
  - **receivers** send membership messages to RP, build tree along the path
  - **senders** send to group by encapsulating packets to RP; RP sends on tree
  - one tree per group
  - but a destination can decide to build an additional tree towards a given source
  - the location of RP is critical
  - protocol exists for repairing RP failures

93

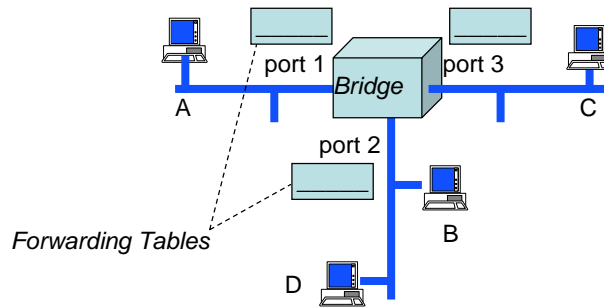
## E: Transparent Bridging (TB)

- ❑ Interconnect systems beyond one LAN segment, keeping main characteristics of LAN
  - without additional addresses
    - MAC addresses used to identify end systems
  - preserve sequence integrity
- ❑ End systems ignore that there are transparent bridges
  - bridge is transparent
  - MAC frames not changed by bridges
  - frames not sent *to* bridge, but rather: bridge is promiscuous
    - (listens to all frames)
- ❑ TB operation
  - connectionless forwarding, unstructured addresses
  - bridges are plug and play: no address configuration (no IP address needed)

94

## TB: Learning Bridge

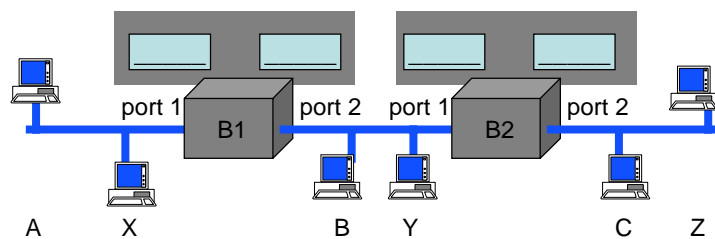
- bridge builds routing table by reading all traffic
  - table built by **learning** from SA field in MAC frame
  - learnt addresses timeout if not re-learnt
- broadcast forwarding if DA unknown



95

## Extension to Several Learning Bridges

- Can the learning bridge be extended to a network of bridges ?
- How does B2 see the network ?



96

## Loop-Free topology

- ❑ Learning bridge works well on Loop-Free topology only
  - Bidirectional graph: node = bridge, edge = connection through LAN
  - Loop free - bidirectional graph = bidirectional tree
    - examples: line, star
  - On a tree, there is only one path from A to B
  - **Proposition:** If bridge topology is loop-free, then there exists only one path from any end system to any bridge
    - Loop-free topology is required and sufficient for

97

## Transparent Bridging

- ❑ Based on learning bridge:
  - table driven forwarding, flooding if unknown DA or multicast, learning
- ❑ Forces topology to a tree
  - Spanning Tree algorithm run by all bridges
  - Some links blocked to prevent loops
    - ports that are allowed to forward frames (in either way) are said to be "in the forwarding state" or called "forwarding ports"

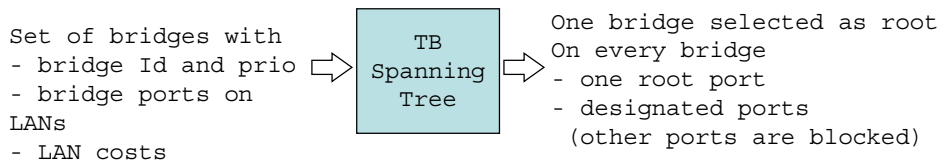
98

## TB Forwarding Method

Individual PDU forwarding	<pre> Copy all frames on all forwarding ports  Frame received on port i -&gt;                                 /* port i is forwarding */ <b>If</b> DA is unicast, is in forwarding table with     port j and j is a forwarding port     <b>then</b> copy to port j     <b>else</b> flood all forwarding ports ° i Update forwarding table with (i, SA)         </pre>
Control method	<pre> Maintain spanning tree and port states  Learn addresses on reading traffic         </pre>

99

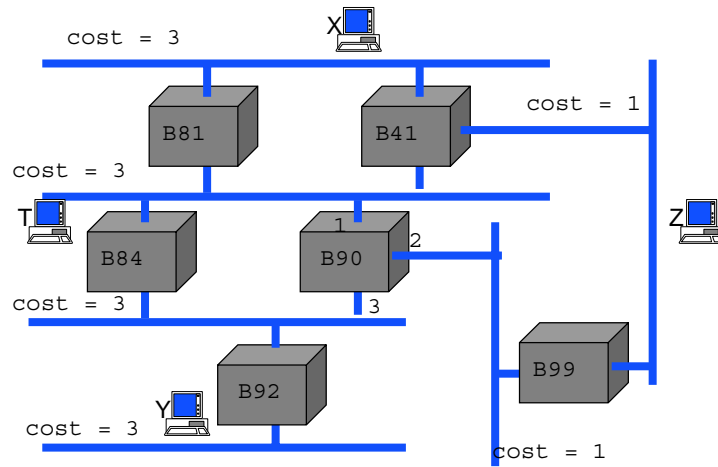
## TB Spanning Tree Specification



- Bridges viewed as a bidirectional graph (nodes = bridges)
- Selection of the root bridge
  - lowest priority with lowest identifier
- Spanning Tree = shortest path tree from root to all bridges
  - edge costs set by management, high cost = less traffic
  - based on distributed Bellman Ford
- Root port on one bridge = port towards root, shortest path
  - in case of equal costs, lowest id chosen
- Designated bridge ports: on one LAN, shortest path to root
- Ports other than root or designated are blocked

100

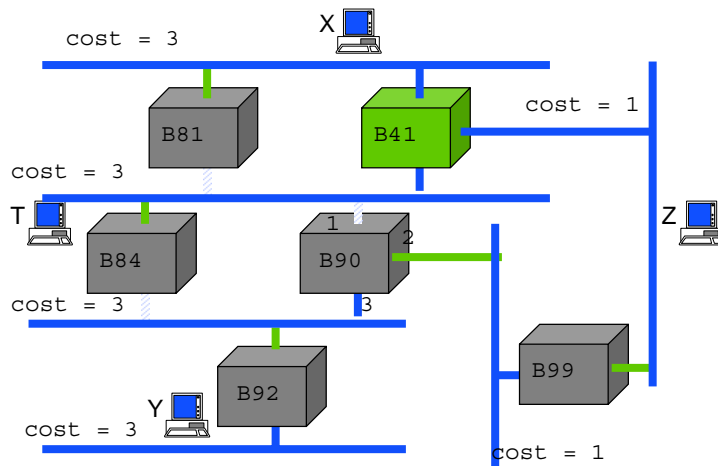
## Example



- ❑ every bridge has one root port + some designated ports; other ports are blocked for individual frame forwarding
- ❑ all ports are active for control flows (BPDUs)

101

## Solution



Forwarding Tables:

B41 1X 2YZ 3T	B81 1XYZT
B84 1XYZT	B90 2XZT 3Y
B92 1XZT 2Y	B99 dXZT gY

102

## Spanning Tree Algorithm

- ❑ Distributed in all bridges
- ❑ Bridges exchange messages with neighbours in order to both
  - elect a root
  - determine shortest path tree to root
    - root port = port towards root on shortest path tree
    - designated port = port for which bridge was designated

103

## Bridge PDUs

- ❑ Control method uses control frames called Bridge PDUs (BPDUs)
  - MAC DA = all bridges (multicast) 01 80 C2 00 00 00
  - SAP = "01000010"
- ❑ BPDUs are not forwarded by bridges
  - unlike all other frames
  - BPDUs are sent by one bridge to all bridges on the same LAN segment
  - reminder: a data frame is never sent to bridge by end system
- ❑ Configuration BPDU contain
  - root Id with priority
  - cost to root (from sender of config BPDU)
  - Id of sender with port number

104

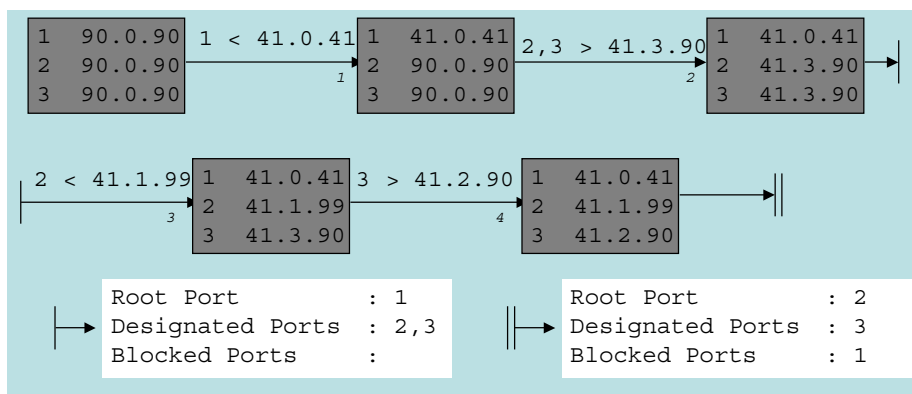
## Initialization of Spanning Tree (1)

- ❑ Bridge initially assumes self is root
- ❑ Bridge computes own new config BPDU based on received information
  - determine best root so far
  - distance to root with Bellman Ford
- ❑ On every port, Bridge transmits config BPDU until it receives a better config BPDU on that port
  - better - closer to root
- ❑ On every port, bridge maintains copy of best config BPDU sent or received

105

## Initialization of Spanning Tree (2)

- ❑ **Example:** Bridge B90 prepares config BPDU 90.0.0.90 and sends on all ports; B90 configuration tables:



message received on port 1: 1 < 41.0.41      message format: root.cost\_to\_root.sender

106

## Basic ST Procedure

```
config BPDU received on any port or port enabled ->

compute new root;
compute new cost to root; /* Bellman Ford */
build new_config_BPDU;
for all ports i do
    if new_config_BPDU better than stored_config[i]
        then store and send on port i;
    end
end

compute root port /* smaller distance to root */
designated ports = ports where config BPDU was sent
blocked ports = other ports
```

```
r.c.s better than r'.c'.s' iff
    (r<r') or (r=r' and c<c') or (r=r' and c=c' and s<s')
```

107

## Topology changes

- New configurations
  - new BPDUs trigger basic procedure
- Failures, partitions: centralized control + distributed monitoring
  - assume bridge B99 fail: is ST recomputed ?

108

## ST: Support for Reconfiguration

- ❑ configuration monitoring triggered by root
  - root refreshes validity of stored configuration
  - timeout causes recomputation of spanning tree

```

Root sends config BPDUs every hello_timer;

Bridge B receives config BPDU on root port i ->
  Reset timer on stored_config[i]
  for all designated ports j
    B sends own config BPDU
    B resets timer on stored_config[j]

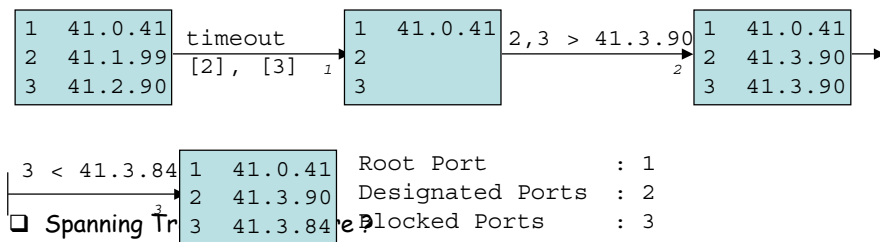
Bridge B.timeout on stored_config[j]->
  delete stored_config[j];
  B performs basic ST procedure;

```

109

## Example

- ❑ B99 powered off; stored\_config at B90 :



Blocked Port configs do not timeout: port 1 becomes root on this bridge. If it would be wrong, then it would timeout

110

## Transient States

- ❑ Transient periods cause loops or loss of connectivity
  - during reconfiguration, topology is not yet (in general) loop free
- ❑ Even transient loops should be avoided
- ❑ TB standard: forwarding state is not immediately operational
  - pre-forwarding states:
    - listening: wait for stabilization of ST (forwarding timer, 15 sec)
    - learning: wait for addresses to be learnt (forwarding timer, 15 sec)

State	Actions		
	Forward	ST	Learn
Blocking			
Listening			
Learning			
Forwarding			

112

## Port FSM

event \ state	Disabled	Blocking	Listening	Learning	Forwarding
Port enabled by NM BPDU	Blocking				
Port disabled (NM, failure)		Disabled	Disabled	Disabled	Disabled
Port Selected as Root or designated		Listening			
Port no longer root or designated			Blocking	Blocking	Blocking
Forwarding timer expires			Learning	Forwarding	

113

## Station Cache Timers

- ❑ Station Cache Timer
  - as long as possible to avoid broadcasts along the spanning tree
  - but wrong cache values cause station to be *unreachable* until timer expires
    - example 1: station moved
    - example 2: show that after failure of B99, U cannot reach Z until some entry in B90 is purged

114

## Long and Short Timers

- ❑ Two timer values are used
  - long timer (5mn): normal case
  - short timer = forwarding timer (15 sec): after spanning tree updates
- ❑ Timer switching mechanism
  - bridge B detects change in ST -> `maxLife = shortTimer`
  - how can bridges detect changes in ST ?

115

## Topology Update Mechanism

- ❑ Topology Update Mechanism :
  - when one bridge port changes out of or into blocking state then bridge sends topology update BPDU towards root (upstream bridges repeat BPDU up to root)
  - root forwards new config BPDU with "topology change flag" set for a time duration = forwarding timer + MaxAge timer
    - causes all bridges to use short timer value for caches
    - until BPDU from root received with "topology change" flag cleared

116

## F.1. Routing in the Internet

- ❑ The Internet is too large to be run by one routing protocol
- ❑ Hierarchical routing is used
  - the Internet is split into Domains, or Autonomous Systems
  - with OSPF: large domains are split into Areas
- ❑ Routing protocols are said
  - **interior**: (Internal Gateway Protocols, IGP): inside ASs: RIP, OSPF (standard), IGRP (Cisco)
  - **exterior**: between ASs: EGP (old) and BGP-1 to BGP-4 (today), IDRP (tomorrow)

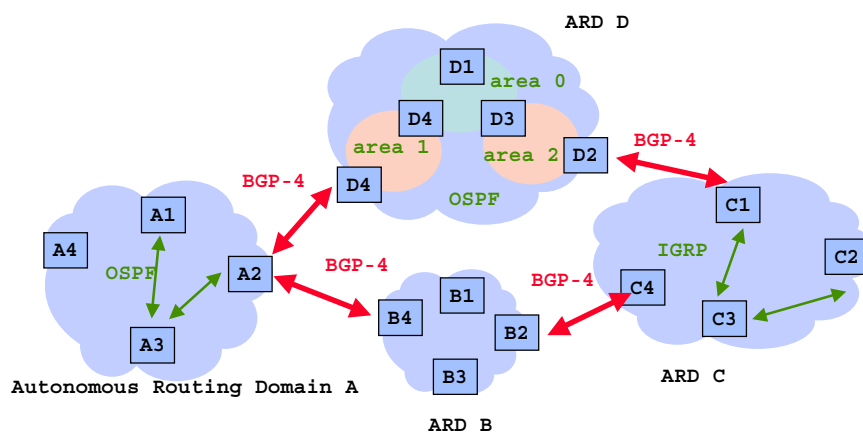
117

## Autonomous Routing Domains Autonomous Systems (ASs)

- ❑ ARD = routing domain under one single administration
  - one or more border routers
  - all subnetworks inside an ARD should be connected
  - should learn about other subnetworks - the routing tables of internal routers should contain entries of all destination of the Internet
- ❑ AS are ARD with a number ("AS number")
  - 16 bits
  - public: 1 - 64511
  - private: 64512 - 65535
- ❑ ARDs with default route to the rest of the world do not need a number
- ❑ Examples
  - AS1942 - CIGG-GRENOBLE, AS1717, AS2200 - Renater
  - AS559 - SWITCH Teleinformatics Services
  - AS5511 - OPENTRANSIT
  - EPFL: one ARD, no number

118

- the figure shows three domains, or ARDs.
- ARDs can be transit (B and D), stub (A) or multihomed (C). Only non stub domains need an AS number, as we can see on the BGP slides later on.



119

## Hierarchical Routing

- ❑ Hierarchical routing is different case by case, however, we can distinguish three elements
  - 1. **routing method** used in the higher level
  - 2. **mapping** higher level nodes to lower level nodes
  - 3. **inter-level** routing information
- ❑ We know two examples
  - hierarchical routing with OSPF (inside a large domain)
    - Centrally Organized
  - inter-domain routing with BGP-4
    - Self-Organized

120

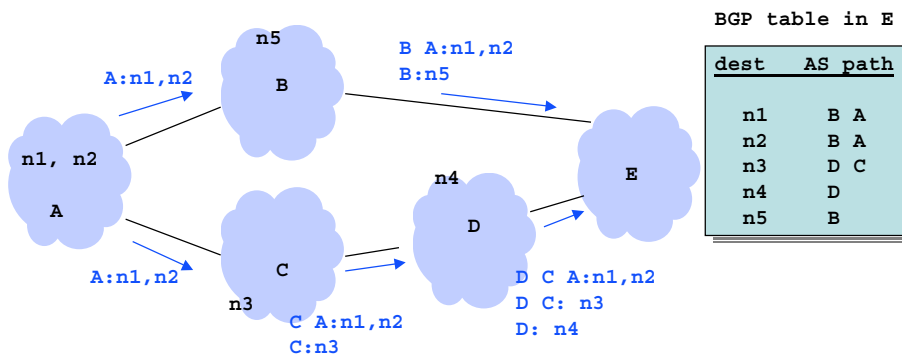
## Inter-Domain Routing

- ❑ Inter domain routing hierarchies
  - BGP-4: one level of hierarchy (one ARD is a virtual node in BGP)  
The ARD interconnection layer is **self-organized**
  - IDRP: several levels of hierarchy (ARDs can be aggregated)
- ❑ The principles of BGP-4 :
  - 1. routing method used in the higher level:
    - *path vector*
    - with *policy* routing
  - 2. mapping higher level nodes to lower level nodes
    - border gateways (= BGP speakers)
  - 3. inter-level routing information
    - summary link state records are injected into the interior routing protocol (OSPF, RIP, etc)

121

## Path Vector Routing

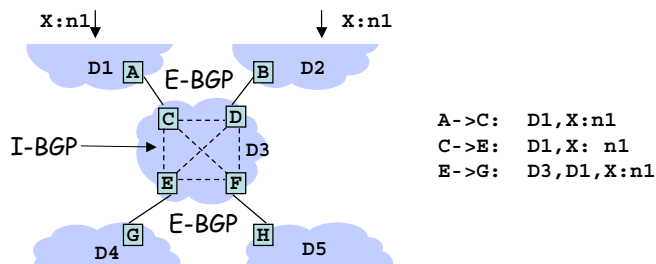
- ❑ a message between neighbours is a set of: (path, dest) (called "routes")
- ❑ every node (here: one AS) maintains a table of best paths known so far
- ❑ paths are announced to neighbours using the same principles as distance vector, ie. AS announces the best paths it knows
- ❑ applies to inter-domain routing
  - no global meaning for costs can be assumed (heterogeneous environment)
  - ASs want control over which paths they use (see policy routing, later)
- ❑ Q. Explain how E chooses the paths to n1 and n2
- ❑ Q. How can loops be avoided ?



122

## Border Gateways, E-BGP and I-BGP

- ❑ BGP runs on routers called *border gateways* = "BGP speakers"-- belong to one AS only
  - two border gateways per boundary
  - Q: compare to OSPF
- ❑ In addition, BGP speakers talk to each other inside the AS using "Internal-BGP" (I-BGP) over TCP connections
  - full mesh called the "BGP mesh"
  - I-BGP is the same as E-BGP except for one rule: routes learned from a neighbour in the mesh are not repeated inside the mesh ( Q. why ? )
  - Q: Is there a need for all BGP speakers in one network to be adjacent ?



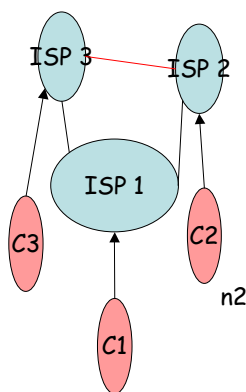
123

## F.2. Policy Routing

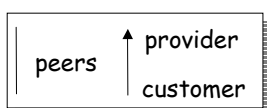
- ❑ Interconnection of ASs is self-organized
  - point to point links between networks: ex: EPFL to Switch, Switch to Telianet
  - interconnection points: NAP (Network Access Point), MAE (Metropolitan Area Ethernet), CIX (Commercial Internet eXchange), GIX (Global Internet eXchange), IXP, SFINX, LINX
- ❑ Mainly 3 types of relations, depending on money flows
  - customer: EPFL is customer of Switch. EPFL pays Switch
  - provider. Switch is provider for EPFL; Switch is paid by EPFL
  - peer: EPFL and CERN are peers: costs of interconnection is shared

124

## What is the Goal of Policy Routing ?

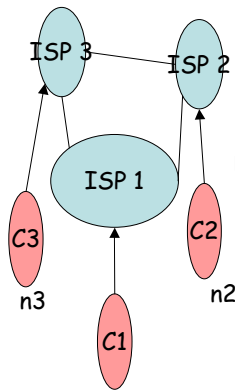


- ❑ Example:
  - ISP3-ISP2 is transatlantic link, cost shared between ISP2 and ISP 3
  - ISP 3- ISP 1 is a local, inexpensive link
  - Ci is customer of ISPi, ISPs are peers
- ❑ It is advantageous for ISP3 to send traffic to n2 via ISP1
- ❑ ISP1 does not agree to carry traffic from C3 to C2
  - ISP1 offers a "transit service" to C1 and a "non-transit" service to ISP 2 and ISP3
- ❑ The goal of "policy routing" is to support this and other similar requirements



125

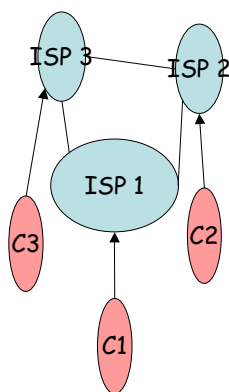
## How does Policy Routing Work ?



- Policy routing is implemented by rules imposed to BGP speakers inside an AS, who may
  - refuse to import or announce some paths
  - modify the attributes that control which path is preferred (see later)
- Example
  - ISP 1 announces to ISP 3 all networks of C1 - so that C1 can be reached by all sources in the world
  - ISP 1 announces to C1 all routes it has learnt from ISP3 and ISP2 - so that C1 can send traffic to all destinations in the world
  - ISP2 announces "ISP2 n2" to ISP3 and ISP1 ; assume that ISP1 announces "ISP1 ISP2 n2" to ISP3.
  - ISP 3 has two routes to n2: "ISP2 n2" and "ISP1 ISP2 n2"; assume that ISP3 gives preference to the latter
  - packets from n3 to n2 are routed via ISP1 - undesired
  - solution: ISP 1 announces to ISP3 only routes to ISP3's customers

126

## Typical Policy Routing Rules



- Provider (ISP1) to customer (C1)
  - announce all routes learnt from other ISs
  - import only routes that belong to domain C1  
example: import from EPFL only one route 128.178/15
- Customer (C1) to Provider (ISP1)
  - announce all routes that belong to domain C1
  - import all routes
- Peers (ISP1 to ISP3)
  - announce only routes to all customers of ISP1
  - import only routes to ISP3's customer
  - these routes are defined as part of peering agreement
- The rules are defined by every AS (self-organization) and implemented in all BGP speakers in one AS

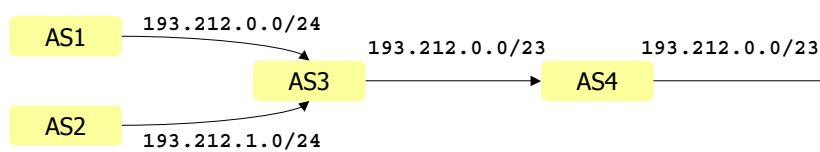
127

## F.3. Aggregation

- ❑ Domains that do not have a default route (i.e. all transit ISPs) must know all routes in the world (> 120 000)
  - in IP routing tables unless default routes are used
  - in BGP announcements
- ❑ Aggregation is a way to reduce the number of routes

128

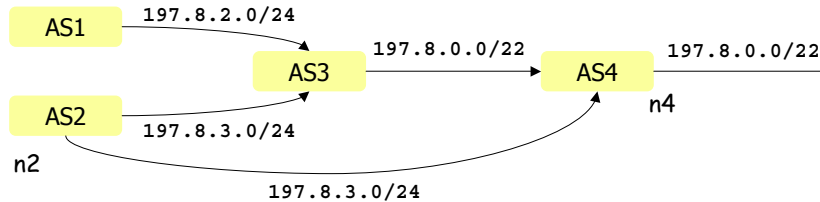
### Aggregation Example 1



● AS1: 193.212.0.0/24	AS_PATH: 1
● AS2: 193.212.1.0/24	AS_PATH: 2
● AS3: 193.212.0.0/23	AS_PATH: 3 {1 2}
● AS4: 193.212.0.0/23	AS_PATH: 4 3 {1 2}

129

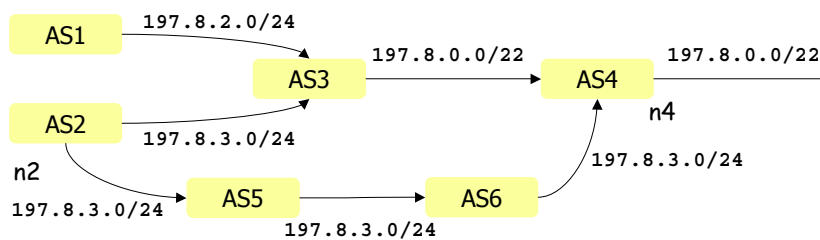
## Aggregation Example 2



- ❑ AS4 receives
  - 197.8.0.0/22 AS\_PATH: 3 {1 2}
  - 197.8.3.0/24 AS\_PATH: 2
- ❑ Both routes are injected into AS4's routing tables
- ❑ Q: what happens to packets from n4 to n2 ?

130

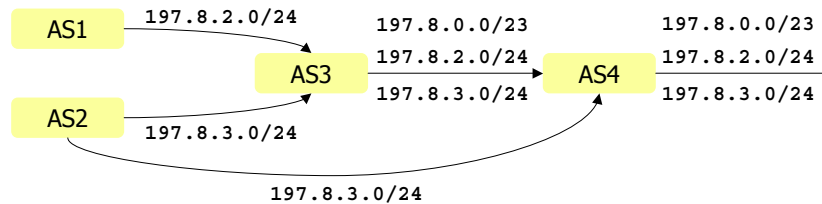
## Aggregation Example 3



- ❑ AS4 receives
  - 197.8.0.0/22 AS\_PATH: 3 {1 2}
  - 197.8.3.0/24 AS\_PATH: 6 5 2
- ❑ Both routes are received by AS4; only shortest AS paths routes are injected into routing tables
  - Q: what happens to packets from n4 to n2 ?

131

## Example Without Aggregation



- Q: If AS3 does not aggregate, what are the routes announced by AS 4 ? Is there any benefit ?

132

## F.4. BGP (Border Gateway Protocol)

- BGP-4, RFC 1771
- AS border router - BGP speaker
  - peer-to-peer relation with another AS border router
  - connected communication
    - on top of a TCP connection, port 179 (vs. datagram (RIP, OSPF))
  - external connections (E-BGP)
    - with border routers of different AS
  - internal connections (I-BGP)
    - with border routers of the same AS
  - BGP only transmits modifications

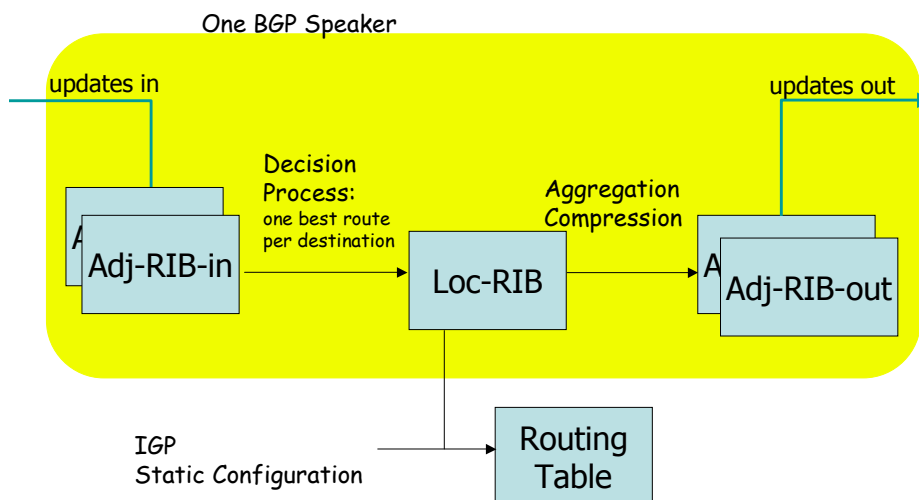
133

# Routes

- ❑ **Route** - unit of information; contains:
  - destination (subnetwork prefix) *A*
  - path to the destination (AS-PATH)
  - attributes
    - degree of preference (LOCAL-PREF)
    - origin of announcement (ORIGIN)
    - others, see later
- ❑ Advertised between a pair of BGP speakers
- ❑ Stored locally in RIBs (Routing Information Base)
- ❑ Every BGP speaker can add or modify the path attributes, using its *decision process*

134

## Routing Information Bases



135

## Operation of BGP Speaker

BGP speaker :

- ❑ stores received routes in **Adj-RIB-in**
  - one per BGP peer (internal or external)
- ❑ applies decision process and stores results in **Loc-RIB** (global to BGP speaker)
  - decide which routes to accept
  - decide how to rank them (set LOCAL-PREF)
  - decide which routes to export and with which attributes
- ❑ dispatches results per outgoing interface into **Adj-RIB-out** (one per BGP peer), after aggregation and information reduction
  
- ❑ maintains adjacency to peers (over TCP connection): open, keep-alive
  
- ❑ sends updates when Adj-RIB-out changes
  
- ❑ Write forwarding entries in its routing table; redistributes routes learnt from E-BGP from Loc-RIB into IGP and vice-versa, unless other mechanisms are used (See Examples)

136

## BGP messages

- ❑ 4 types
  - OPEN
  - KEEPALIVE
  - NOTIFICATION
  - UPDATE
- ❑ Size: from 19 to 4096 bytes
- ❑ Security by MD5

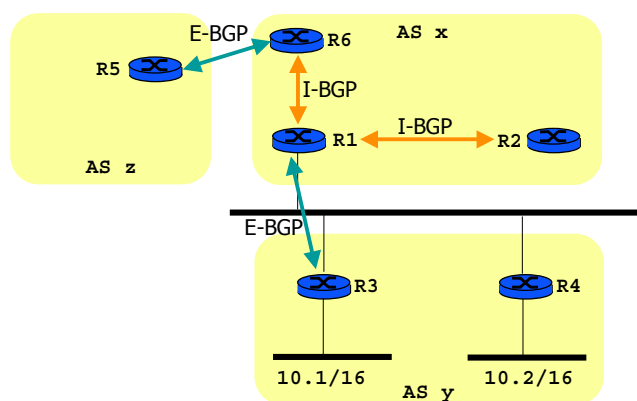
137

## Route Attributes

- ❑ Well-known Mandatory
  - ORIGIN (route learnt from IGP, BGP or static)
  - AS-PATH
  - NEXT-HOP (see later)
- ❑ Well-known Discretionary
  - LOCAL-PREF (see later)
  - ATOMIC-AGGREGATE (= route cannot be dis-aggregated)
- ❑ Optional Transitive
  - MULTI-EXIT-DISC (MED)(see later)
  - AGGREGATOR (who aggregated this route)
- ❑ Optional Nontransitive
  - WEIGHT (see later)

138

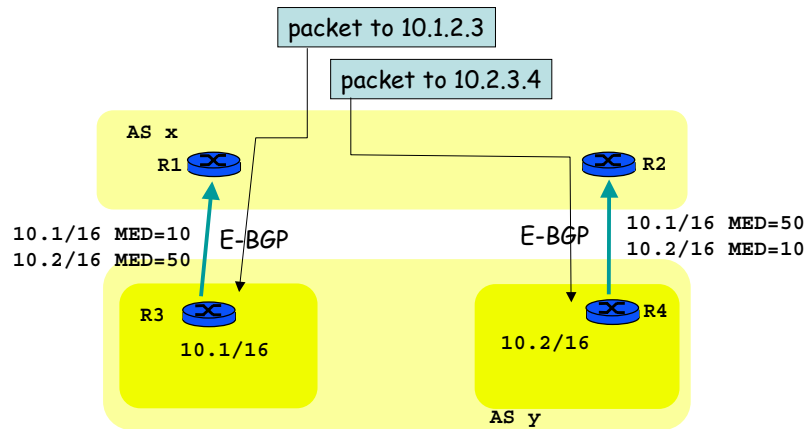
## NEXT-HOP



- ❑ R3 advertises 10.2/16 to R1, NEXT-HOP = R4 IP address
- ❑ R6 advertises 10.2/16 to R5, NEXT-HOP = R6 IP address
- ❑ Q. where is such a scenario likely to happen ?

139

## MULTI-EXIT-DISC (MED)



- ❑ One AS connected to another over several links
  - ex: multinational company connected to worldwide ISP
  - AS y advertises its prefixes with different MEDs (low = preferred)
  - If AS x accepts to use MEDs put by ASy: traffic goes on preferred link

140

## MED Example

- ❑ Q1: by which mechanisms will R1 and R2 make sure that packets to ASy use the preferred links ?
- ❑ Q2: router R3 crashes; can 10.1/16 still be reached ? explain the sequence of actions.

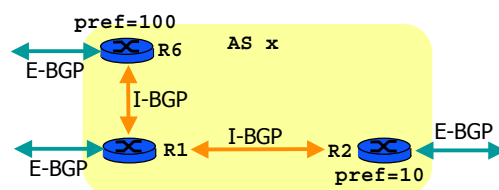
141

## MED Question

- ❑ Q1: Assume now ASx and ASy are peers (ex: both are ISPs). Explain why ASx is not interested in taking MED into account.
- ❑ Q2: By which mechanisms can ASx pick the nearest route to ASy ?

142

## LOCAL-PREF

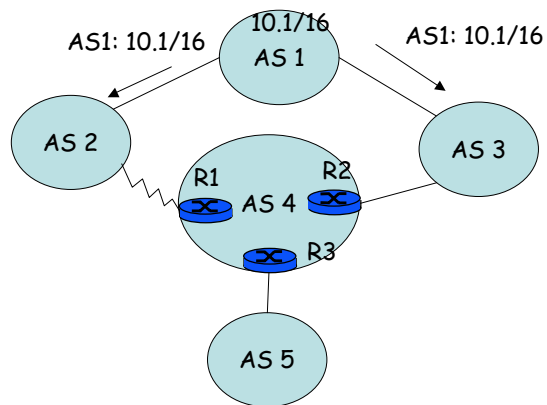


- ❑ Used inside an AS to select a best *AS path*
  - ❑ Assigned by border router when receiving route over E-BGP
    - Propagated without change over I-BGP
  - ❑ Example
    - R6 associates pref=100, R2 pref=10
    - R1 chooses the largest preference
- `bgp default local-preference pref-value`

143

## LOCAL-PREF Example

- ❑ Q1: The link AS2-AS4 is expensive. How should AS 4 set local-prefs on routes received from AS 3 and AS 2 in order to route traffic preferably through AS 3 ?
- ❑ Q2: Explain the sequence of events for R1, R2 and R3.



144

## LOCAL-PREF Question

- ❑ Q: Compare MED to LOCAL-PREF

145

## Choice of the best route

- ❑ Done by decision process ; result is: route installed in Loc-RIB
- ❑ At most one best route to exactly the same prefix is chosen
  - Only one route to 2.2/16 can be chosen
  - But there can be different routes to 2.2.2/24 and 2.2/16
- ❑ Decision Process uses the following priorities (for example)
  1. Highest LOCAL-PREF
  2. Shortest AS-PATH
  3. Lowest MED, if taken seriously by this network
  4. E-BGP > I-BGP
  5. Shortest path to NEXT-HOP, according to IGP
  6. Lowest BGP identifier

146

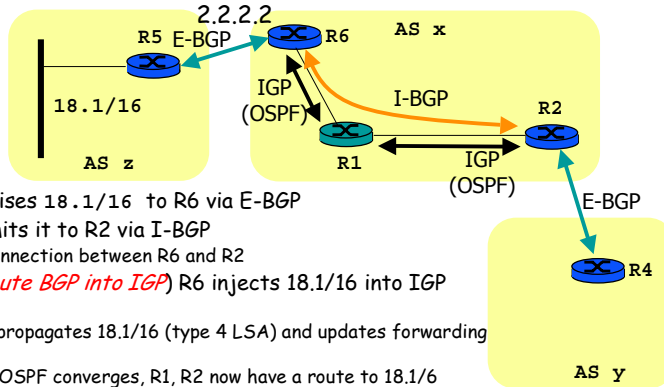
## F.5. Interaction BGP—IGP—Packet Forwarding

There are three interactions between BGP and internal routing that you have to know

- ❑ **Redistribution:** routes learnt by BGP are passed to IGP (ex: OSPF)
  - Called "redistribution of BGP into OSPF"
  - OSPF propagates the routes using type 4 LSAs to all routers in OSPF cloud
- ❑ **Injection:** routes learnt by BGP are written into the forwarding table of this router
  - Routes do not propagate; this helps only this router
- ❑ **Synchronization:** see later

147

## Redistribution Example

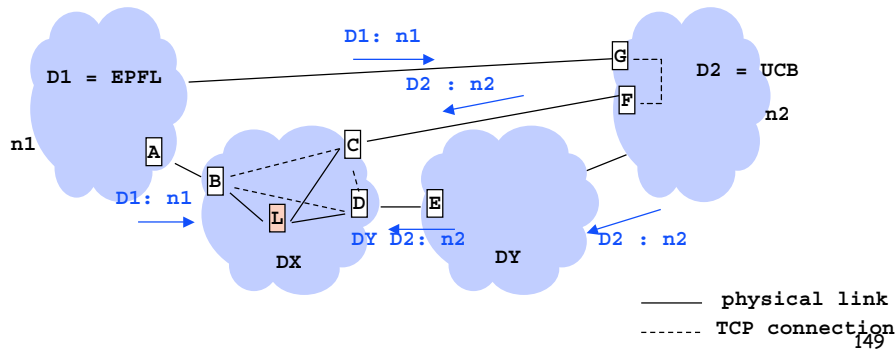


- ❑ R5 advertises 18.1/16 to R6 via E-BGP
- ❑ R6 transmits it to R2 via I-BGP
  - TCP connection between R6 and R2
- ❑ (*redistribute BGP into IGP*) R6 injects 18.1/16 into IGP (OSPF)
  - OSPF propagates 18.1/16 (type 4 LSA) and updates forwarding tables
  - After OSPF converges, R1, R2 now have a route to 18.1/6
- ❑ R2 advertises route to R4 via E-BGP
  - (*synchronize with IGP*) R2 must wait for the OSPF entry to 18.1/6 before advertising via E-BGP
- ❑ Packet to 18.1/16 from AS y finds forwarding table entries in R2, R1 and R6

148

## Example with Re-Distribution

- by \_\_\_\_, F learns from G the route D2-D1-n1
- C redistributes the external route D2:n2 into OSPF
- by \_\_\_\_, D learns the route D2:n2; by \_\_\_\_, D learns the route DYD2:n2; D selects D2:n2 and does not redistribute it to OSPF
- by \_\_\_\_, B learns the route D2:n2
- by \_\_\_\_, A learns the route DX:D2:n2
- by \_\_\_\_, L learns the route to n2 via C



—— physical link  
 - - - - TCP connection

149

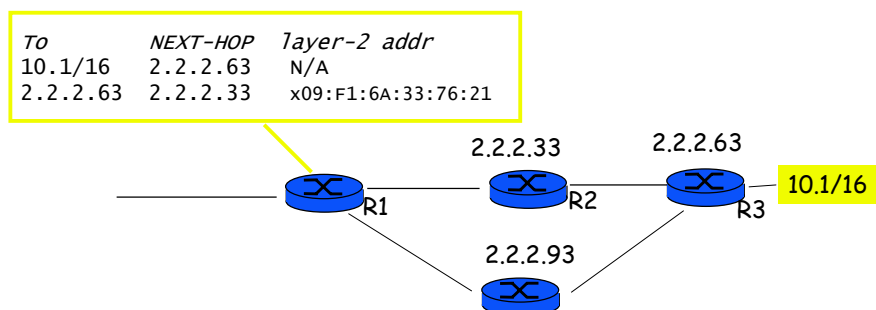
## Re-Distribution Considered Harmful

- ❑ In practice, operators avoid re-distribution of BGP into IGP
  - Large number of routing entries in IGP
  - Reconvergence time after failures is large if IGP has many routing table entries
- ❑ A classical solution is based on *recursive table lookup*
  - When IP packet is submitted to router, the forwarding table may indicate a "NEXT-HOP" which is not on-link with router
  - A second table lookup needs to be done to resolve the next-hop into an on-link neighbour
    - in practice, second lookup is done in advance - not in real time- by preprocessing the routing table

150

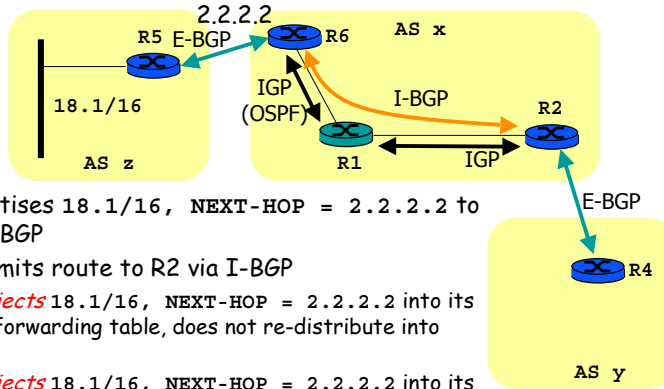
## Example: Recursive Table Lookup

- ❑ At R1, data packet to 10.1.x.y is received
- ❑ The forwarding table at R1 is looked up
  - Q: what are the next events ?



151

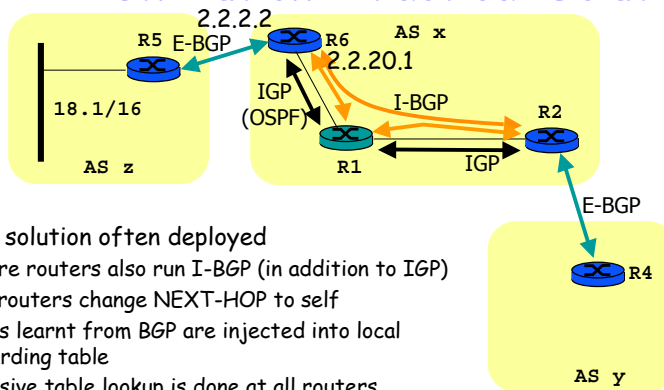
## Avoid Redistribution: Combine Recursive Lookup and NEXT-HOP



- ❑ R5 advertises 18.1/16, NEXT-HOP = 2.2.2.2 to R6 via E-BGP
- ❑ R6 transmits route to R2 via I-BGP
  - R6 *injects* 18.1/16, NEXT-HOP = 2.2.2.2 into its local forwarding table, does not re-distribute into OSPF
  - R2 *injects* 18.1/16, NEXT-HOP = 2.2.2.2 into its local forwarding table
- ❑ Data packet to 18.1.2.3 is received by R2
  - Recursive table lookup at R2 can be used
  - Q: there is a problem at R1: how can we solve it ?

152

## Avoid Redistribution: Practical Solution



- ❑ Practical solution often deployed
  - All core routers also run I-BGP (in addition to IGP)
  - Edge routers change NEXT-HOP to self
  - Routes learnt from BGP are injected into local forwarding table
  - Recursive table lookup is done at all routers
  - Q: repeat the sequence of previous slide with this new assumption
- ❑ Potential problem: I-BGP mesh -> use reflectors
- ❑ IGP handles only internal networks - very few

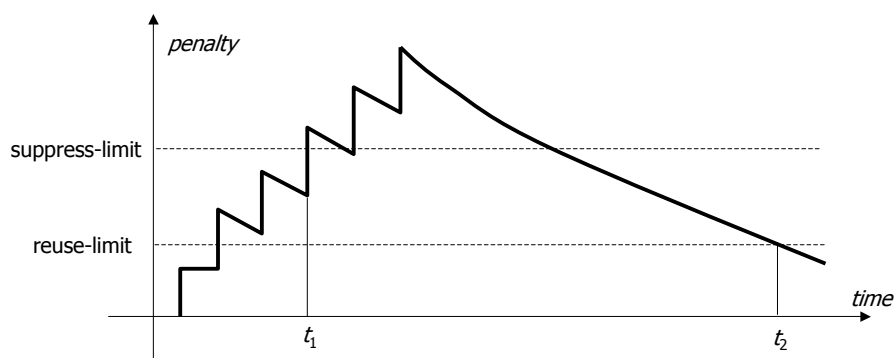
153

## F.6. Other Mechanisms in BGP Route Flap Dampening

- ❑ Route modification propagates everywhere
- ❑ Sometimes routes are *flapping*
  - successive UPDATE and WITHDRAW
  - caused for example by BGP speaker that often crashes and reboots
- ❑ Solution:
  - decision process eliminates flapping routes
- ❑ How
  - withdrawn routes are kept in Adj-RIN-in
  - if comes up again soon (ie : flap), route receives a penalty
  - penalty fades out exponentially
  - used to suppress or restore routes

154

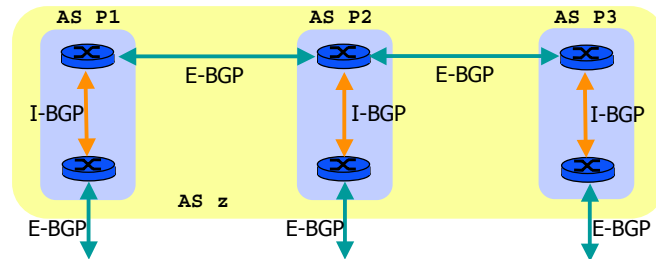
## Route Flap Dampening



- ❑ Route suppressed at  $t_1$ , restored at  $t_2$

155

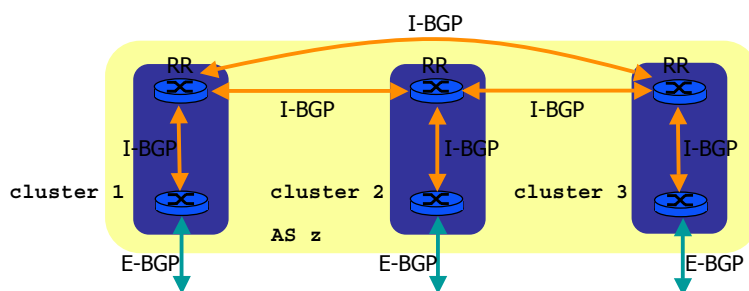
## Avoid I-BGP Mesh: Confederations



- AS decomposed into sub-AS
  - private AS number
  - similar to OSPF areas
    - I-BGP inside sub-AS (full interconnection)
    - E-BGP between sub-AS

156

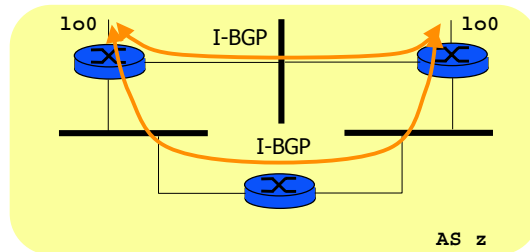
## Avoid I-BGP Mesh : Route reflectors



- Cluster of routers
  - one I-BGP session between one client and RR
  - CLUSTER\_ID
- Route reflector
  - re-advertises a route learnt via I-BGP
  - to avoid loops
    - ORIGINATOR\_ID attribute associated with the advertisement

157

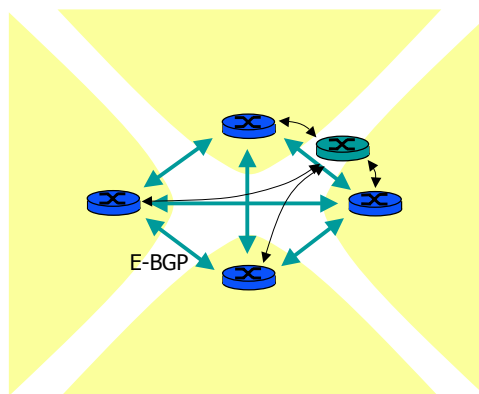
## I-BGP configuration



- ❑ I-BGP configured on loopback interface (lo0)
  - interface always up
  - IP address associated with the interface
  - IGP routing guarantees packet forwarding to the interface

158

## Avoid E-BGP mesh: Route server



- ❑ At interconnection point
- ❑ Instead of  $n(n-1)/2$  peer to peer E-BGP connections
- ❑  $n$  connections to Route Server
- ❑ To avoid loops ADVERTISER attribute indicates which router in the AS generated the route

159

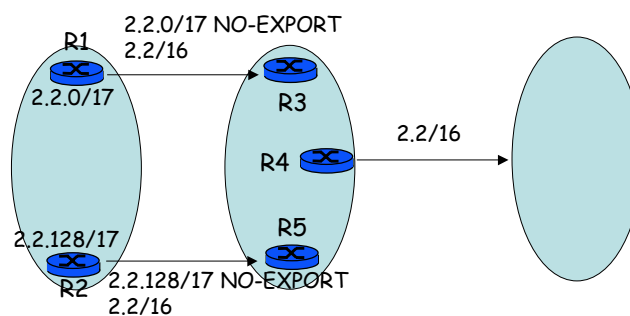
## Communities

- ❑ Other attributes can be associated with routes in order to *simplify* rules. They are called « communities »
  - Pre-defined: Example: NO-EXPORT ( a well known, pre-defined attribute) - see later for an example
  - Defined by one AS (a label of the form ASN:x where AS= AS number, x = a 2 byte-number)

160

## NO-EXPORT

- ❑ Written on E-BGP by one AS, transmitted on I-BGP by accepting AS, not forwarded
- ❑ Example: AS2 has different routes to AS1 but AS2 sends only one aggregate route to AS3
  - simplifies the aggregation rules at AS2
  - What is the route followed by a packet sent to 2.2.48 received by R4 ?



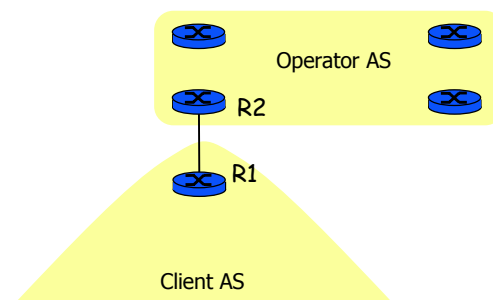
161

## F.7. Examples

- ❑ Dual Homing
- ❑ Hot potato routing

162

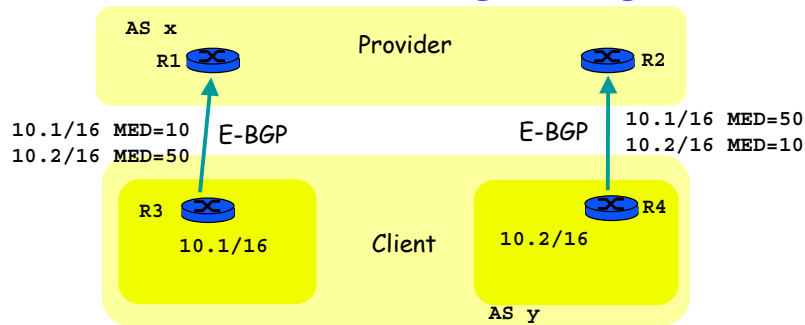
### Ex1: Stub Area



- ❑ BGP not needed between Client and Operator
- ❑ No AS number for client
- ❑ R2 learns all prefixes in Client by static configuration or RIP on link R1–R2
- ❑ Example: EPFL and Switch
- ❑ Q: what if R1 fails ?

163

## Ex2: Stub Area, Dual Homing to Single Provider

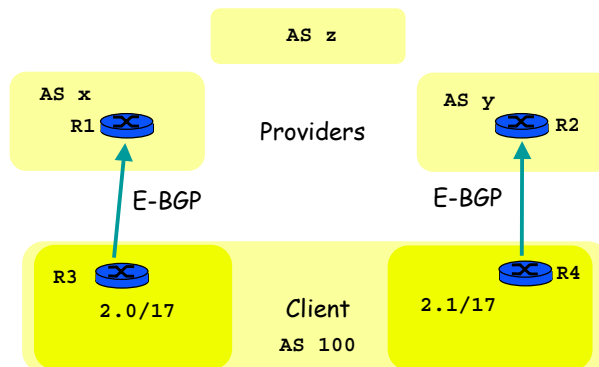


### □ With numbered Client AS

- Use MED to share traffic from ISP to Client on two links
- Use Client IGP configuration to share traffic from Client to two links
- Q1: is it possible to avoid distributing BGP routes into Client IGP ?
- Q2: is it possible to avoid assigning an AS number to Client ?
- Q3: is it possible to avoid BGP between Client and Provider ?

164

## Ex3: Stub Area, Dual Homing to Several Providers



- Client has own address space and AS number
- Q: how can routes be announced between AS 100 and AS x ? AS x and AS z ?
- Q: assume Client wants most traffic to favour AS x. How can that be done ?

165

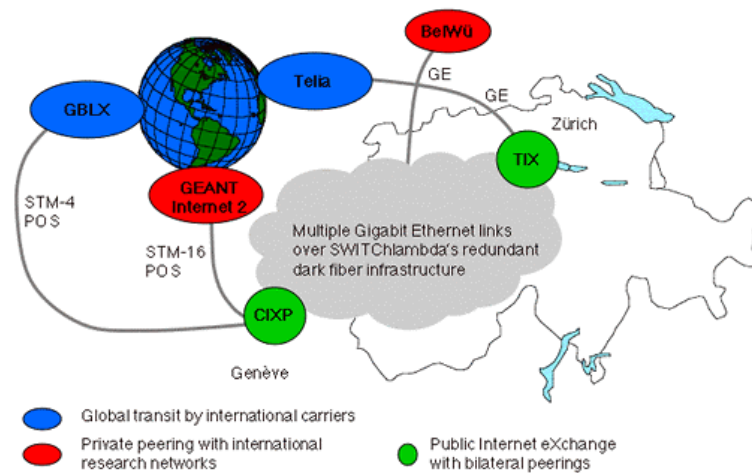
## Ex4: Hot Potato Routing



- ❑ Packets from Customer 2 to Customer 1
  - Both R21 and R22 have a route to Customer 1
  - Shortest path routing favours R21
  - Q1: by which mechanism is that done ?
- ❑ Q2: what is the path followed in the reverse direction ?

166

## F.8. Illustrations: Switch



167

# An Interconnection Point



[E-Mail](#) | [Credits](#)

[Expand all](#) | [Collapse all](#)

## General Information

### Services

### Costs

[Membership fees](#)  
[Connection fees](#)

### Legal

[Articles of association](#)  
[Peering Policy](#)  
[Connection agreement](#)

### Members

[Member list](#)  
[Board members](#)  
[Membership application](#)

### Member Login

### Tech Corner

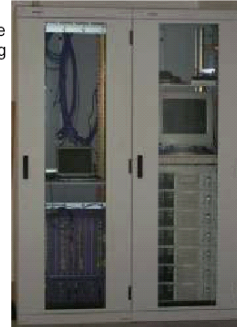
### Links

## Welcome to swissix

The Swissix (Swiss Internet Exchange) in Zurich, Switzerland, is now open. We are pleased to welcome ISPs and hosting companies as members and peering partners.

With continued growth of Internet traffic, we want to make sure that there is sufficient reliability built into the Swiss Internet. By exchanging traffic at multiple exchanges points, you can help ensure that consumers have fast Internet access and network operators have multiple routes for their traffic flows.

The Swiss Internet Exchange (swissix) is a neutral and independent exchange and a place for Internet Service Providers (ISPs) to interconnect and exchange IP traffic with each other at a national or international level.



168



[E-Mail](#) | [Credits](#)

[Expand all](#) | [Collapse all](#)

## General Information

### Services

### Costs

[Membership fees](#)  
[Connection fees](#)

### Legal

[Articles of association](#)  
[Peering Policy](#)  
[Connection agreement](#)

### Members

[Member list](#)  
[Board members](#)  
[Membership application](#)

### Member Login

### Tech Corner

### Links

## Membership fees

The yearly membership fee is CHF 100.- per company. The membership fee is not refundable.

## Connection fees

**Action: Till end of 2003 all port and connection fees are free.**

The connection fees consist of a monthly and a one-time installation fee and depend on the connection port type.

Port type	Monthly CHF	One-time CHF
100BaseTX	275.00	1150.00
1000BaseSX/LX	1520.00	3000.00

Prices are excluding VAT (7.6%)

Deployment of 100BaseTX: immediately  
Deployment of 1000BaseSX/LX: 2-4 weeks

169

from [www.ris.ripe.net](http://www.ris.ripe.net): all routes to 128.178.0.0/15 on RIPE Route Collectors

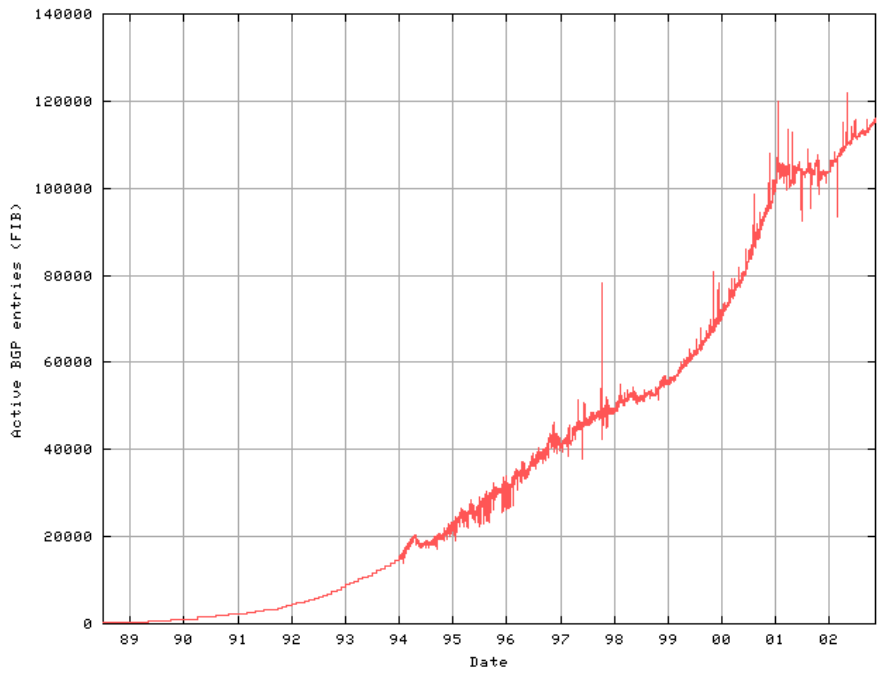
Type			Next HOP		MED	Origin	Community	RRC ID
A	<a href="#">128.178.0.0/15</a>	2003-10-02 05:05:49Z	<a href="#">129.250.0.2/32</a>	<a href="#">129.250.0.2/32</a>	9	Not defined	<a href="#">2914 1299 559</a>	2914:420 2914:2000 2914:3000 RIPE NCC
A	<a href="#">128.178.0.0/15</a>	2003-10-02 06:16:00Z	<a href="#">193.10.252.5</a>	<a href="#">193.10.252.5</a>	0	IGP	<a href="#">2603 3356 1299 559</a>	2603:666 3356:2 3356:86 3356:507 3356:666 3356:2076 Netnod
A	<a href="#">128.178.0.0/15</a>	2003-10-02 06:16:17Z	<a href="#">194.68.48.1</a>	<a href="#">194.68.48.1</a>	0	IGP	<a href="#">12381 1653 2603 20965 559</a>	12381:1653 Netnod
A	<a href="#">128.178.0.0/15</a>	2003-10-02 06:16:37Z	<a href="#">194.68.48.1</a>	<a href="#">194.68.48.1</a>	0	IGP	<a href="#">12381 1653 2603 3356 1299 559</a>	12381:1653 Netnod
A	<a href="#">128.178.0.0/15</a>	2003-10-02 06:21:08Z	<a href="#">193.10.252.5</a>	<a href="#">193.10.252.5</a>	0	IGP	<a href="#">2603 20965 559</a>	2603:222 2603:666 20965:155 Netnod
A	<a href="#">128.178.0.0/15</a>	2003-10-02 06:21:17Z	<a href="#">194.68.48.1</a>	<a href="#">194.68.48.1</a>	0	IGP	<a href="#">12381 1653 2603 20965 559</a>	12381:1653 Netnod
A	<a href="#">128.178.0.0/15</a>	2003-10-02 07:24:06Z	<a href="#">129.250.0.2/32</a>	<a href="#">129.250.0.2/32</a>	9	Not defined	<a href="#">2914 3549 559</a>	2914:420

170

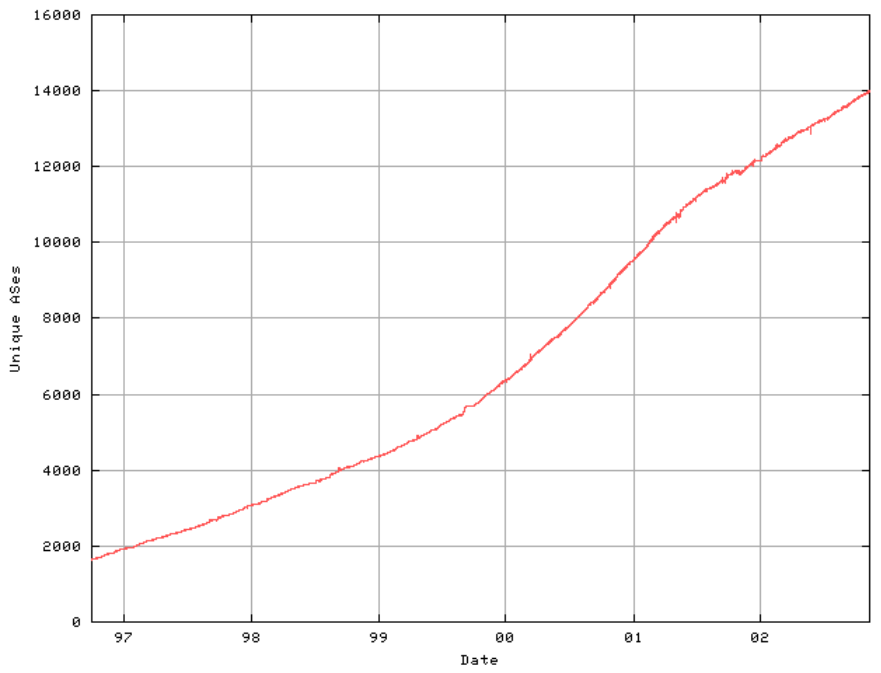
## Some statistics

- ❑ Number of routes
  - 1988-1994: exponential increase
  - 1994-1995: CIDR
  - 1995-1998: linear increase (10000/year)
  - 1999-2000: return to exponential increase (42% per year)
  - since 2001: return to linear increase, ~120,000
- ❑ Number of ASs
  - 51% per year for 4 last years
  - 14000 AS effectively used
- ❑ Number of IP addresses
  - 162,128,493 (Jul 2002)
  - 7% per year

171

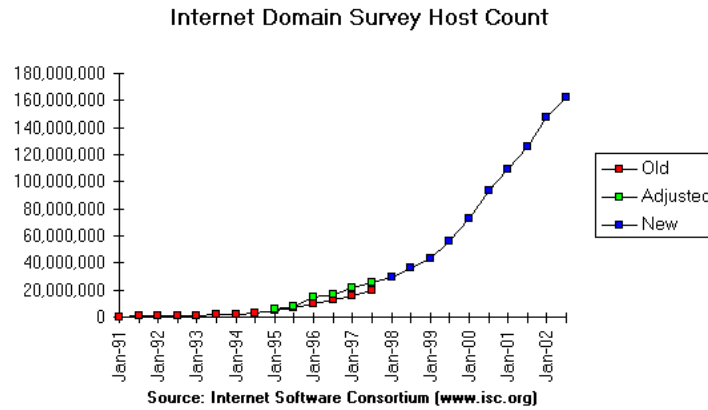


172



173

## Number of hosts



174

## BGP statistics

BGP routing table entries examined:	17013
Total ASes present in the Internet Routing Table:	4042
Origin-only ASes present in the Internet Routing Table:	12159
Transit ASes present in the Internet Routing Table:	1883
Transit-only ASes present in the Internet Routing Table:	63
Average AS path length visible in the Internet Routing Table:	5.3
Max AS path length visible:	23
Number of addresses announced to Internet:	1182831464
Equivalent to 70 /8s, 128 /16s and 147 /24s	
Percentage of available address space announced:	31.9
Percentage of allocated address space announced:	58.5

175



## Exercise

- What ASs does EPFL receive service from ?
- What ASs does Switch receive service from ?
- Find the names of the networks that have these AS numbers

178

## Exercise

- Lookup <http://rpsl.info.ucl.ac.be>. to find out the relationships between Switch and other providers
- How does the software on this site decide whether a relationship is client, provider or peer ?

179

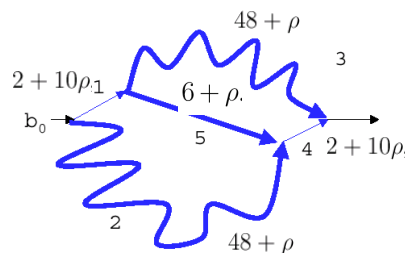
## G. Load Dependent Routing

- ❑ Instead of maximizing a "path quality" metric (nb hops, delay) assume we want to maximize the *total network utility*
  - for example: total transported flows
  - see congestion control chapter for other definitions
- ❑ how should routing be done ?
- ❑ Q1: show an example where shortest path routing does not provide the optimal total flow (where path cost is static)
- ❑ One solution might be to take delay as the path cost
  - high load on a link  $\Rightarrow$  high cost  $\Rightarrow$  link is less used
  - however, this does not solve the problem: there is the Braess paradox

180

## Braess Paradox (1)

- ❑ Assume all flows pick the route with shortest delay
- ❑ Assume parallel paths exist and flows can make use of them
- ❑ Delay is function of load as given below; link 5 is (temporarily) closed
- ❑ Total offered load is  $b_0 = 6 \text{ Gb/s}$
- ❑ For example,
  - if we split traffic into : route 1-3:  $b = 1$ , route 2-4  $b = 5$
  - the delay along route 1-3 is 61, along route 2-4 is 105
  - thus the link costs will change and routing decisions will change also
- ❑ Eventually, there will be an equilibrium (called "Wardrop Equilibrium")
  - delay is equal on all competing routes
- ❑ Q: compute the equilibrium traffic flow on every link



181

## Braess Paradox (2)

- Q: same question when we open link 5 with delay function:

$$f_5(\rho) = 6 + \rho.$$

182

## Braess Paradox and Beyond

- With shortest delay routing, adding a new link may decrease overall throughput
  - shortest delay routing is not a global optimum
- The global optimum problem:
  - minimize total delay subject to flow constraints
  - this is a well posed optimization problem
  - the optimal solution depends on all flows
  - but it can be implemented in a distributed algorithm similar to TCP congestion control [BertsekasGallager“]
- In practice, it can be implemented in a network through a centralized network management procedure that updates the link costs (used by distance vector routing).
  - given link costs  $c_i$  and traffic matrix compute total throughput or average delay ( a hard optimization problem, solved with heuristics)
  - every few minutes, update the link costs in all routers - let the routing algorithm compute new paths

183

## Further Reading

- ❑ Slow convergence after route suppression - "BGP path exploration", similar to (but worse than) distance vector slow convergence. Is in the nature of path vector routing with explicit suppression.
  - Craig Labovitz, [Abha Ahuja](#), [Abhijit Bose](#), [Farnam Jahanian](#): Delayed Internet routing convergence. [IEEE/ACM Trans. Netw.](#) 9(3): 293-306 (2001)
- ❑ Route flap dampening slows down convergence
  - [Zhuoqing Morley Mao](#), Ramesh Govindan, [George Varghese](#), [Randy H. Katz](#): Route flap damping exacerbates internet routing convergence. [SIGCOMM 2002](#): 221-233
- ❑ Path vector + policy may suffer from incompatibilities (loops)
  - The stable paths problem and interdomain routing  
Griffin, T.G.; Shepherd, F.B.; Wilfong, G.  
[ACM/IEEE ToN April 2002](#), Page(s): 232-243

184

## References

- ❑ Timothy Griffin's home page at Intel
- ❑ RFC 1771 (BGP-4)
- ❑ C. Huitema, "Le Routage dans l'Internet"
- ❑ John W. Stewart III " BGP 4"
- ❑ [www.ris.ripe.net](http://www.ris.ripe.net) : AS paths
- ❑ [www.cidr-report.org](http://www.cidr-report.org) aggregation statistics
- ❑ [www.caida.org](http://www.caida.org) map of Internet
- ❑ [rpsl.info.ucl.ac.be](http://rpsl.info.ucl.ac.be) relations between ASs

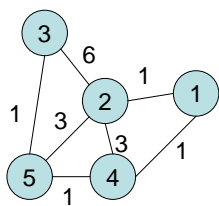
185

## Solutions

186

## Example

- Apply the theorem: write  $p^k(i,1)$ ,  $\text{pred}(i)$  and draw the shortest paths to node 1.



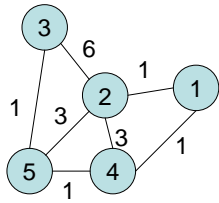
$i \backslash k$	0	1	2	3
1	0	0	0	0
2	$\infty$	1	1	1
3	$\infty$	$\infty$	7	3
4	$\infty$	1	1	1
5	$\infty$	$\infty$	2	2

$i$	1	2	3	4	5
$\text{pred}(i)$	4	2	3	4	5

187

## Impact of Initial Conditions

- ❑ Example: does the algorithm converge to the shortest path with initial condition as shown ?



$i \backslash k$	0	1	2	3	4
1	0	0	0	0	0
2	0	1	1	1	1
3	0	1	2	3	3
4	0	1	1	1	1
5	0	1	2	2	2

$i \backslash k$	0	1	2	3
1	0	0	0	0
2	6	1	1	1
3	1	1	2	3
4	1	1	1	1
5	0	2	2	2

188

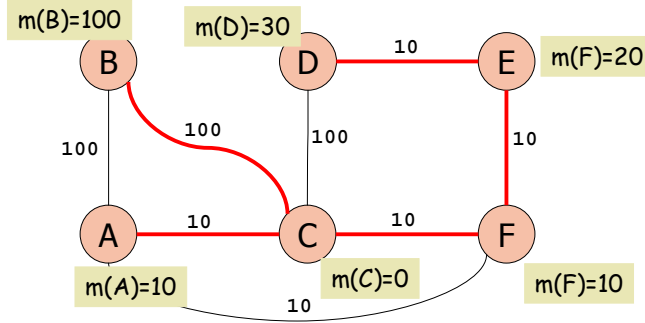
## Conclusions from Example 2

- ❑ Q: during convergence time, how are routing tables ?  
 A:
- they are incorrect
  - there are loops - packets are discarded (TTL expires)

189

## Test Your Understanding

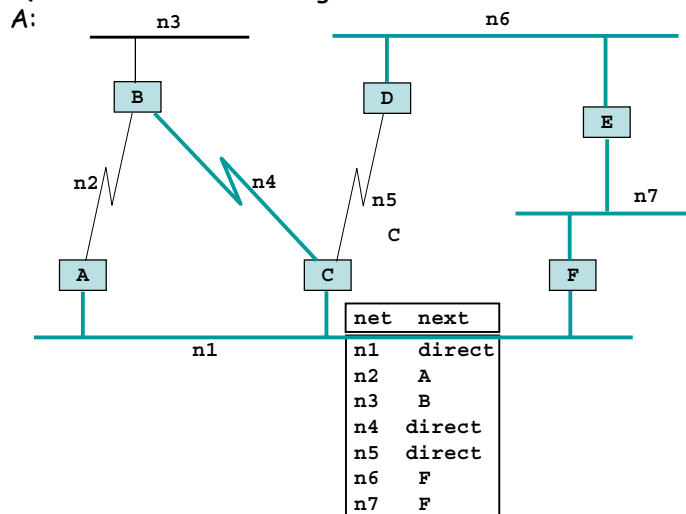
- ❑ Q1: Run Dijkstra at C  
A: (final step)
- ❑ Q2: What are the routing tables at C



190

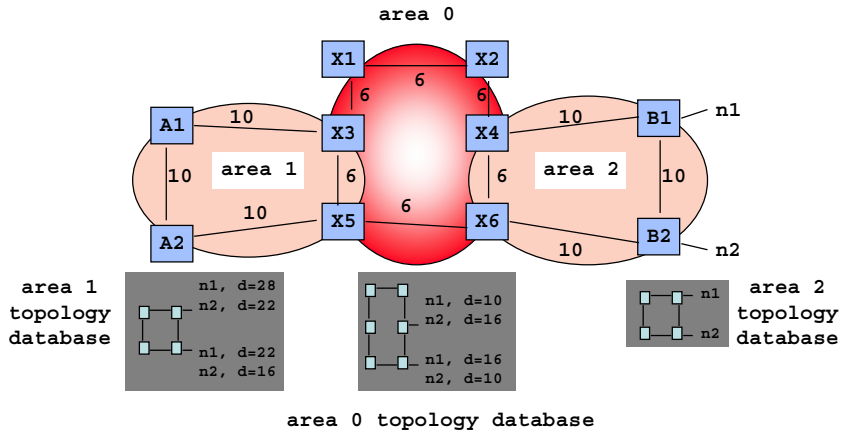
## Test Your Understanding

- ❑ Q2: What are the routing tables at C



191

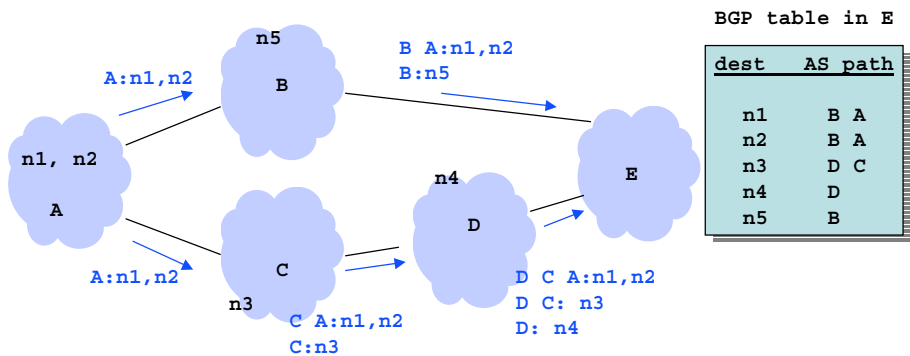
# Solution



192

# Path Vector Routing

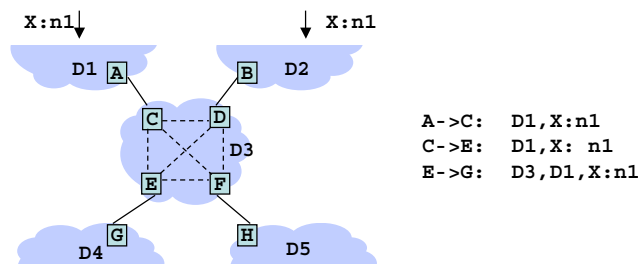
- ❑ Q. Explain how E chooses the paths to n1 and n2
  - A. E receives the routes "B A n1" and "D C A n1". E selects as best routes the ones with shorter AS path.
- ❑ Q. How can loops be avoided ?
  - A: BGP routers recognize looping announcements by the repetition of the same AS in the path. Such announcements are discarded



193

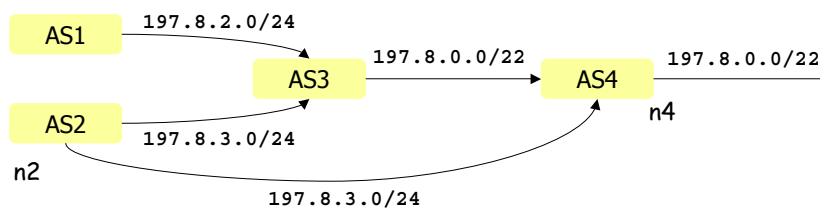
## Border Gateways, e-BGP and I-BGP

- BGP runs on routers called *border gateways* = "BGP speakers"-- belong to one AS only
  - Q: compare to OSPF
  - A: there is one single inter-area router per area boundary: it belongs to both areas
- In addition, BGP speakers talk to each other inside the AS using "Internal-BGP" (I-BGP) over TCP connections
  - I-BGP is the same as E-BGP except for one rule: routes learned from a neighbour in the mesh are not repeated inside the mesh (Q. why?)
    - A: otherwise loops cannot be avoided (same AS number!)
  - Q: Is there a need for all BGP speakers in one network to be adjacent?
    - A: no, they are generally not. The mesh is over TCP connections.



194

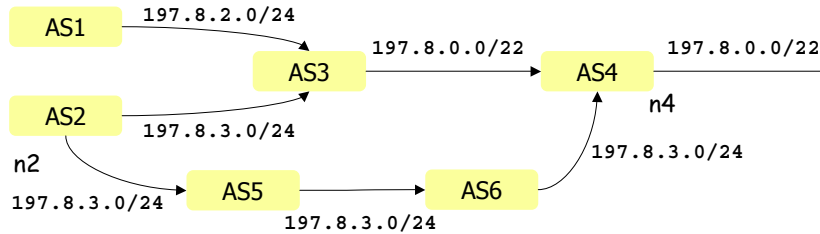
## Aggregation Example 2



- AS4 receives
  - 197.8.0.0/22 AS\_PATH: 3 {1 2}
  - 197.8.3.0/24 AS\_PATH: 2
- Both routes are injected into AS4's routing tables
  - Q: what happens to packets from n4 to n2?
  - A: it depends on the attributes set by the rules in AS4; by default, the direct route to n2 is preferred (fewer ASs in path). There are two routing entries in AS4 routers: one for 197.8.0.0/22 and one for 197.8.3.0/24. Longest prefix match in the packet forwarding algorithm ensures that packets to n2 go on the direct route.

195

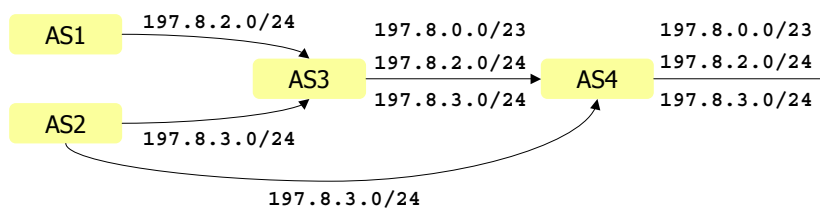
## Aggregation Example 3



- ❑ AS4 receives
  - 197.8.0.0/22 AS\_PATH: 3 {1 2}
  - 197.8.3.0/24 AS\_PATH: 6 5 2
- ❑ Both routes are received by AS4; only shortest AS paths routes are injected into routing tables Q: what happens to packets from n4 to n2 ?
  - A: now packets to n2 go via AS3

196

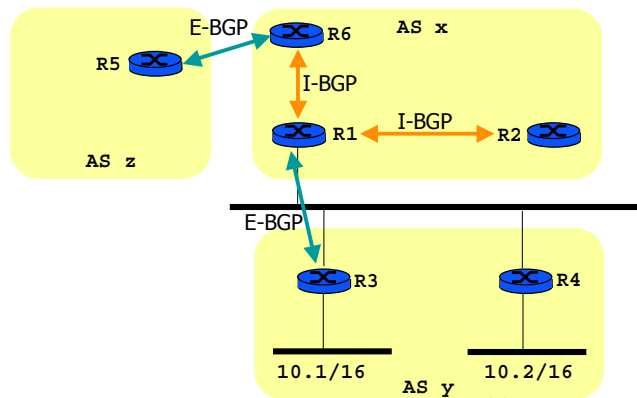
## Example Without Aggregation



- ❑ Q: If AS3 does not aggregate, what are the routes announced by AS 4 ? Is there any benefit ?
- ❑ A:
  - 197.8.0.0/23 AS\_PATH: 4 3
  - 197.8.2.0/24 AS\_PATH: 4 3 1
  - 197.8.3.0/24 AS\_PATH: 4 2
- ❑ A: there is no benefit since all routes go via AS 4 anyhow. AS4 should aggregate.

197

## NEXT-HOP



- ❑ R3 advertises 10.2/16 to R1, NEXT-HOP = R4 IP address
- ❑ R6 advertises 10.2/16 to R5, NEXT-HOP = R6 IP address
- ❑ Q. where is such a scenario likely to happen ?
- ❑ A: in interconnection points with many providers interconnected on one LAN

198

## MED Example

- ❑ Q1: by which mechanisms will R1 and R2 make sure that packets to ASy use the preferred links ?
- A:
  - R1 and R2 exchange their routes to ASy via I-BGP
  - R1 has 2 routes to 10.1/16, one of them learnt over E-BGP; prefers route via R1; injects it into IGP
  - R1 has 2 routes to 10.2/16, one of them learnt over E-BGP; prefers route via R2; does not inject a route to 10.2/16 into IGP
- ❑ Q2: router R3 crashes; can 10.1/16 still be reached ? explain the sequence of actions.
- A:
  - R1 clears routes to ASy learnt from R1 (keep-alive mechanism)
  - R2 is informed of the route suppression by I-BGP
  - R2 has now only 1 route to 10.1/16 and 1 route to 10.2/16; keeps both routes in its local RIB and injects them into IGP since both were learnt via E-BGP
  - traffic to 10.1/16 now goes to R2

199

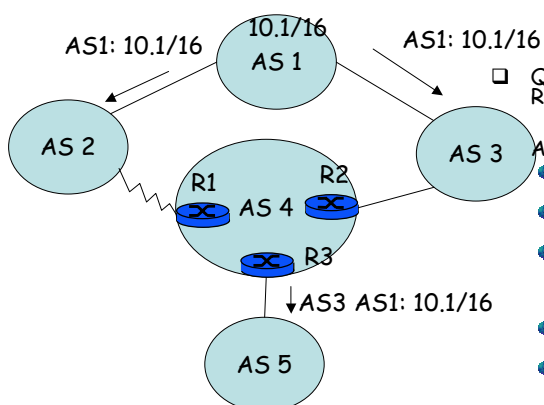
## MED Question

- ❑ Q1: Assume now ASx and ASy are peers (ex: both are ISPs). Explain why ASx is not interested in taking MED into account.  
A: ASx is interested in sending traffic to ASy to the nearest exit, avoiding transit inside ASx as much as possible. Thus ASx will choose the nearest route to ASy, and will ignore MEDs
- ❑ Q2: By which mechanisms can ASx pick the nearest route to ASy ?  
A: it depends on the IGP. With OSPF: all routes to ASy are injected into OSPF by means type 5 LSAs. These LSAs say: send to router R3 or R4. Every OSPF router inside ASx knows the cost (determined by OSPF weights) of the path from self to R3 and R4. Packets to 10.1/16 and 10.2/16 are routed to the nearest among R3 and R4 (nearest = lowest OSPF cost).

200

## LOCAL-PREF Example

- ❑ Q1: The link AS2-AS4 is expensive. How should AS 4 set local-prefs on routes received from AS 3 and AS 2 in order to route traffic preferably through AS 3 ?  
A: for example: set LOCAL-PREF to 100 to all routes received from AS 3 and to 50 to all routes received from AS 2



- ❑ Q2: Explain the sequence of events for R1, R2 and R3

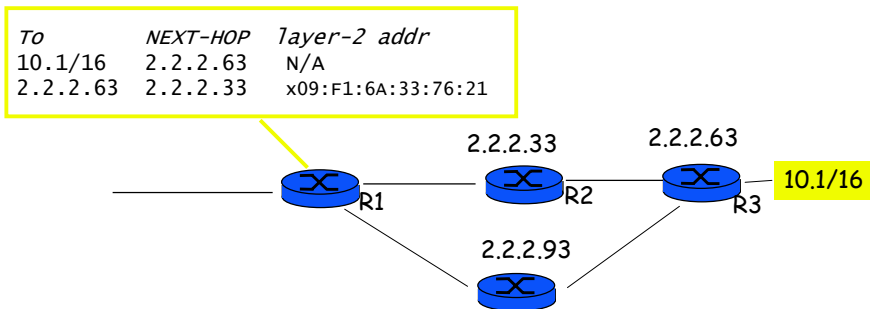
- A:
- R1 receives the route AS2 AS1 10.1/16 over E-BGP; sets LOCAL-PREF to 50
  - R2 receives the route AS3 AS1 10.1/16 over E-BGP; sets LOCAL-PREF to 100
  - R3 receives AS2 AS1 10.1/16, LOCAL-PREF=50 from R1 over I-BGP and AS3 AS1 10.1/16, LOCAL-PREF=100 from R1 over I-BGP
  - R3 selects AS3 AS1 10.1/16, LOCAL-PREF=100 and installs it into local-RIB
  - R3 announces only AS3 AS1 10.1/16 to AS 5

201



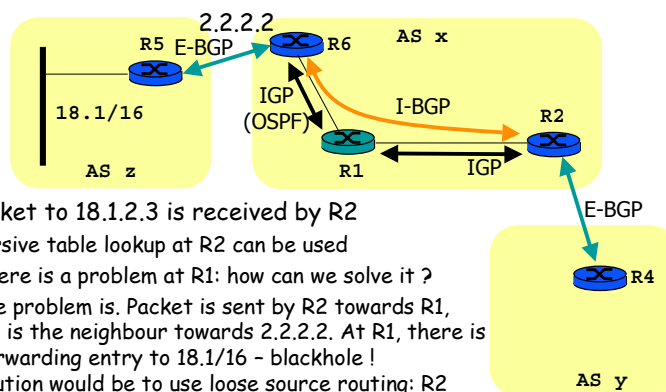
## Example: Recursive Table Lookup

- ❑ At R1, data packet to 10.1.x.y is received
- ❑ The forwarding table at R1 is looked up
  - Q: what are the next events ?
  - A: first, the nex-hop 2.2.2.63 is found; a second lookup for 2.2.2.63 is done; the packet is sent to MAC address x09:F1:6A:33:76:21



204

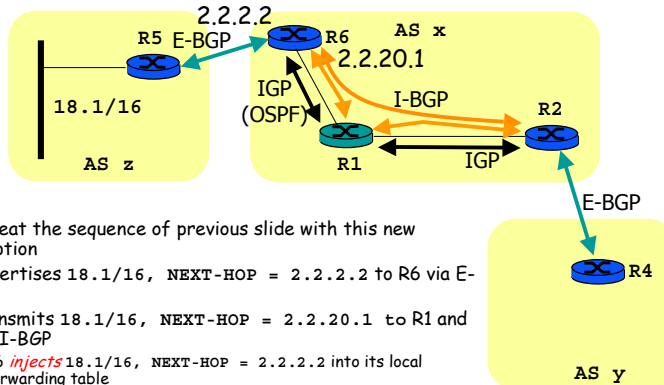
## Avoid Redistribution: Combine Recursive Lookup and NEXT-HOP



- ❑ Data packet to 18.1.2.3 is received by R2
  - Recursive table lookup at R2 can be used
  - Q: there is a problem at R1: how can we solve it ?
  - A: the problem is. Packet is sent by R2 towards R1, which is the neighbour towards 2.2.2.2. At R1, there is no forwarding entry to 18.1/16 - blackhole !  
A solution would be to use loose source routing: R2 adds 2.2.2.2 as loose source routing info into packet. In practice however, source routing is not used with IPv4. See later in the section for another solution.

205

## Avoid Redistribution: Practical Solution

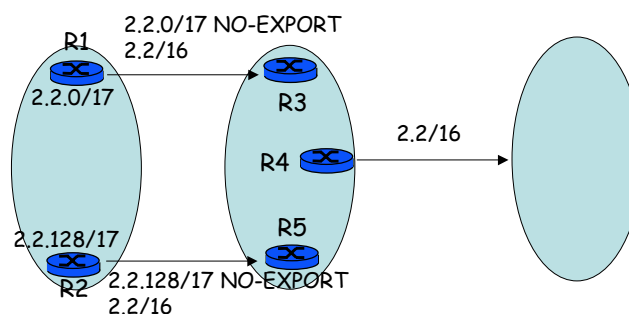


- Q: repeat the sequence of previous slide with this new assumption
- R5 advertises 18.1/16, NEXT-HOP = 2.2.2.2 to R6 via E-BGP
- R6 transmits 18.1/16, NEXT-HOP = 2.2.2.2 to R1 and R2 via I-BGP
  - R6 *injects* 18.1/16, NEXT-HOP = 2.2.2.2 into its local forwarding table
  - R2 *injects* 18.1/16, NEXT-HOP = 2.2.2.2 into its local forwarding table
- Independently, IGP finds that, at R2, packets to 2.2.10.1 should be sent to R1
- Data packet to 18.1.2.3 is received by R2
  - At R2, recursive table lookup determines that packet should be forwarded to R1
  - At R1, recursive table lookup determines that packet should be forwarded to R6
  - At R6, recursive table lookup determines that packet should be forwarded to 2.2.2.2

206

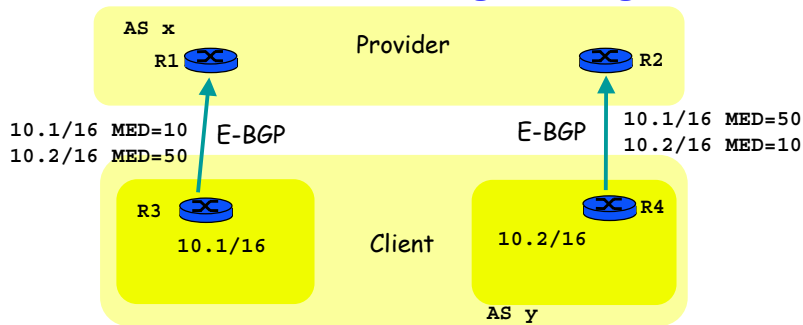
## NO-EXPORT

- Q: What is the route followed by a packet sent to 2.2.48 received by R4 ?
- A: the packet is sent via R3 and R1



207

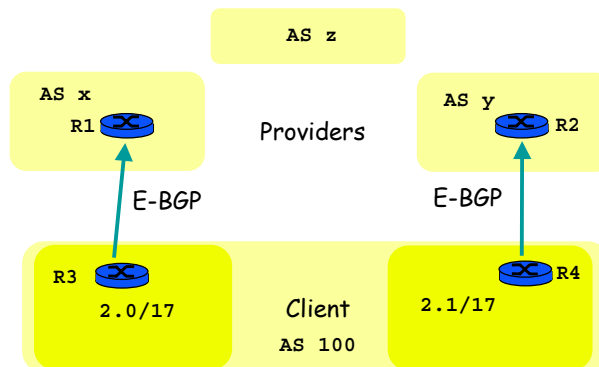
## Ex2: Stub Area, Dual Homing to Single Provider



- Q1: is it possible to avoid distributing BGP routes into Client IGP ?
- A: yes, for example: configure R3 and R4 as default routers in Client AS; traffic from Client AS is forwarded to nearest of R3 and R4. If R3 or R4 fails, to the remaining one
- Q2: is it possible to avoid assigning an AS number to Client ?
- A: Yes, it is sufficient to assign to Client a private AS number: Provider translates this number to its own.
- Q3: is it possible to avoid BGP between Client and Provider ?
- A: Yes, by running a protocol like RIP between Client and Provider and redistributing Client routes into Provider IGP. Thus Provider pretends to the rest of the world that the prefixes of Client are its own.

208

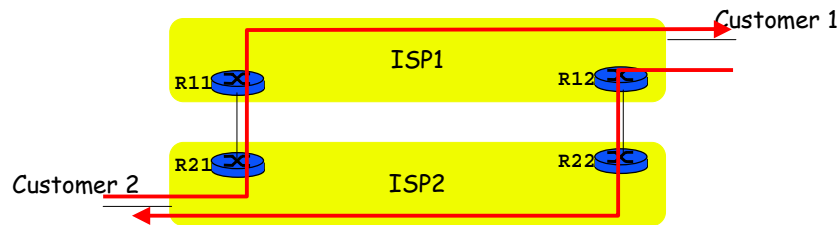
## Ex3: Stub Area, Dual Homing to Several Providers



- Client has own address space and AS number
- Q: how can routes be announced between AS 100 and AS x ? AS x and AS z ?
- A: R3 announces 2.0/17 and 2.0/16; traffic from AS x to 2.0/17 will flow via AS x; if R3 fails, it will use the longer prefix and flow via ASy. ASx announces 2.0/17 and 2.0/16 to AS z
- Q: assume Client wants most traffic to favour AS x. How can that be done ?
- A: R3 announces an artificially inflated path: 100 100 100 100 : 2.0/17. AS z will favour the path via ASy which has a shorter AS path length

209

## Ex4: Hot Potato Routing



- ❑ Packets from Customer 2 to Customer 1
  - Both R21 and R22 have a route to Customer 1
  - Shortest path routing favours R21
  - Q1: by which mechanism is that done ?
  - A: « Choice of the best route » (criteria 5), assuming all routers in ISP2 run BGP
- ❑ Q2: what is the path followed in the reverse direction ?
  - A: see picture. Note the asymmetric routing

210

## Exercise

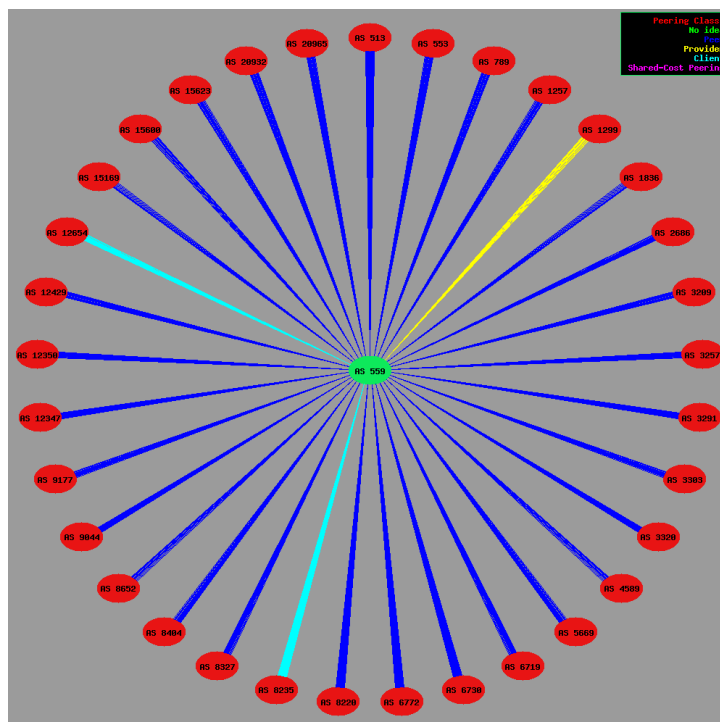
- ❑ What ASs does EPFL receive service from ?
  - from the previous routes, we find AS 559 (Switch)
- ❑ What ASs does Switch receive service from ?
  - from the previous routes we see that there are at least:
    - AS 1299
    - AS 20965
    - AS 3549
- ❑ Find the names of the networks that have these AS numbers
  - from whois on [www.ripe.net](http://www.ripe.net):
    - AS 1299: Telianet
    - AS 20965: Geant
    - AS 3549: Global Crossing

211

## Exercise

- ❑ Lookup <http://rpsl.info.ucl.ac.be>. to find out the relationships between Switch and other providers
- ❑ How does the software on this site decide whether a relationship is client, provider or peer ?
  - AS X is client of Switch if AS X accepts ANY path and announces only self (AS X)
  - AS X is provider of Switch if AS X announces ANY path and accepts only AS Switch
  - AS X is a peer if AS X accepts and announces only a small set of routes

212

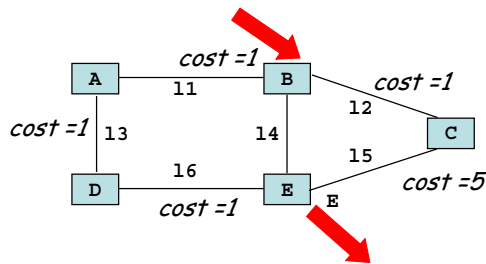


213

## G. Load Dependent Routing

Q1: show an example where shortest path routing does not provide the optimal total flow (where path cost is static)

A: assume all data flow goes from B to E: Static shortest path routing will pick the direct link BE only instead of distributing the load also on some of the longer links (BADE and BCE)



214

## Braess Paradox (1)

□A. there are two paths  
 1: links 1, 3; 2: links 2,4  
 let  $b_i$  be the traffic on path I

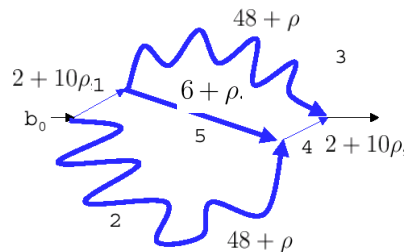
Delay equations:

$$50 + 11b_1 = 50 + 11b_2$$

Total flow

$$b_1 + b_2 = b_0$$

equilibrium is for  $b_1 = b_2$  :  
 delay is 83



215

## Braess Paradox (2)

- Q: same question when we open link 5 with delay function:

$$f_5(\rho) = 6 + \rho.$$

- A: there are three paths  
links 1, 3; 2: links 2,4; 3: links 1, 5, 4

delay equations

$$50 + 11b_1 + 10b_3 = 50 + 11b_2 + 10b_3 = 10 + 10b_1 + 10b_2 + 21b_3$$

total flow

$$b_1 + b_2 + b_3 = b_0$$

We find  $b_1 = b_2 = b_3 = 2 \text{ Gb/s}$

The total delay on all paths is the same, equal to 92 : larger than before!