

Chapter 4 roadmap

4.1 Introduction and Network Service Models

4.2 Routing Principles

- Link state routing
- Distance vector routing

4.3 Hierarchical Routing

4.4 The Internet (IP) Protocol

4.5 Routing in the Internet

4.6 What's Inside a Router

4.7 IPv6

4.8 Multicast Routing

4.9 Mobility

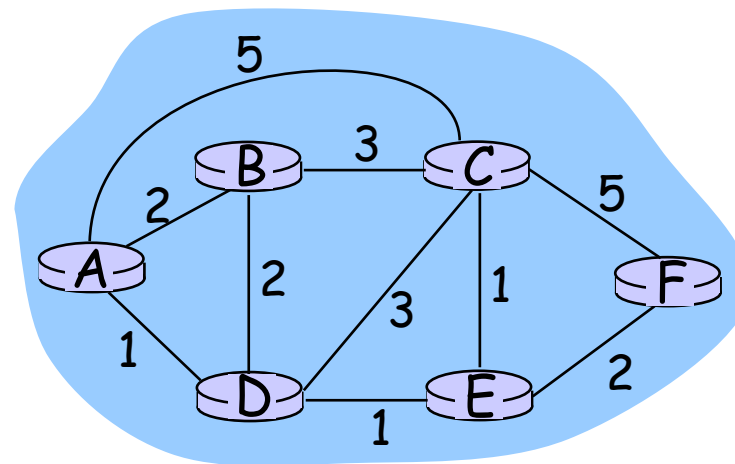
Summary on graphs...(1/3)

A graph $G=(N,A)$ is made of a finite nonempty set of nodes N and a set of edges A . An edge is an unordered pair (i,j) with i and j in N .

A walk in a graph G is a sequence of nodes $(n_1 n_2 \dots n_l)$ such that (n_i, n_{i+1}) is an edge of G . A walk with no repeated nodes is a path. A walk with $n_1 = n_l$ and no repeated nodes is called a cycle.

A graph is connected if for any pair of nodes i,j in N there is a path between them.

A weighted graph is a graph where each edge (i,j) has associated a weight $w_{i,j}$

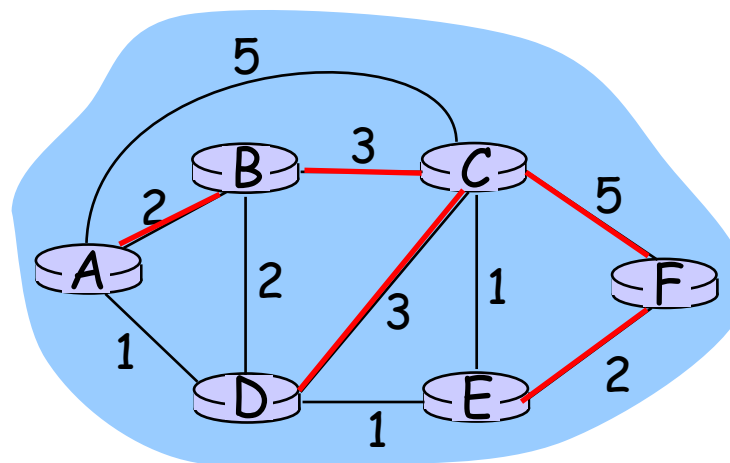


Summary on graphs...(2/3)

Given a graph $G=(N,A)$ we say that a graph $G'=(N',A')$ is a subgraph of G if G' is a graph, $N' \subset N$, $A' \subset A$.

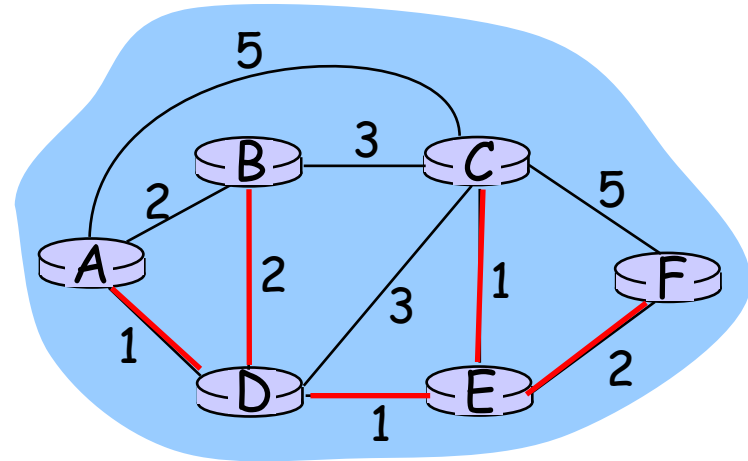
A tree is a connected acyclic graph. A spanning tree of a graph G is a subgraph of G which is a tree and includes all nodes in G .

In a spanning tree there are $|N|-1$ edges, and only one path between each pair of nodes.

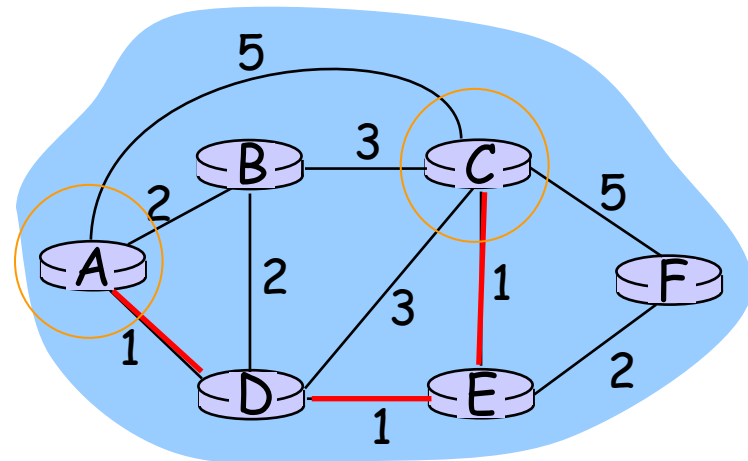


Summary on graphs...(3/3)

A minimum weight spanning tree is a spanning tree with minimum sum of arc weights.



Given two nodes i and j , the shortest path between i and j is a path such that the sum of arc weights is minimum.



Shortest path algorithms: Dijkstra, Bellman-Ford, Floyd-Warshall

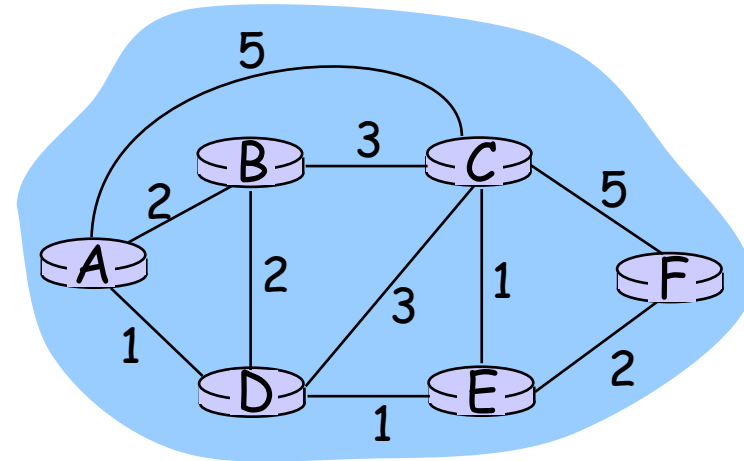
Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
 - link cost: delay, \$ cost, or congestion level



- "good" path:
 - typically means minimum cost path
 - other def's possible

Graph theory is indeed useful!!!

Routing Algorithm classification

Global or decentralized information?

Global:

- ❑ all routers have complete topology, link cost info
- ❑ "link state" algorithms

Decentralized:

- ❑ router knows physically-connected neighbors, link costs to neighbors
- ❑ iterative process of computation, exchange of info with neighbors (which is the cost of the 'best route' from the neighbor?)
- ❑ "distance vector" algorithms

Centralized Shortest Paths Algorithms can be run to compute the routes (—indeed we will see Dijkstra is used in link state algorithms)

Distributed Algorithms based only on local information are needed → distributed Bellman Ford

Routing Algorithm classification

Static or dynamic?

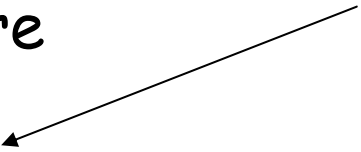
Static:

- ❑ routes change slowly over time, basically manually

Dynamic:

- ❑ routes change more quickly
 - periodic update
 - in response to link cost changes

Distance vector
And link state
Algorithms are dynamic



A Link-State Routing Algorithm

Dijkstra's algorithm

- ❑ net topology, link costs known to all nodes
 - accomplished via "link state broadcast"
 - all nodes have same info
- ❑ computes least cost paths from one node ('source') to all other nodes
 - gives routing table for that node
- ❑ iterative: after k iterations, know least cost path to k dest.'s

Notation:

- ❑ $c(i,j)$: link cost from node i to j . cost infinite if not direct neighbors
- ❑ $D(v)$: current value of cost of path from source to dest. v
- ❑ $p(v)$: predecessor node along path from source to v , that is next v
- ❑ N : set of nodes whose least cost path definitively known

Algorithm assumption: No negative weights!! Network Layer 4-9

Dijkstra's Algorithm

1 **Initialization:**

2 $N = \{A\}$

3 for all nodes v

4 if v adjacent to A

5 then $D(v) = c(A, v)$

6 else $D(v) = \text{infinity}$

7

8 **Loop**

9 find w not in N such that $D(w)$ is a minimum

10 add w to N

11 update $D(v)$ for all v adjacent to w and not in N :

12 $D(v) = \min(D(v), D(w) + c(w, v))$

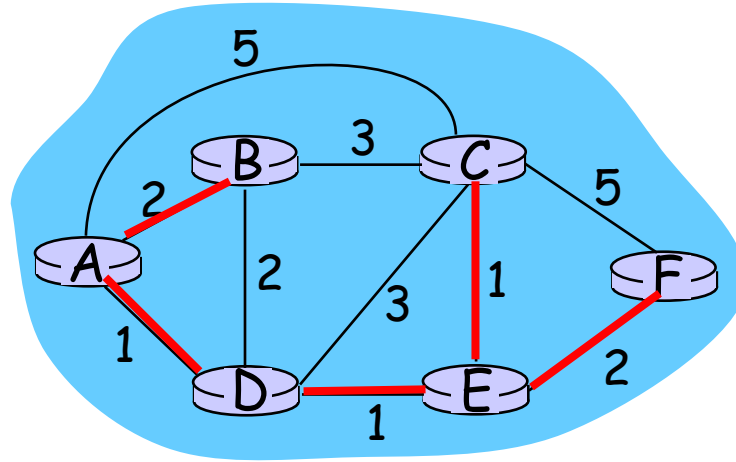
13 /* new cost to v is either old cost to v or known

14 shortest path cost to w plus cost from w to v */

15 **until all nodes in N**

Dijkstra's algorithm: example (shortest paths from A to other nodes)

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	A	2,A	5,A	1,A	infinity	infinity
→ 1	AD	2,A	4,D		2,D	infinity
→ 2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E
→ 4	ADEBC					4,E
5	ADEBCF					



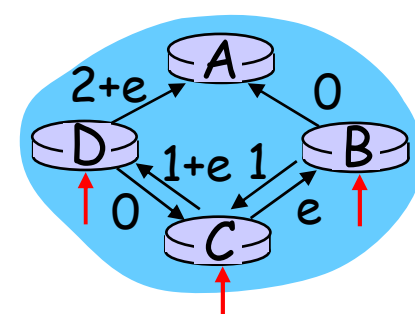
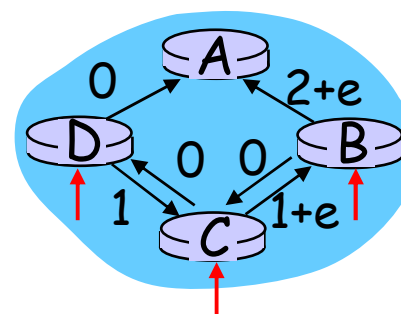
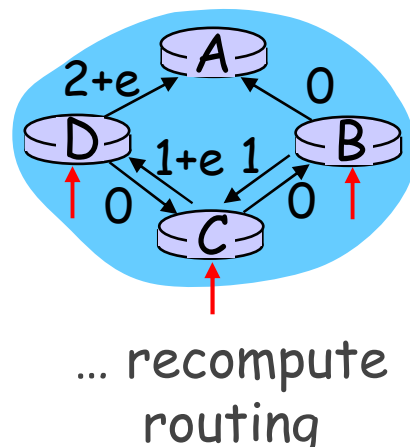
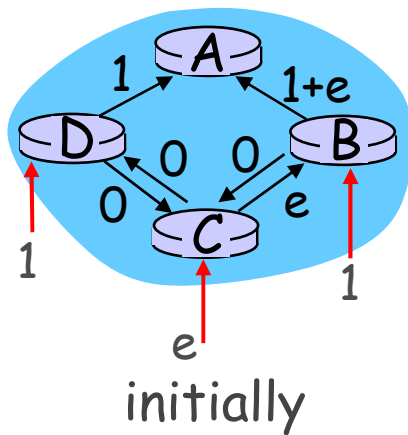
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w , not in N
- $n*(n+1)/2$ comparisons: $O(n^2)$

Oscillations possible:

- e.g., link cost = amount of carried traffic



Usually metric adopted in routing simply the number of hops

Dijkstra's Algorithm (complexity)

1 **Initialization:**

2 $N = \{A\}$

3 for all nodes v

4 if v adjacent to A

5 then $D(v) = c(A, v)$

6 else $D(v) = \text{infinity}$

7

$O(n)$

8 **Loop**

9 find w not in N such that $D(w)$ is a minimum

10 add w to N

11 update $D(v)$ for all v adjacent to w and not in N :

12 $D(v) = \min(D(v), D(w) + c(w, v))$

13 /* new cost to v is either old cost to v or known

14 shortest path cost to w plus cost from w to v */

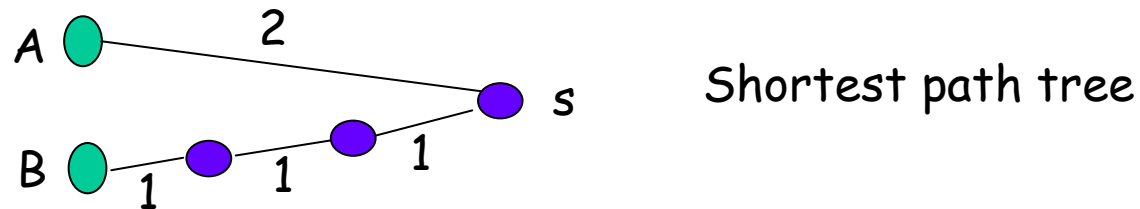
15 **until all nodes in N**

$O(\text{number of nodes not in } N)$
 $n, n-1, n-2, \dots$ at the different iterations

n

Shortest paths vs. Min. weight Spanning tree

- The shortest paths form a (spanning) tree
- Is it a minimum weight spanning tree? NO.
 - Example:



Why not a shortest path tree? s -A has length 3 instead of 2!!
But the overall sum of arc weights is 4 vs. 4 in the shortest path tree

Bellman-Ford

Given a graph $G=(N,A)$ and a node s finds the shortest path from s to every node in N .

A shortest walk from s to i subject to the constraint that the walk contains at most h arcs and goes through node s only once, is denoted $\text{shortest}(\leq h)$ walk and its length is D^h_i .

Bellman-Ford rule:

Initialization $D^h_s=0$, for all h ; $w_{i,k} = \text{infinity}$ if (i,k) NOT in A ; $w_{k,k}=0$;
 $D^0_i=\text{infinity}$ for all $i \neq s$.

Iteration:

$$D^{h+1}_i = \min_k [w_{i,k} + D^h_k]$$

Assumption: non negative cycles (this is the case in a network!!)

The Bellman-Ford algorithm first finds the one-arc shortest walk lengths, then the two-arc shortest walk length, then the three-arc...etc. \rightarrow distributed version used for routing

Bellman-Ford

$$D^{h+1}_i = \min_k [w_{i,k} + D^h_k]$$

Can be computed locally.

What do I need?

For each neighbor k , I need to know

- the cost of the link to it (known info)

- The cost of the best route from the neighbor k to the destination
(←this is an info that each of my neighbor has to send to me via messages)

In the real world: I need to know the best routes among each pair of nodes → we apply distributed Bellman Ford to get the best route for each of the possible destinations

Distance Vector Routing Algorithm

-Distributed Bellman Ford

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info/iterate in lock step!

Distributed, based on local info:

- each node communicates *only* with directly-attached neighbors

Distance Table data structure

each node has its own

- row for each possible destination
- column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

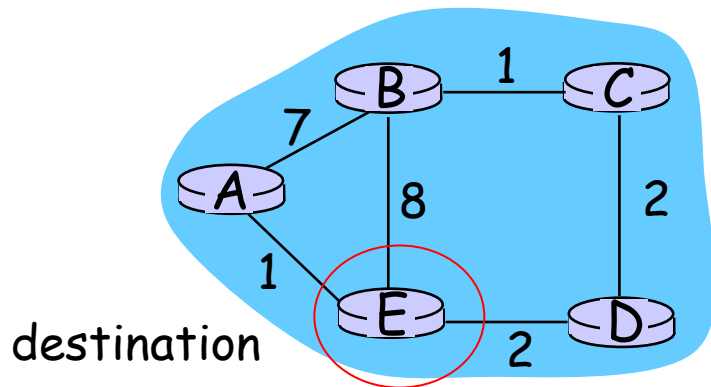
Cost associated to the (X,Z) link

$$D^X(Y,Z) = \text{distance from X to Y, via Z as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Info maintained at Z. Min must be communicated

Distance Table: example



$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\}$$

$$= 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\}$$

$$= 2+3 = 5$$

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\}$$

$$= 8+6 = 14$$

Path B-C-D-E-A

Distance table in node E after the algorithm has converged

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

First example

Distance table gives routing table

		cost to destination via		
destination	$D^E()$	A	B	D
	A	1	14	5
	B	7	8	5
	C	6	9	4
	D	4	11	2

		Outgoing link to use, cost
destination	A	A,1
	B	D,5
	C	D,4
	D	D,2

Distance table \longrightarrow Routing table

Distance Vector Routing: overview

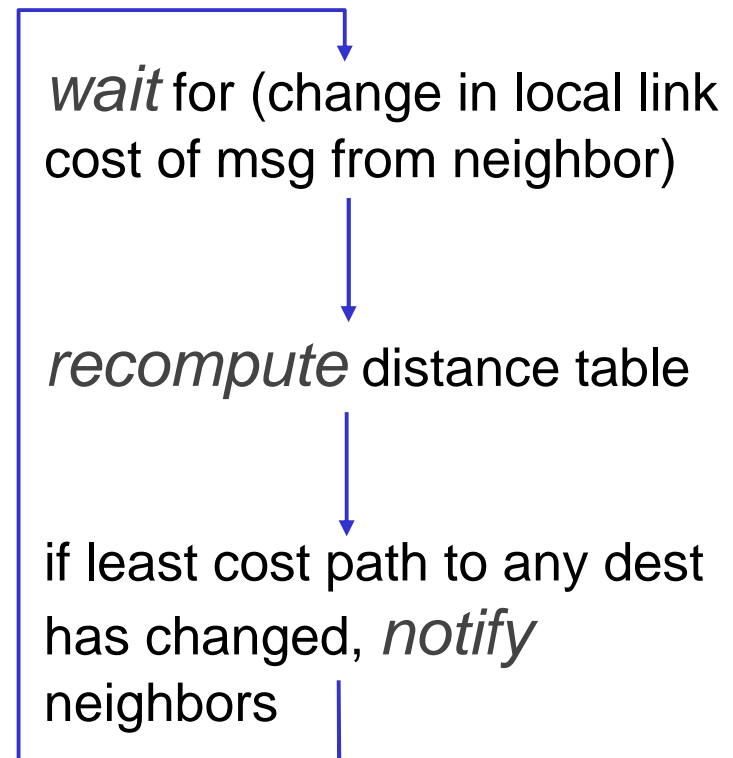
Iterative, asynchronous:
each local iteration caused
by:

- ❑ local link cost change
- ❑ message from neighbor: its
least cost path change
from neighbor

Distributed:

- ❑ each node notifies
neighbors *only* when its
least cost path to any
destination changes
 - neighbors then notify
their neighbors if
necessary

Each node:



Distance Vector Algorithm:

At all nodes, X:

- 1 Initialization:
- 2 for all adjacent nodes v:
- 3 $D^X(*,v) = \text{infinity}$ /* the * operator means "for all rows" */
- 4 $D^X(v,v) = c(X,v)$
- 5 for all destinations, y
- 6 send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

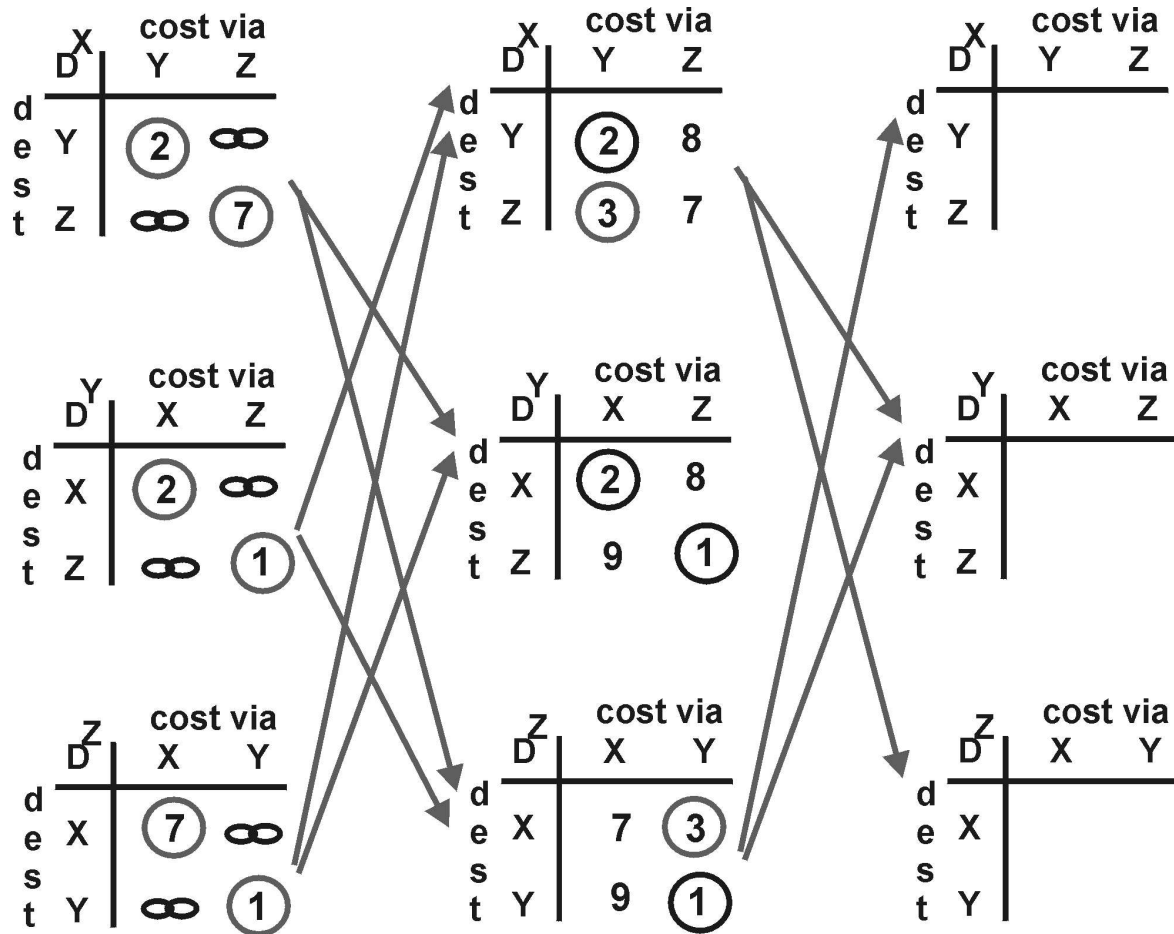
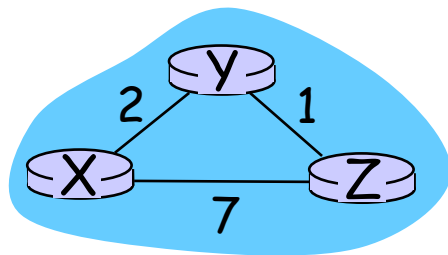


From the node to whatever destination going through v

Distance Vector Algorithm (cont.):

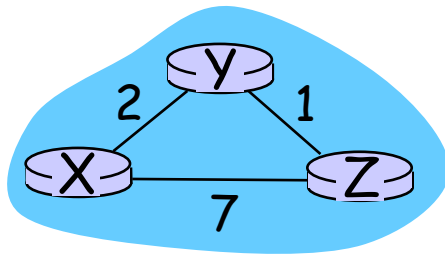
```
8 loop
9   wait (until I see a link cost change to neighbor V
10      or until I receive update from neighbor V)
11
12   if (c(X,V) changes by d)
13     /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15     for all destinations y:  $D^X(y,V) = D^X(y,V) + d$ 
16
17   else if (update received from V wrt destination Y)
18     /* shortest path from V to some Y has changed */
19     /* V has sent a new value for its  $\min_w DV(Y,w)$  */
20     /* call this received new value is "newval" */
21     for the single destination y:  $D^X(Y,V) = c(X,V) + \text{newval}$ 
22
23   if we have a new  $\min_w D^X(Y,w)$  for any destination Y
24     send new value of  $\min_w D^X(Y,w)$  to all neighbors
25
26 forever
```

Distance Vector Algorithm: example



Cost updates from the neighbors are used for sake of recomputing
The best routes and may lead to new cost updates...

Distance Vector Algorithm: example



		cost via	
		Y	Z
d e s t	X	2	∞
	Z	∞	7

		cost via	
		X	Z
d e s t	Y	2	∞
	Z	∞	1

		cost via	
		X	Y
d e s t	Z	7	∞
	Y	∞	1

		cost via	
		Y	Z
d e s t	X	2	8
	Z	3	7

Line 21 of the algorithm description

$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$$

$$= 7 + 1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\}$$

$$= 2 + 1 = 3$$

Distributed Bellman Ford correctness

- ❑ Completely asynchronous
- ❑ Starting from arbitrary estimates of the cost of the 'best route' from node i to the destination, if:
 - links weights are constant for enough time for the protocol to converge
 - stale info expire after a while
 - once in a while updated info are sent from a node to its neighbors

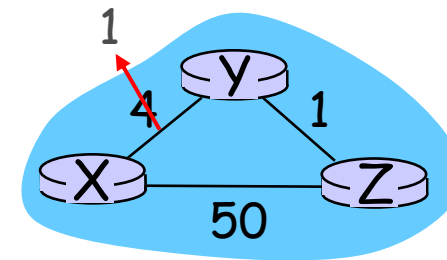
the Distributed Bellman Ford algorithm converges,
i.e. each node correctly estimates the cost of the
best route to the destination

Distance Vector: link cost changes

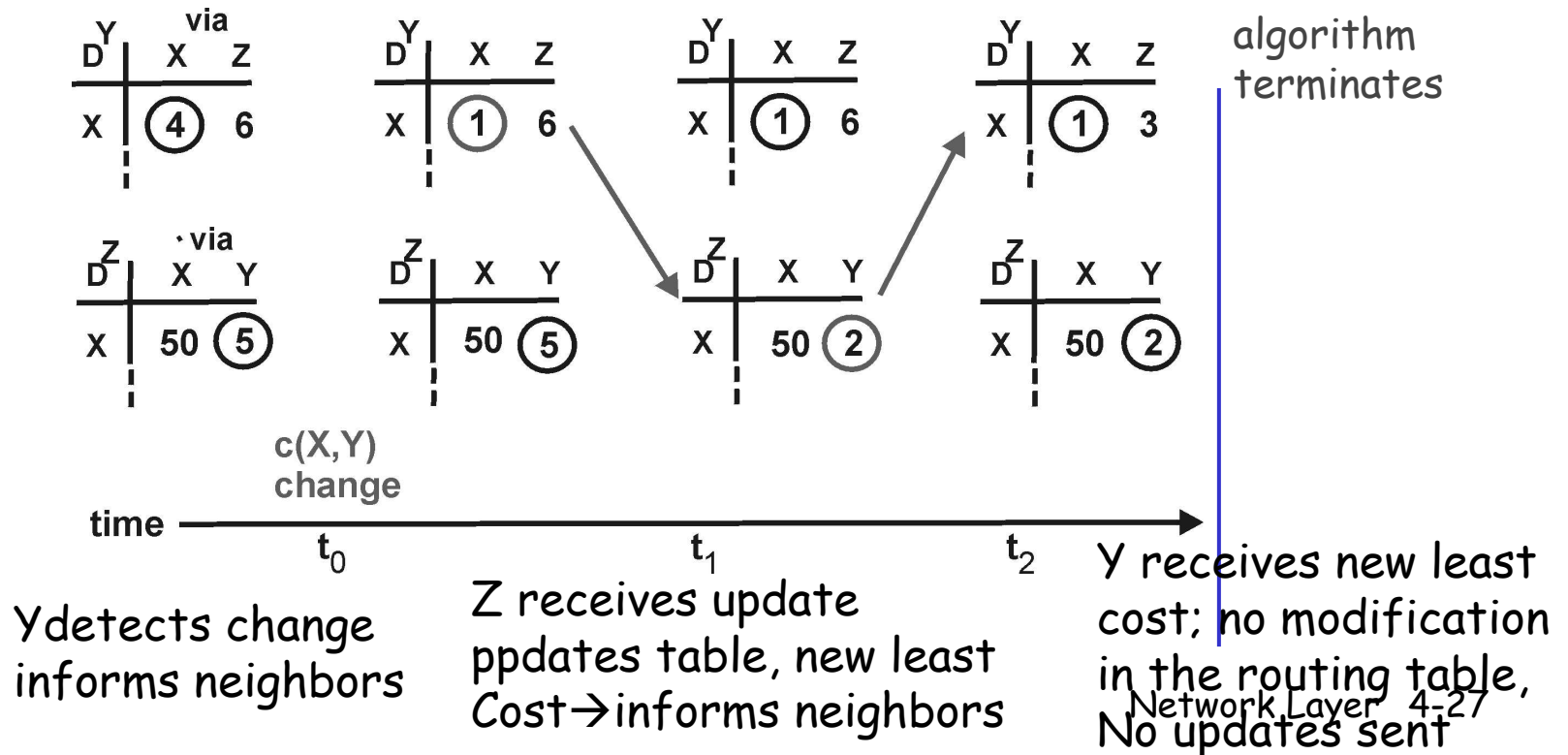
Subtitle: Distributed Bellman Ford converges but how fast?

Link cost changes:

- ❑ node detects local link cost change
- ❑ updates distance table (line 15)
- ❑ if cost change in least cost path, notify neighbors (lines 23,24)



"good news travels fast"



Previous lecture. Summary:

Distributed Belman Ford

- Based on Distributed Bellman Ford Equation

Cost associated to the (X,Z) link

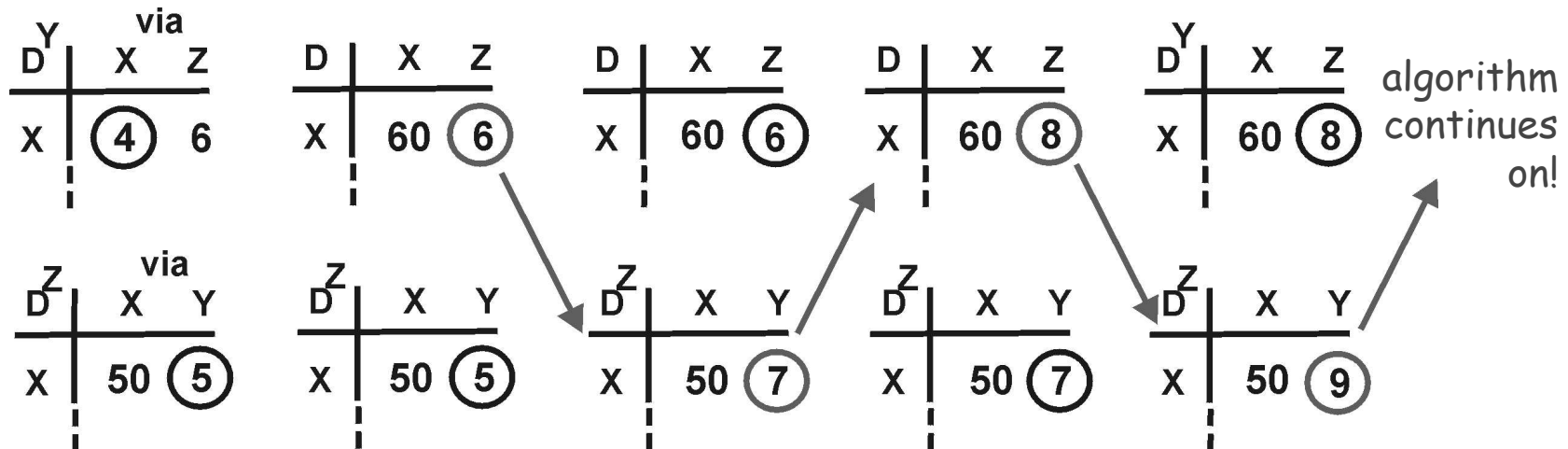
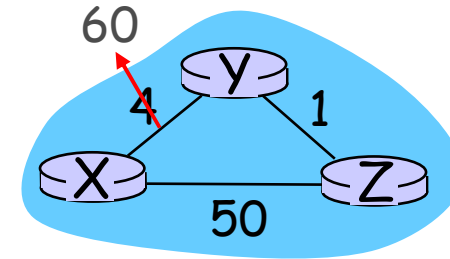
$$D^X(Y,Z) = \text{distance from X to Y, via Z as next hop}$$
$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

- $D^X(Y,Z)$ recomputed:
 - Upon reception of updates from the neighbors
 - Upon link cost change
- $\min_z D^X(Y,Z)$ communicated to the neighbors whenever its value changes, or periodically
- How long does it take for the algorithm to converge? ‘good news travel fast, bad news may not → count to infinity’

Distance Vector: link cost changes

Link cost changes:

- ❑ good news travels fast
- ❑ bad news travels slow - "count to infinity" problem!



Y detects link cost
Increase but think can
Reach X through Z at a
total cost of 6 (wrong!!)

The path is Y-Z-Y-X

Count-to-infinity -an everyday life example

Which is the problem here?

the info exchanged by the protocol!! ‘the best route to X I have has the following cost...’ (no additional info on the route)

A Roman example...

-assumption: there is only one route going from Colosseo to Altare della Patria: Via dei Fori Imperiali. Let us now consider a network, whose nodes are Colosseo., Altare della Patria, Piazza del Popolo



Count-to-infinity –everyday life example (2/2)



The Colosseo. and Alt. Patria nodes exchange the following info

- Colosseo says ‘the shortest route from me to P. Popolo is 2 Km’
- Alt. Patria says ‘the shortest path from me to P. Popolo is 1Km’

Based on this exchange from Colosseo you go to Al. Patria, and from there to Piazza del Popolo OK Now due to the big dig they close Via del Corso (Al. Patria—P.Popolo)

• Al. Patria thinks ‘I have to find another route from me to P.Popolo. Look there is a route from Colosseo to P.Popolo that takes 2Km, I can be at Colosseo in 1Km → I have found

a 3Km route from me to P.Popolo!!’ Communicates the new cost to

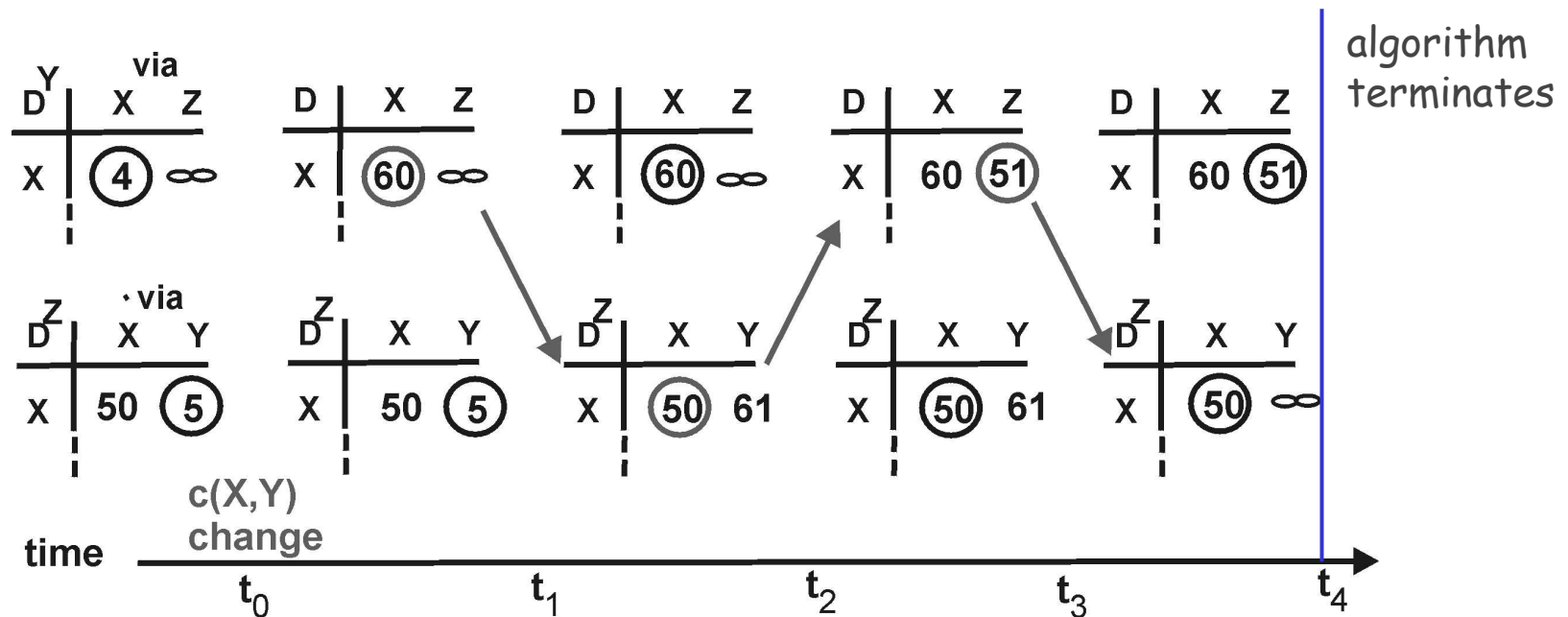
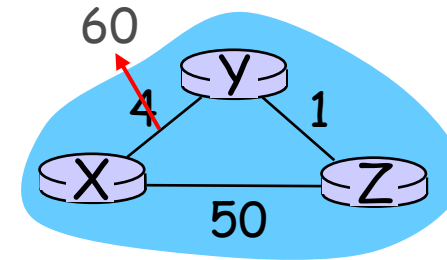
Colosseo that updates ‘OK I can go to P.Popolo via Al. Patria in 4Km’

VERY WRONG!! Why is it so? I didn’t know that the route from Colosseo to P.Popolo was going through Via del Corso from Al.Patria to P.Popolo (which is closed)!!

Distance Vector: poisoned reverse

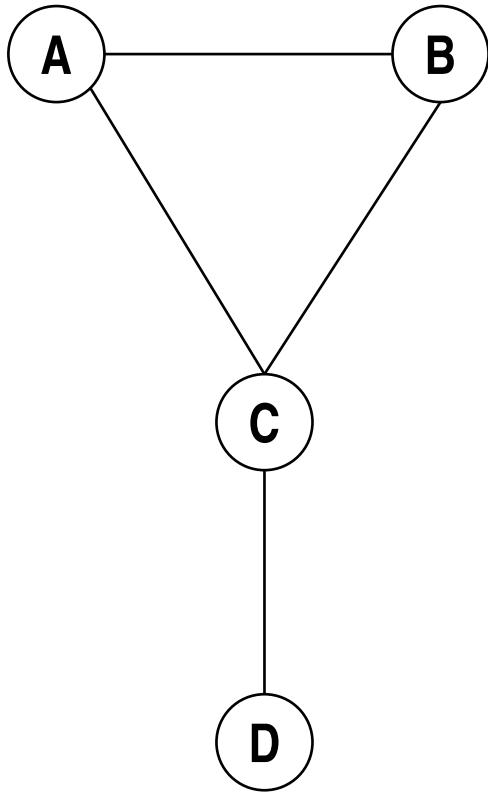
If Z routes through Y to get to X :

- ❑ Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- ❑ will this completely solve count to infinity problem?



Infinity is advertized by Y
(poisoned reverse)

Split horizon poison reverse failure



Line CD goes down...

- 1) because of split horizon rule,
A and B tell C that $\text{dist}(D)=\text{inf}$
- 2) C concludes that D is unreachable
and reports this to A and B
- 3) but A knows from B that $\text{dist}(D)=2$, and
sets its $\text{dist}=3$
- 4) similarly, B knows from A distance from D...
C estimates new value 4; A and B again through C
estimate a value of 5....then again 1)
... etc until distance = infinite

*Regardless the hack used, there is always a network topology
that makes the trick fail!*

Routing in the Internet

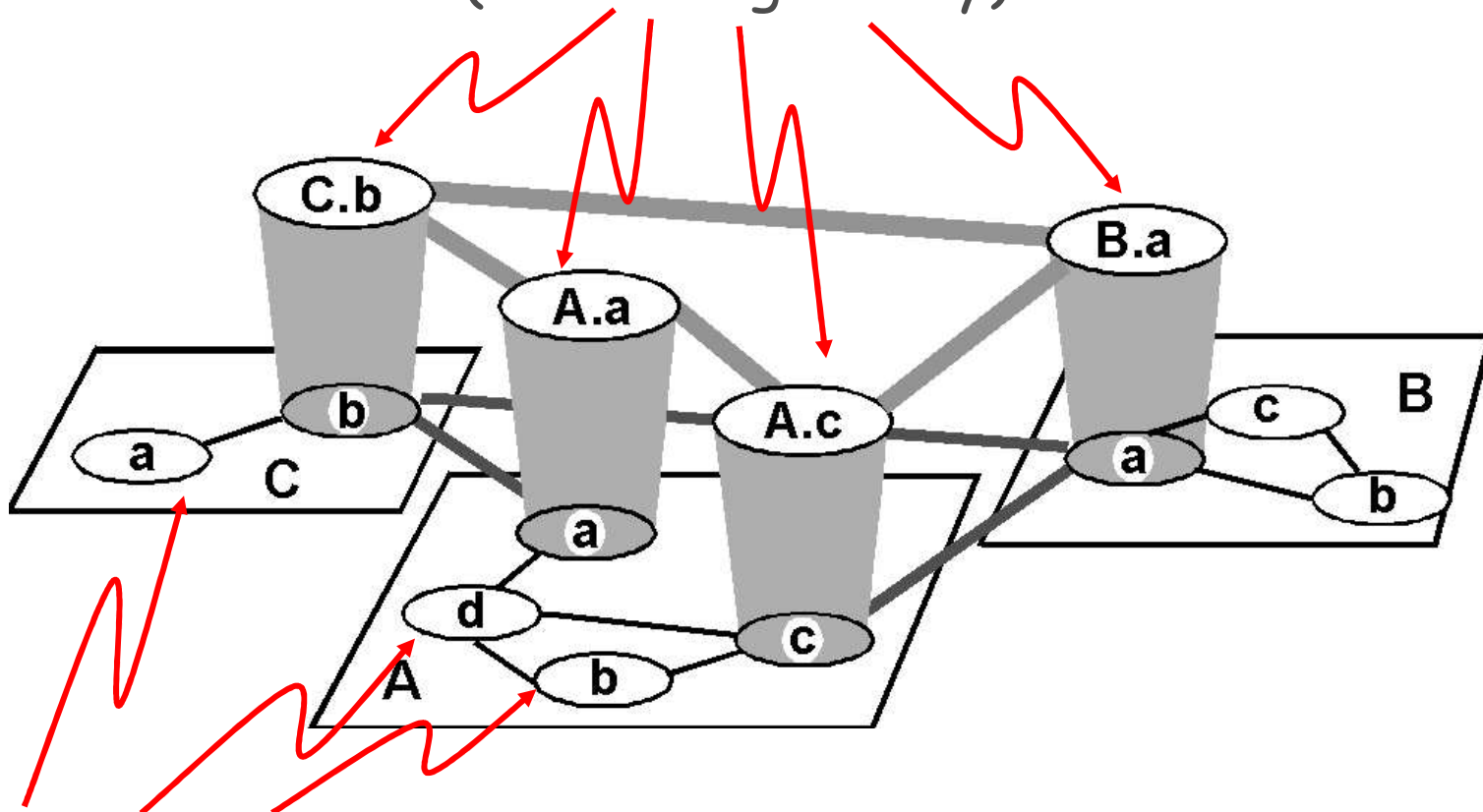
- ❑ The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
 - **Stub AS**: small corporation: one connection to other AS's
 - **Multihomed AS**: large corporation (no transit): multiple connections to other AS's
 - **Transit AS**: provider, hooking many AS's together

- ❑ Two-level routing:
 - **Intra-AS**: administrator responsible for choice of routing algorithm within network
 - **Inter-AS**: unique standard for inter-AS routing: BGP

Autonomous system: administered (or at least some degree of technical and administrative control) by a single entity, characterized by a given Routing technology.

Internet AS Hierarchy

Inter-AS border (exterior gateway) routers



Intra-AS interior (gateway) routers

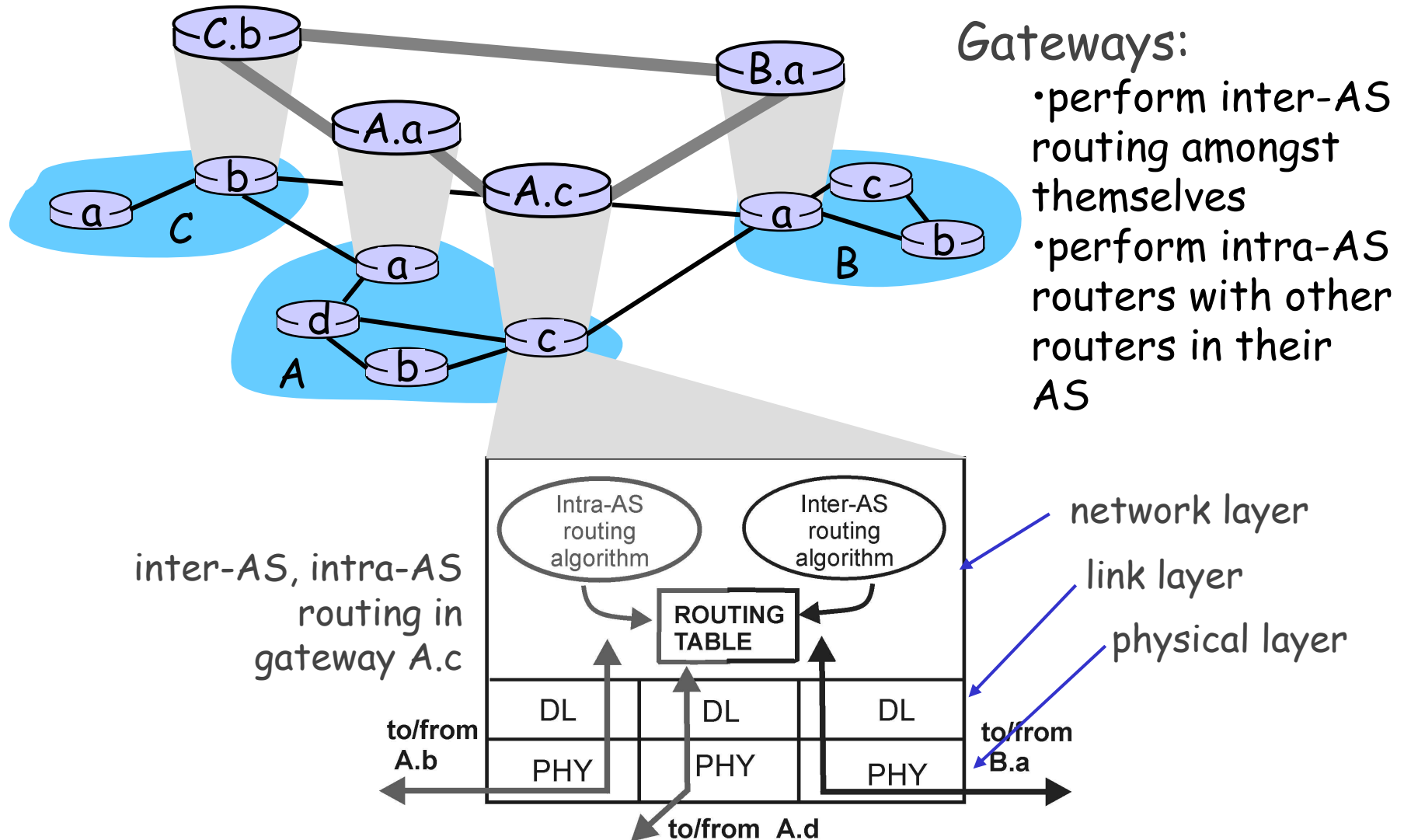
Hierarchical Routing

- ❑ aggregate routers into regions, “autonomous systems” (AS)
- ❑ routers in same AS run same routing protocol
 - “intra-AS” routing protocol
 - routers in different AS can run different intra-AS routing protocol

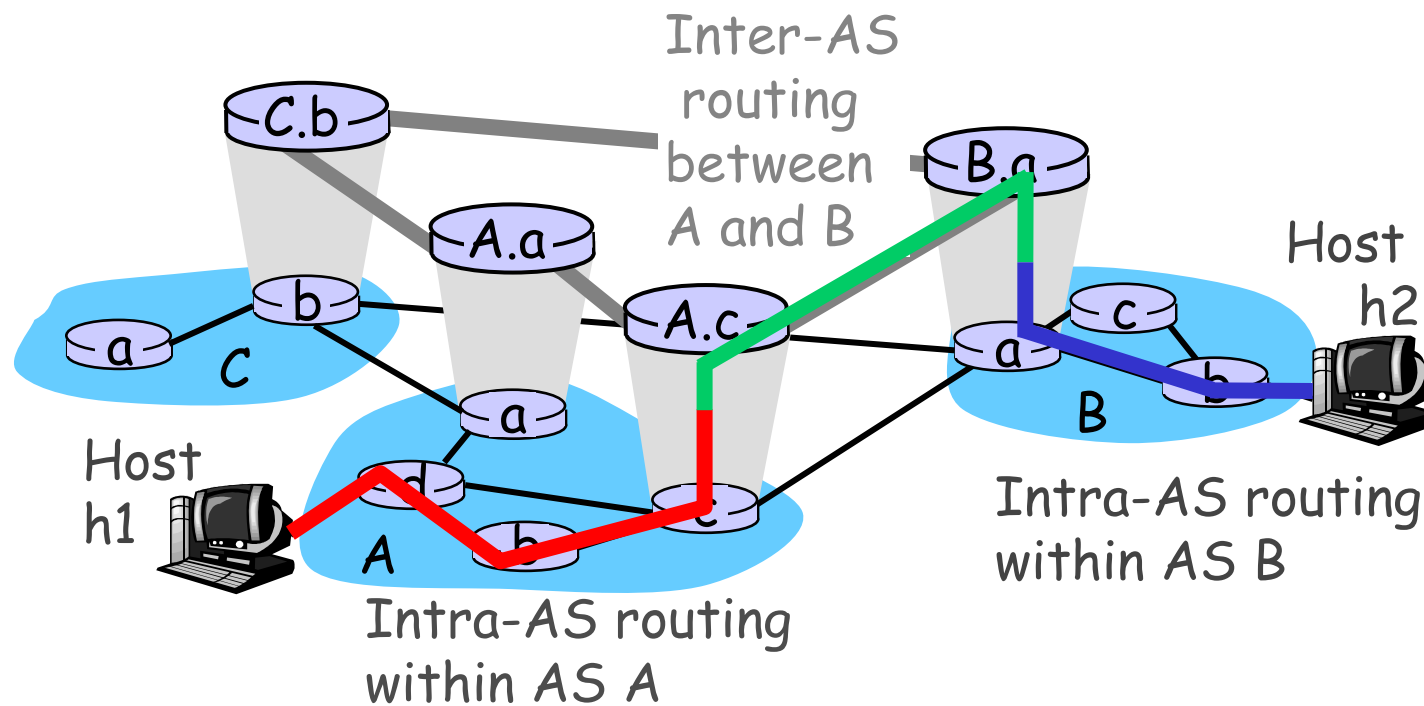
gateway routers

- ❑ special routers in AS
- ❑ run intra-AS routing protocol with all other routers in AS
- ❑ *also* responsible for routing to destinations outside AS
 - run *inter-AS routing* protocol with other gateway routers

Intra-AS and Inter-AS routing



Intra-AS and Inter-AS routing



- We'll examine specific inter-AS and intra-AS Internet routing protocols shortly

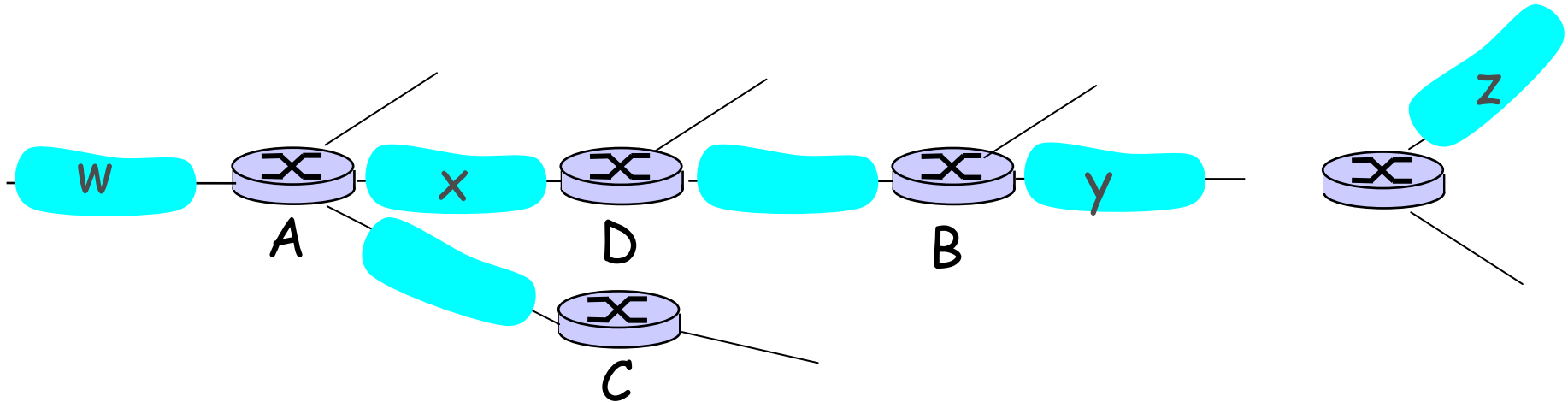
Intra-AS Routing

- ❑ Also known as Interior Gateway Protocols (IGP)
- ❑ Most common Intra-AS routing protocols:
 - RIP: Routing Information Protocol
 - basically implements the distributed Bellman Ford Algorithm
 - Distance metric: # of hops (max = 15 hops)
 - *Can you guess why? (why limit on the num. of hops, why #hops as a metric?)*
 - OSPF: Open Shortest Path First (link state protocol, uses Dijkstra, more later)
 - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

- ❑ Distance vector algorithm
- ❑ Included in BSD-UNIX Distribution in 1982
- ❑ designed for IGP, moderate size networks
- ❑ Distance metric: # of hops (max = 15 hops)
 - *Can you guess why?*
- ❑ Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called **advertisement**)
- ❑ Each advertisement: list of up to 25 destination nets within AS

RIP: Example



Destination Network	Next Router	Num. of hops to dest.
W	A	2
Y	B	2
Z	B	7
X	--	1
....

Routing table in D

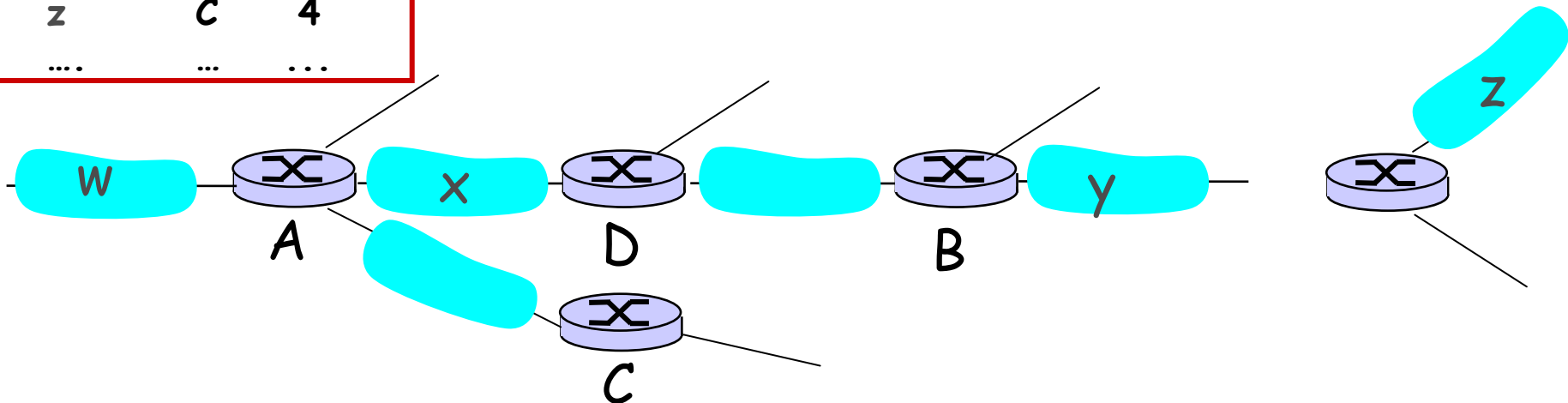
Info: dest. Address, next router, interface to the next router, metric value
 Timer(amount of time since last update of the entry)

RIP: Example

1) takes track only of the 'best route'

Dest	Next	hops
w	-	-
x	-	-
z	C	4
...

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B A	7 5
x	--	1
....

Routing table in D

RIP logic

- ❑ *Distance Vector routing protocol*
 - Bellman-Ford algorithm
 - METRIC = distance
 - STATE INFO = vector
- ❑ each router maintains a table with:
 - best known distance (in hop count) to each destination
 - link to use to reach the destination
- ❑ fully distributed protocol
 - vector (table) updates via periodic (30 s) communication with neighbors
 - when an update arrives from neighbors G' compare if cost of network shared with G' + advertised cost smaller than what stored currently in the table. In case update the table. Also update if G' is the router through which the current route in the table goes through or if there is no entry in the table.
- ❑ What if the next-hop router crashes? Timeouts → when it expires and no updates is received from the router, the route is deleted from the table.

RIP: Link Failure and Recovery

If no advertisement heard after 180 sec -->
neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

Real limit of RIP

- ❑ Hop count is a too simple metric!
 - But Bellman-ford algorithm does not require to operate with hop count! Can be trivially extended to different distance metric: the core of the algorithm does not change!
 - queue length on considered hop
 - time delay in ms
 - packet loss ratio measured
 - etc...
- ❑ Slow convergence
 - routers distant N hops need N steps to update their tables
- ❑ Limited to small network sizes
- ❑ Insane transient reaction to node/link failures!
 - Convergence still remains, but very slow
 - Loops may occur while routing tables stabilize
 - the slower, the higher value infinite is chosen!!
 - Values higher than 16 are terrible
 - An intrinsic and unavoidable drawback for all Distance Vector schemes

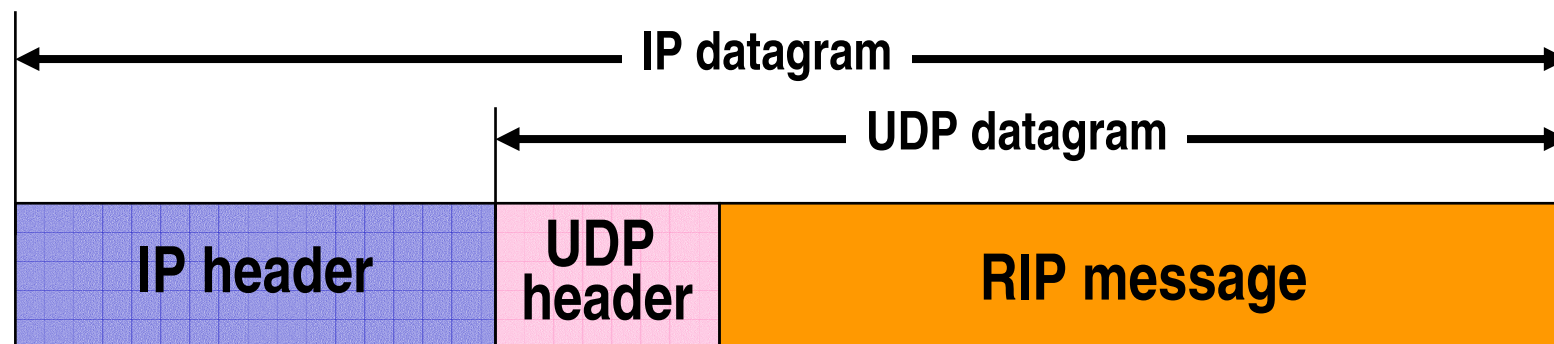
RIP Solution: trigger updates

- **Just an attempt to speed up convergence**
- **Whenever a gateway change the metric for a route it is required to send an update message almost immediately (even if it is not time for the Regular periodic update message)**
- **→ In case of (and only if) changes modifying the best route or the best route metric value updates are sent → indeed trigger updates lead to a cacade of updates → fast convergence**

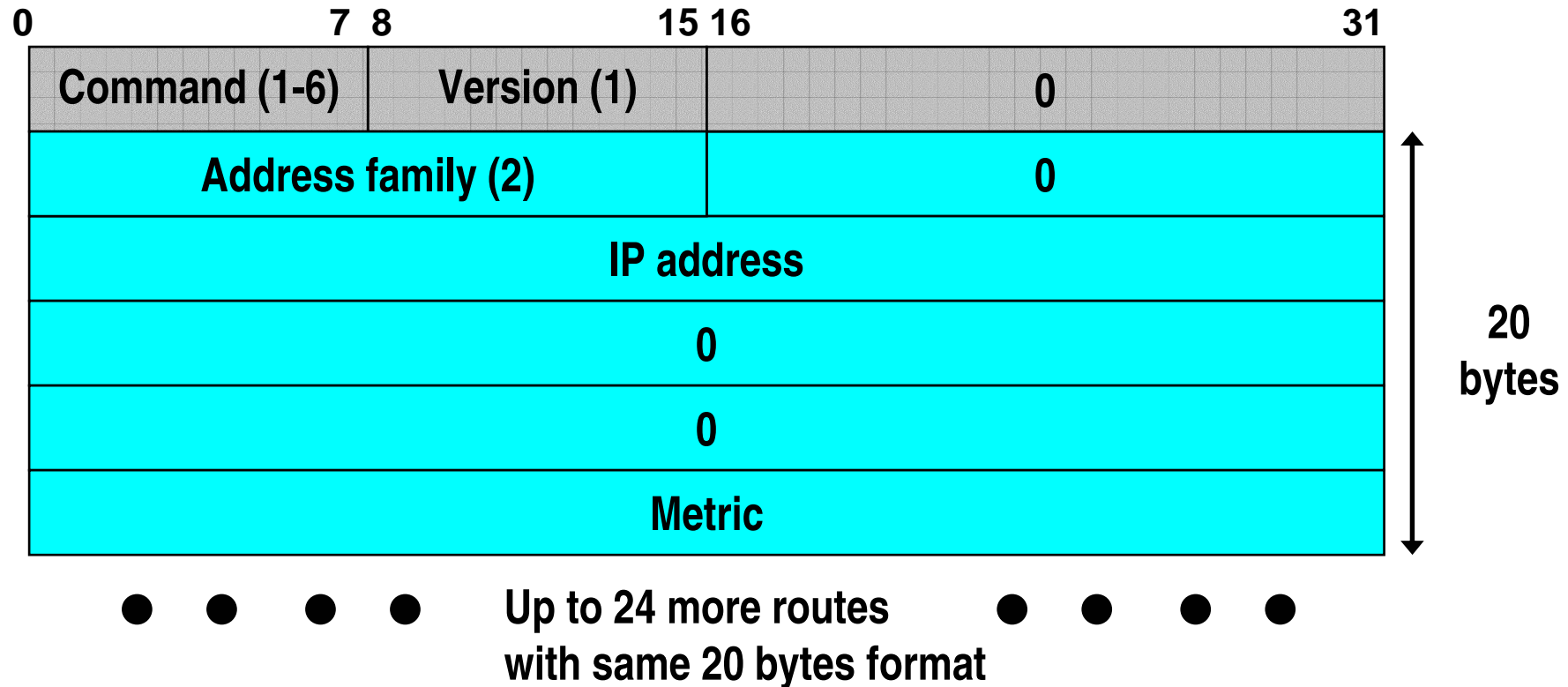
Routing Information Protocol

- ❑ Most widely used
 - and most criticized...
- ❑ Official specification: RFC 1058 (1988)
 - but used from several years before
- ❑ Uses UDP to exchange messages
 - well known UDP port = 520

NOTE: IT IS ON
THE CONTROL PLANE



RIP message



Command: 1=request to send all or part of the routing table; 2=reply (3-6 obsolete or non documented)

Address family: 2=IP addresses

metric: distance of *emitting router* from the specified IP address in *number of hops* (valid from 1 to 15; 16=infinite)

Message size

- 8 UDP header
- 4 bytes RIP header
- 20 bytes x up to 25 entries
- ❑ total: maximum of 512 bytes UDP datagram

- ❑ 25 entries: too little to transfer an entire routing table
 - more than 1 UDP datagram generally needed

Initialization

- ❑ When routing daemon started, send special RIP request on every interface
 - command = 1 (request)
 - address family = 0 (instead of 2)
 - metric set to 16 (infinite)
- ❑ This asks for complete routing table from all connected routers
 - allows to discover adjacent routers!

Operation after initialization

- ❑ Request:
 - asks for response relative to specific IP addresses listed in the request message
- ❑ Response:
 - return list of IP addresses with associated metric
 - if router does not have a route to the specified destination, returns 16
- ❑ Regular update:
 - routers send their table every 30s to adjacent routers
 - a router deletes (set metric to 16) an entry from its routing table if not refreshed within 6 cycles (180s)
- ❑ triggered update:
 - upon change of metric for a route (transmits only entries changed)

RIP Table example (continued)

Router: *girofflee.eurocom.fr*

Destination	Gateway	Flags	Ref	Use	Interface
-----	-----	-----	-----	-----	-----
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

- ❑ Three attached class C networks (LANs)
- ❑ Router only knows routes to attached LANs
- ❑ Default router used to "go up"
- ❑ Route multicast address: 224.0.0.0
- ❑ Loopback interface (for debugging)

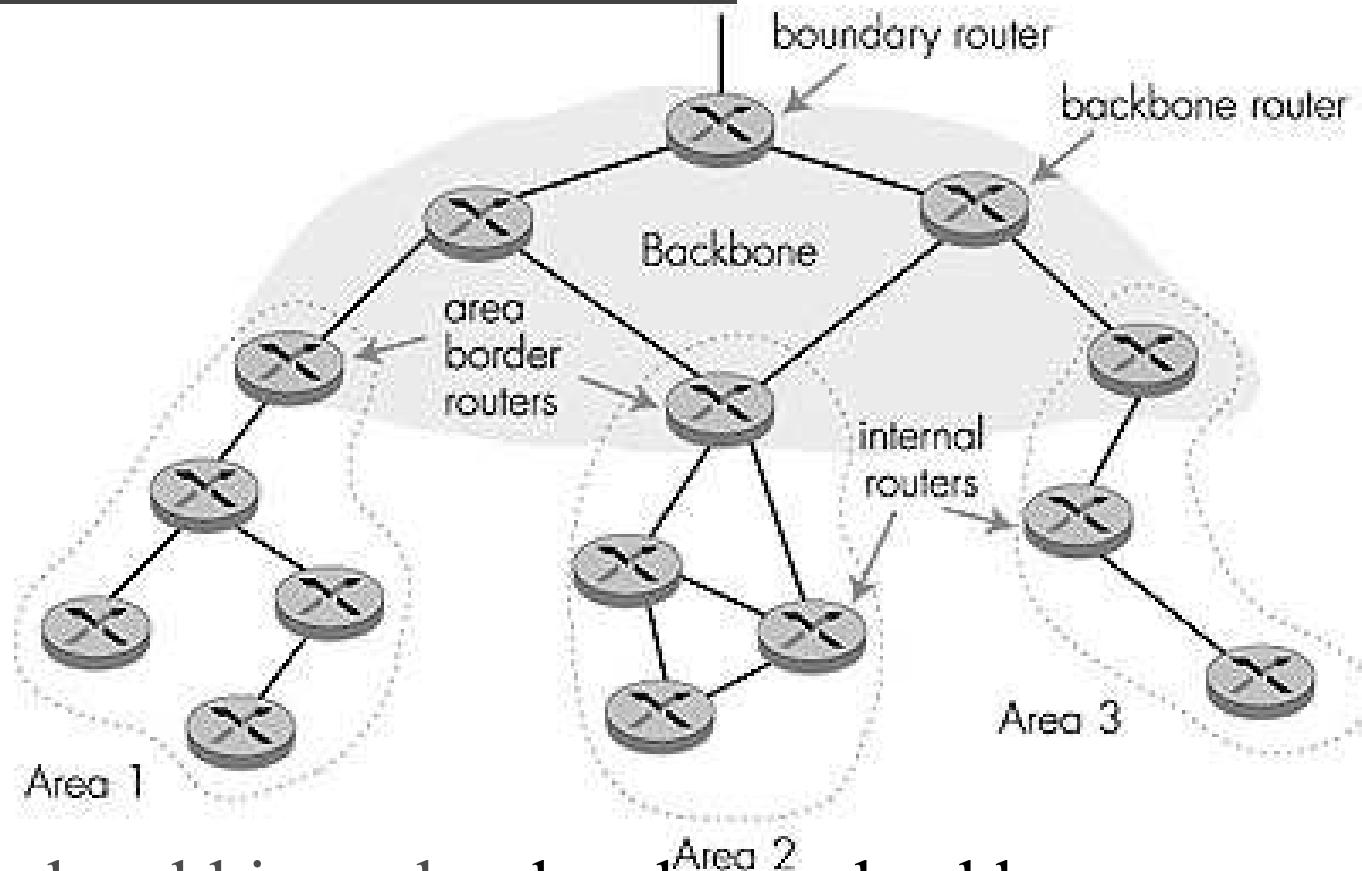
OSPF (Open Shortest Path First)

- ❑ “open”: publicly available
 - ❑ Uses Link State algorithm
 - LS packet dissemination
 - Topology map at each node
 - Route computation using Dijkstra's algorithm
 - ❑ OSPF advertisement carries one entry per neighbor router
 - ❑ Advertisements disseminated to entire AS (via flooding)
 - Carried in OSPF messages directly over IP
- Network Layer 4-53

OSPF "advanced" features (not in RIP)

- ❑ Security: all OSPF messages authenticated (to prevent malicious intrusion)
- ❑ Multiple same-cost paths allowed (only one path in RIP)
- ❑ Load balancing
- ❑ For each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort; high for real time)
- ❑ Integrated uni- and multicast support:
 - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- ❑ Hierarchical OSPF in large domains.

Hierarchical OSPF



- Two-level hierarchy: local area, backbone.

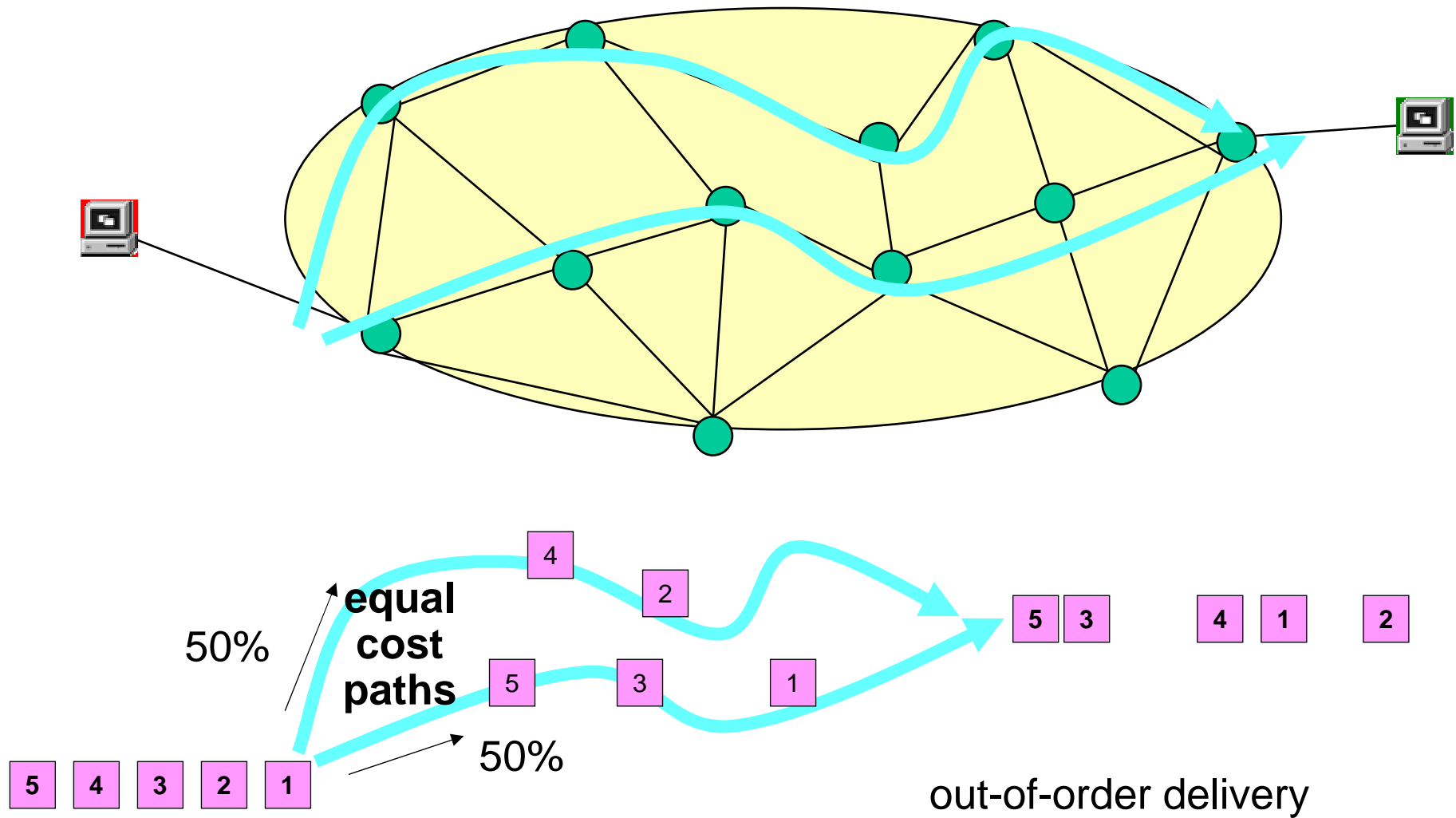
- Link-state advertisements only in area

- each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.

Hierarchical OSPF

- ❑ Two-level hierarchy: local area, backbone.
 - Link-state advertisements only in area
 - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- ❑ **Area border routers:** “summarize” distances to nets in own area, advertise to other Area Border routers.
- ❑ **Backbone routers:** run OSPF routing limited to backbone.
- ❑ **Boundary routers:** connect to other AS's.

Load Balancing drawbacks



Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, $O(nE)$ msgs sent each
- DV: exchange between neighbors only
 - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires $O(nE)$ msgs
 - may have oscillations
- DV: convergence time varies
 - may be routing loops
 - count-to-infinity problem

Robustness: what happens if router malfunctions?

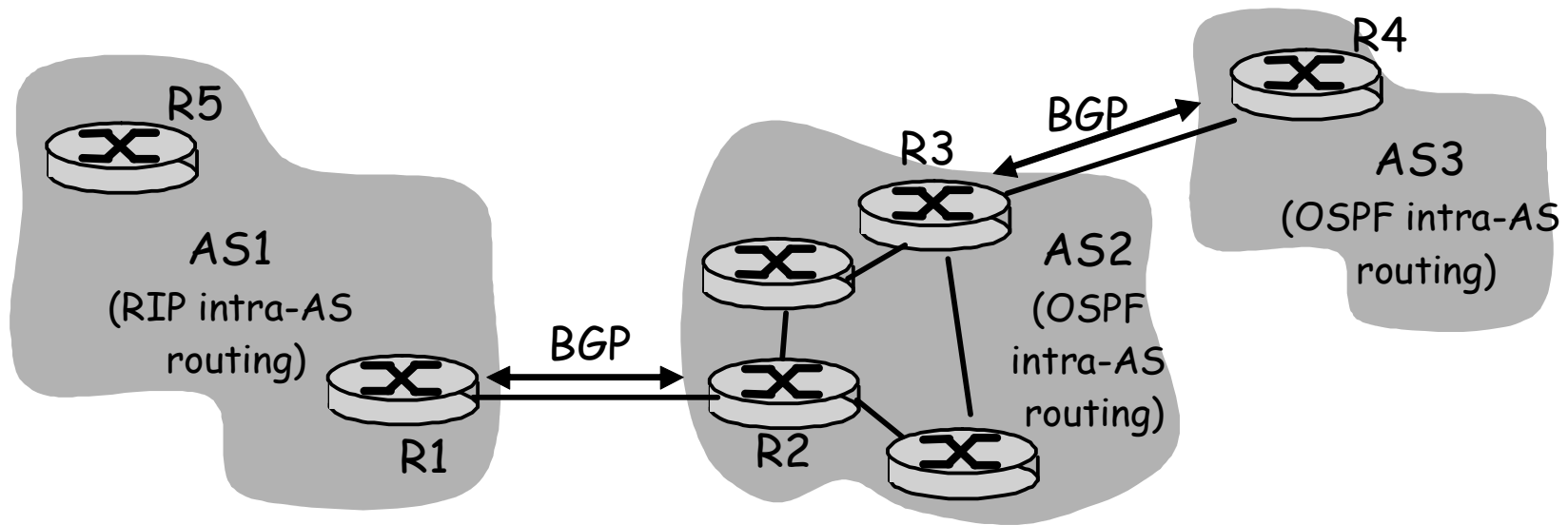
LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
 - error propagate thru network

Inter-AS routing in the Internet: BGP



AS2:

BGP used to determine routes to destinations outside AS2

OSPF routes to destinations within AS2

Internet inter-AS routing: BGP

- ❑ BGP (Border Gateway Protocol): *the de facto standard*
- ❑ **Path Vector** protocol:
 - similar to Distance Vector protocol
 - each Border Gateway broadcast to neighbors (BGP peers) *entire path* (i.e., sequence of AS's) to destination
 - BGP routes to networks (ASs), not individual hosts
 - E.g., Gateway X may send its path to dest. Z:

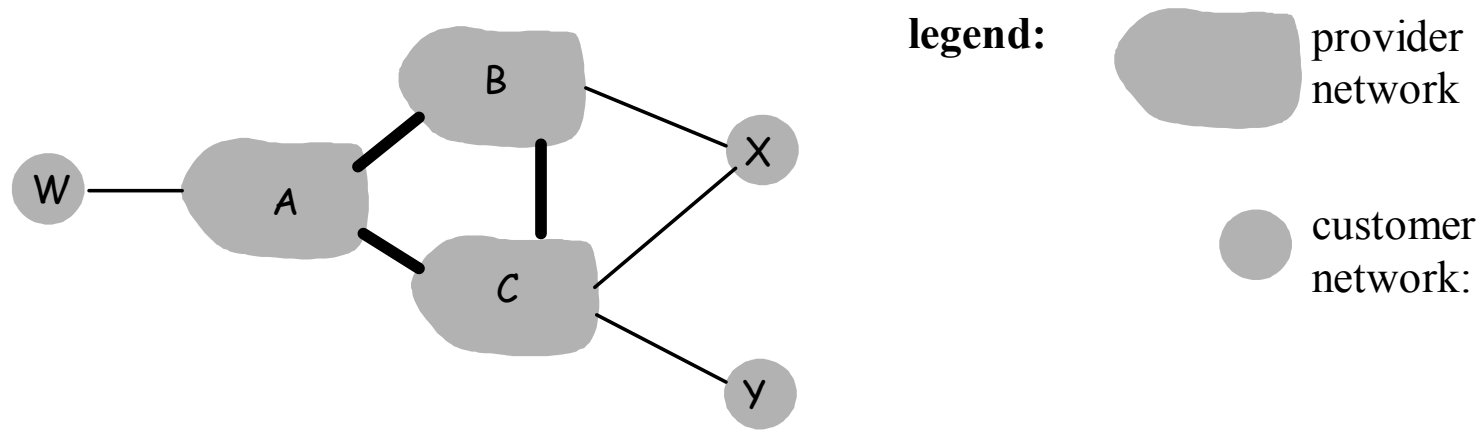
$\text{Path (X,Z)} = X, Y_1, Y_2, Y_3, \dots, Z$

Internet inter-AS routing: BGP

Suppose: gateway X send its path to peer gateway W

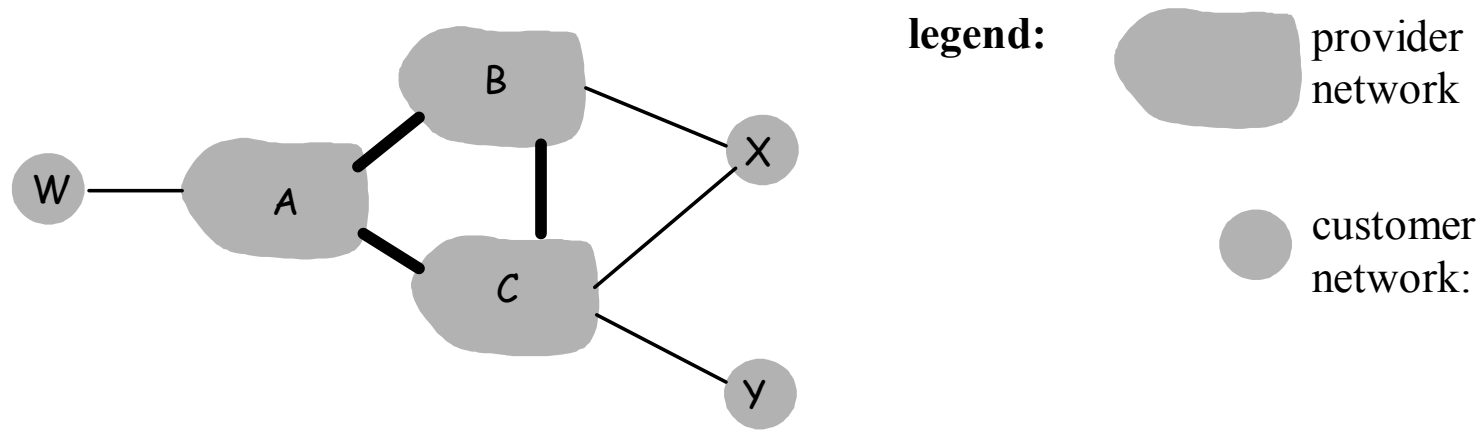
- ❑ W may or may not select path offered by X
 - cost, policy (don't route via competitors AS), loop prevention reasons.
- ❑ If W selects path advertised by X, then:
$$\text{Path}(W,Z) = w, \text{Path}(X,Z)$$
- ❑ Note: X can control incoming traffic by controlling its route advertisements to peers:
 - e.g., don't want to route traffic to Z -> don't advertise any routes to Z

BGP: controlling who routes to you



- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
 - X does not want to route from B via X to C
 - .. so X will not advertise to B a route to C

BGP: controlling who routes to you



- ❑ A advertises to B the path AW
- ❑ B advertises to W the path BAW
- ❑ Should B advertise to C the path BAW?
 - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
 - B wants to force C to route to w via A
 - B wants to route *only* to/from its customers!

BGP operation

Q: What does a BGP router do?

- ❑ Receiving and filtering route advertisements from directly attached neighbor(s).
- ❑ Route selection.
 - To route to destination X, which path (of several advertised) will be taken?
- ❑ Sending route advertisements to neighbors.

BGP messages

- ❑ BGP messages exchanged using TCP.
- ❑ BGP messages:
 - OPEN: opens TCP connection to peer and authenticates sender
 - UPDATE: advertises new path (or withdraws old)
 - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
 - NOTIFICATION: reports errors in previous msg; also used to close connection

Flooding (quale il routing piu' semplice ma anche il meno scalabile?-non usato dal livello rete dell'Internet)

- ❑ Obiettivo: inviare a tutti i nodi un messaggio con un determinato Sequence Number
- ❑ Quando un messaggio con SN arriva ad un nodo questo lo rforwarda verso tutti i suoi vicini
 - Tranne il nodo da cui il messaggio è stato ricevuto
 - Se non ha ancora mai forwardato messaggi con sequence number \geq SN
- ❑ Estremamente semplice. Richiede un numero di messaggi $O(\text{numero di link})$.
- ❑ Meno messaggi se si usa uno spanning tree MA necessario mantenere lo spanning tree, MENTRE flooding può essere effettuato immediatamente, richiede solo la conoscenza dei vicini.