Shortest Paths Algorithms and the Internet: The Distributed Bellman Ford

Lecturer: Prof. Chiara Petrioli

Dipartimento di Informatica
Rome University "La Sapienza"

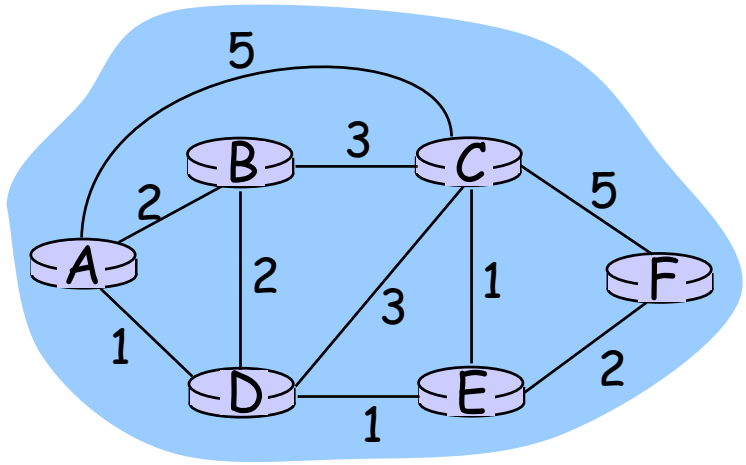
G205: Fundamentals of Computer Engineering

Material for the slides of this lecture was taken from the Kurose Ross book:
"Computer Networking: A top down approach featuring the Internet"

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 defs 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

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 shortest(<=h) walk and its length is D^h_i.

Bellman-Ford rule:

Initiatilization $D^h_s=0$, for all h; $w_{i,k}$ = infinity if (i,k) NOT in *A*; $w_{k,k}=0$; **D^0_i=infinity for all i!=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. →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:

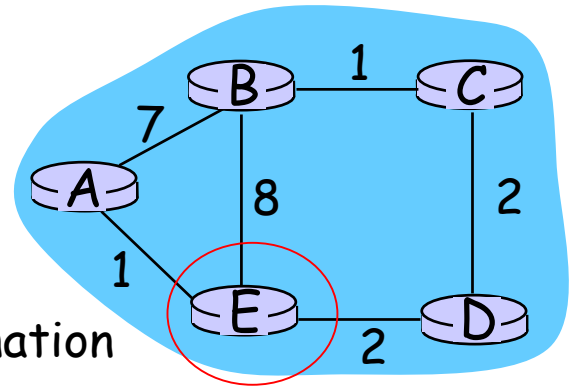
Cost associated to the (X,Z) link

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

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

Info maintained at Z. Min must be communicated

Distance Table: example



destination

$D^E(C,D) = c(E,D) + \min_w\{D^D(C,w)\}$
$= 2+2 = 4$

$D^E(A,D) = c(E,D) + \min_w\{D^D(A,w)\}$
$= 2+3 = 5$

loop! Best path from D goes through E

$D^E(A,B) = c(E,B) + \min_w\{D^B(A,w)\}$
$= 8+6 = 14$

loop!

Path B-C-D-E-A

Distance table in node E after the algorithm has converged

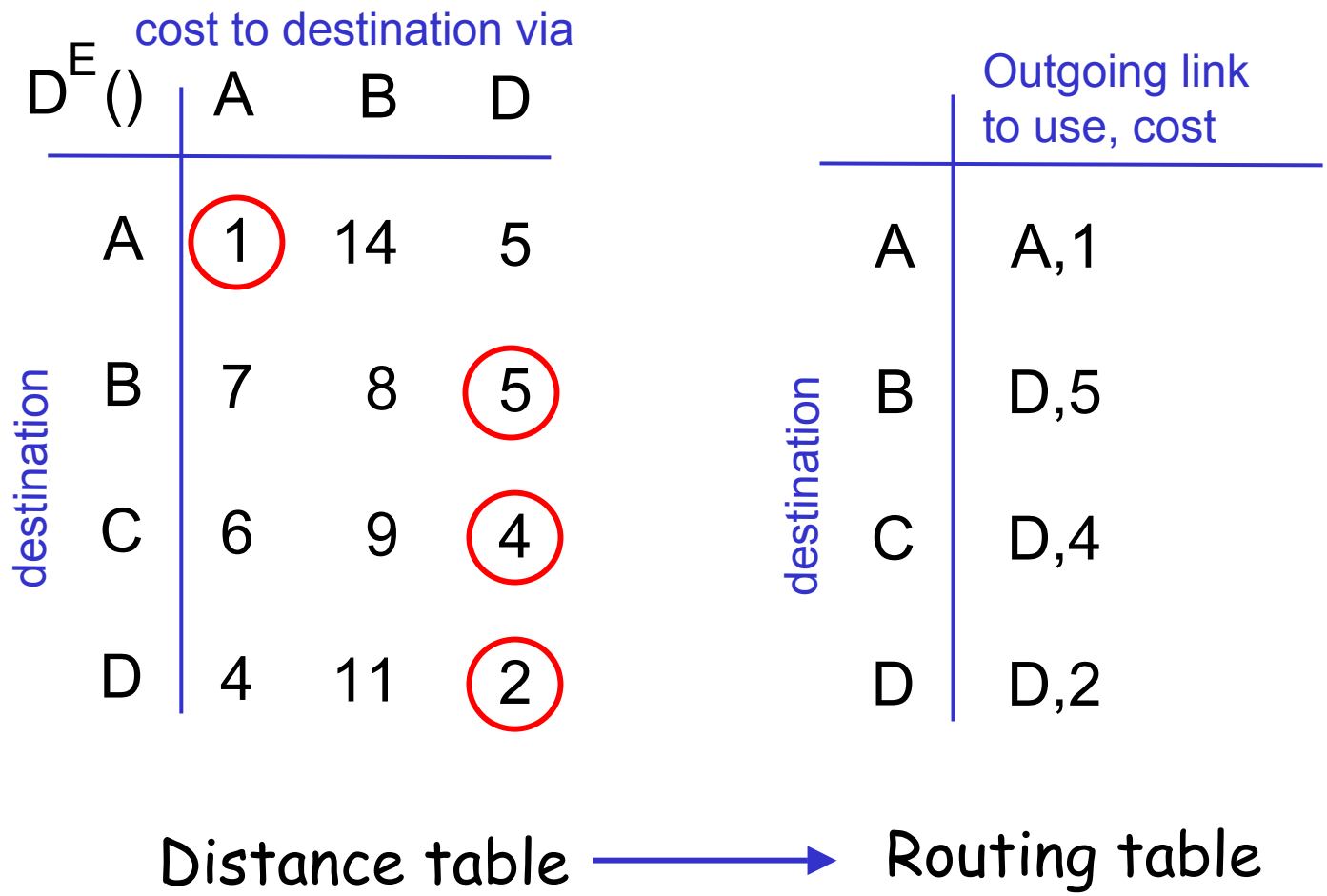
cost to destination via

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

destination

First example

Distance table gives routing table



Distance Vector Routing: overview

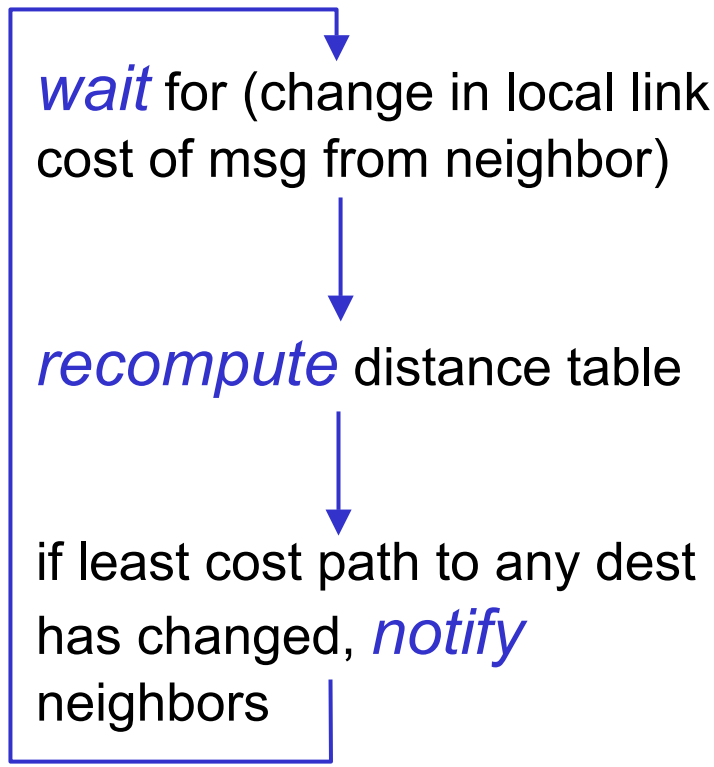
Iterative, asynchronous:
Each local iteration caused by:

- 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:



There are periodic exchanges of estimates

Distance Vector Algorithm:

At all nodes, X:

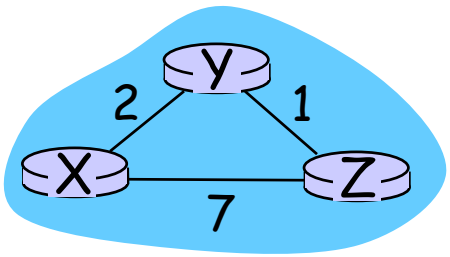
1 Initialization:
2 for all adjacent nodes v:
3 $D^X(*,v) = \text{infinity}$ /* the * operator means "for all rows" */
4 $D^X(v,v) = c(X,v)$
5 for all destinations, y
6 send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

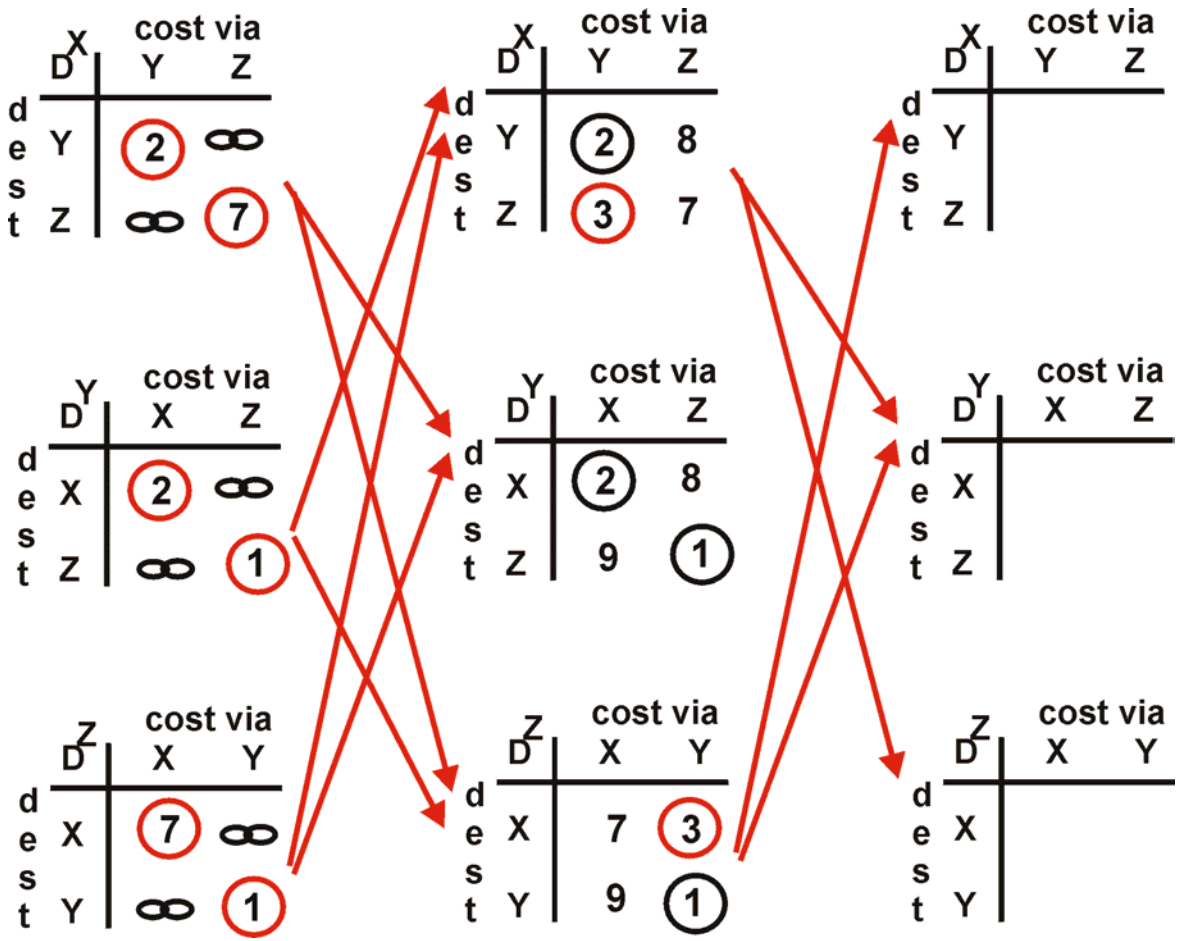
From the node to whatever destination going through v

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 D^V(Y,w) */
20     /* call this received new value is "newval"     */
21     for the single destination y: D^X(Y,V) = c(X,V) + 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   + periodic asynchronous transmissions
26  forever
```

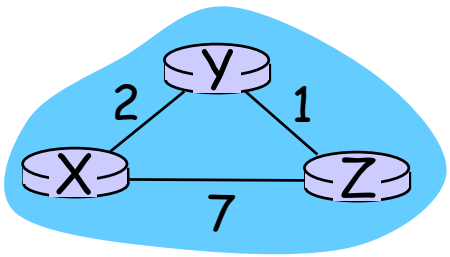
Distance Vector Algorithm: example





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



D^X

dest	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y

dest	cost via X	cost via Z
X	2	∞
Z	∞	1

D^Z

dest	cost via X	cost via Y
X	7	∞
Y	∞	1

D^X

dest	cost via Y	cost via Z
Y	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:
  - link weights and topology 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

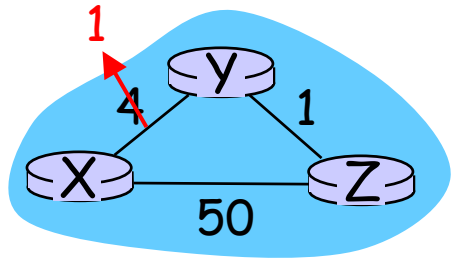
the Distributed Bellman Ford algorithm converges, i.e. each node correctly estimates the cost of the best route to the destination

Distance Vector: link cost changes

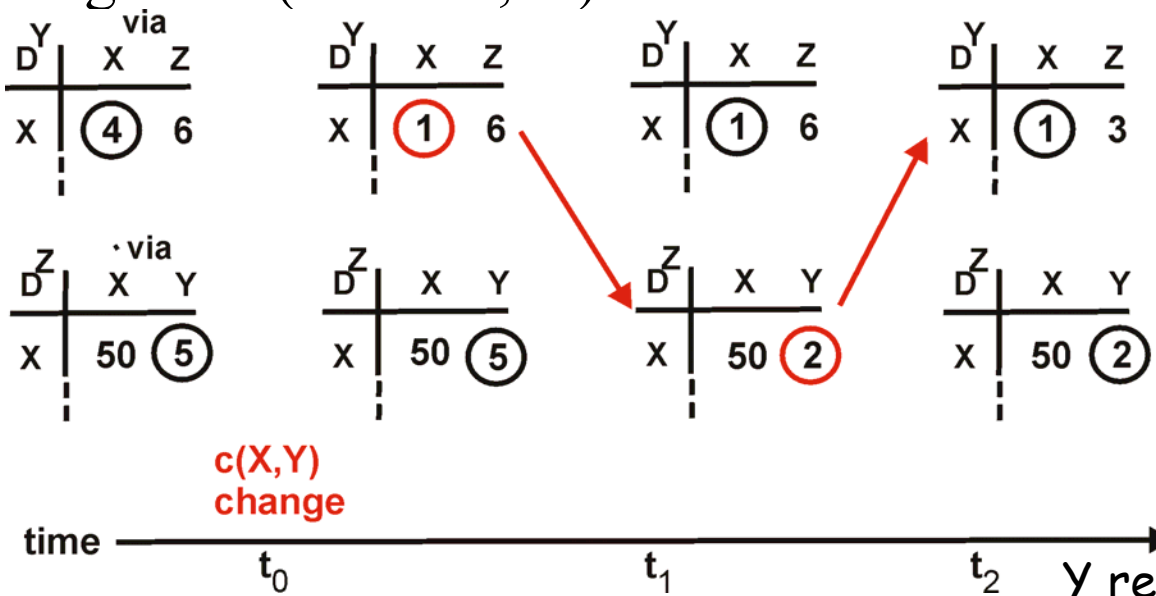
Subtitle: Distributed Bellman Ford converges *but* how fast?

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"



algorithm Terminates (exchange of info triggered by change completes

Ydetects change informs neighbors

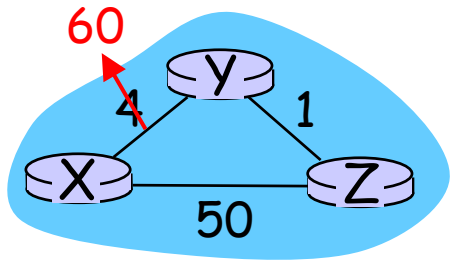
Z receives update updates table, new least Cost→informs neighbors

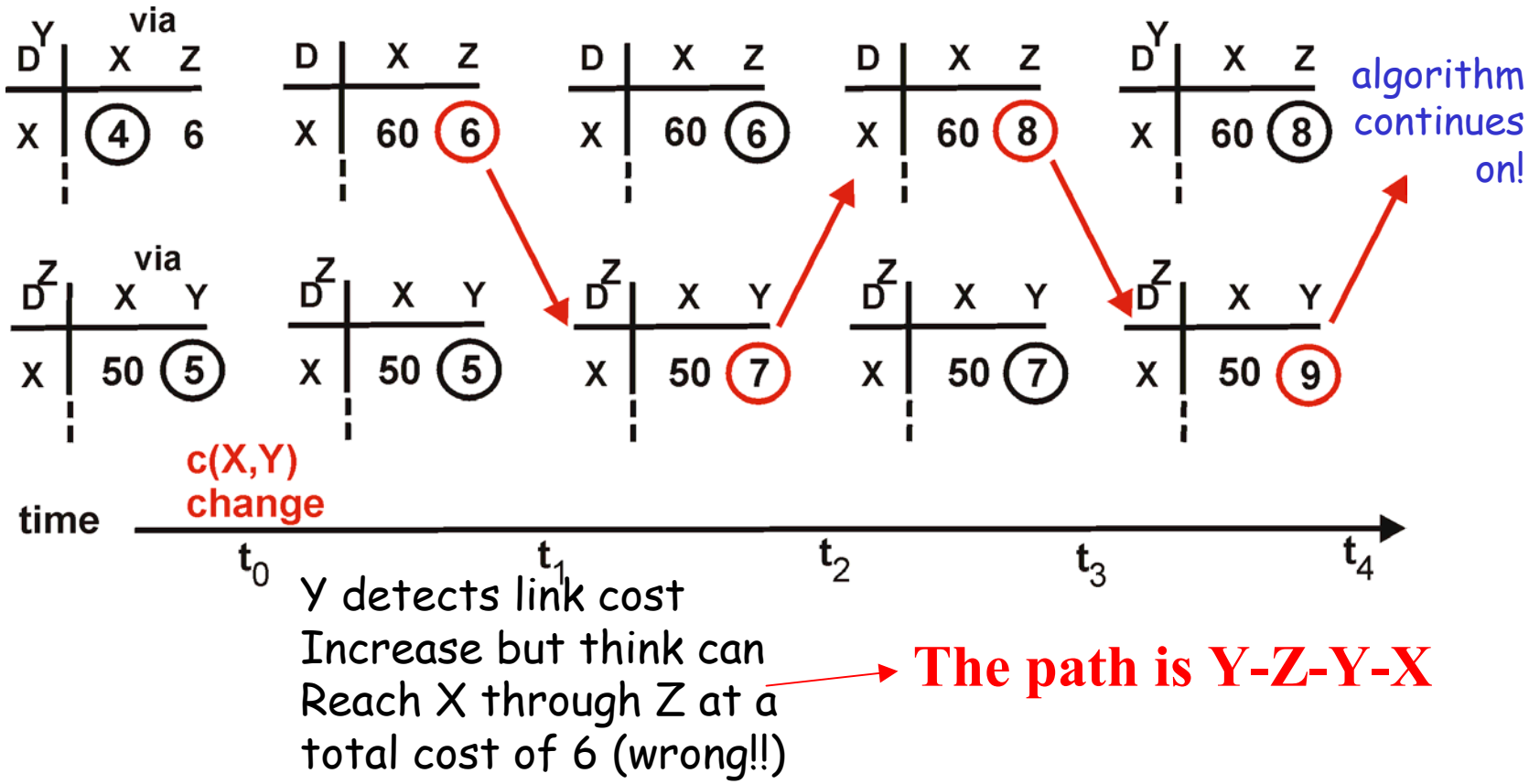
Y receives new least cost; no modification in the routing table, No updates sent

Distance Vector: link cost changes

Link cost changes:

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





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 Bostonian example…

-assumption: there is only one route going from Northeastern to Prudential center: Huntington Ave. Let us now consider a network, whose nodes are Northeastern Un., Simphony Hall, Prudential Center

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

The Northeastern Un. and Symphony nodes exchange the following info

- NU says 'the shortest route from me to Prudential is 1 mile'
- Symphony says 'the shortest path from me to Prudential is .5 mile'

Based on this exchange from NU you go to SY, and from there to Prudential OK

Now due to the big dig they close Huntington Ave from Symphony to Prudential

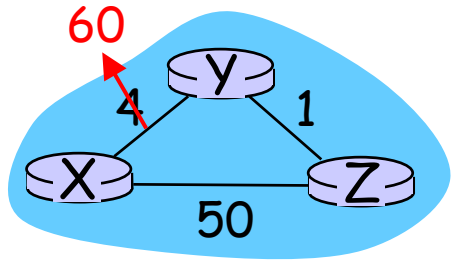
- Symphony thinks 'I have to find another route from me to Prudential. Look there is a route from Northeastern University to Prudential that takes 1 mile, I can be at Northearn University in .5 mile → I have found a 1.5 mile route from me to Prudential!!' Communicates the new cost to Northeastern that updates 'OK I can go to Prudential via SY in 2 miles'

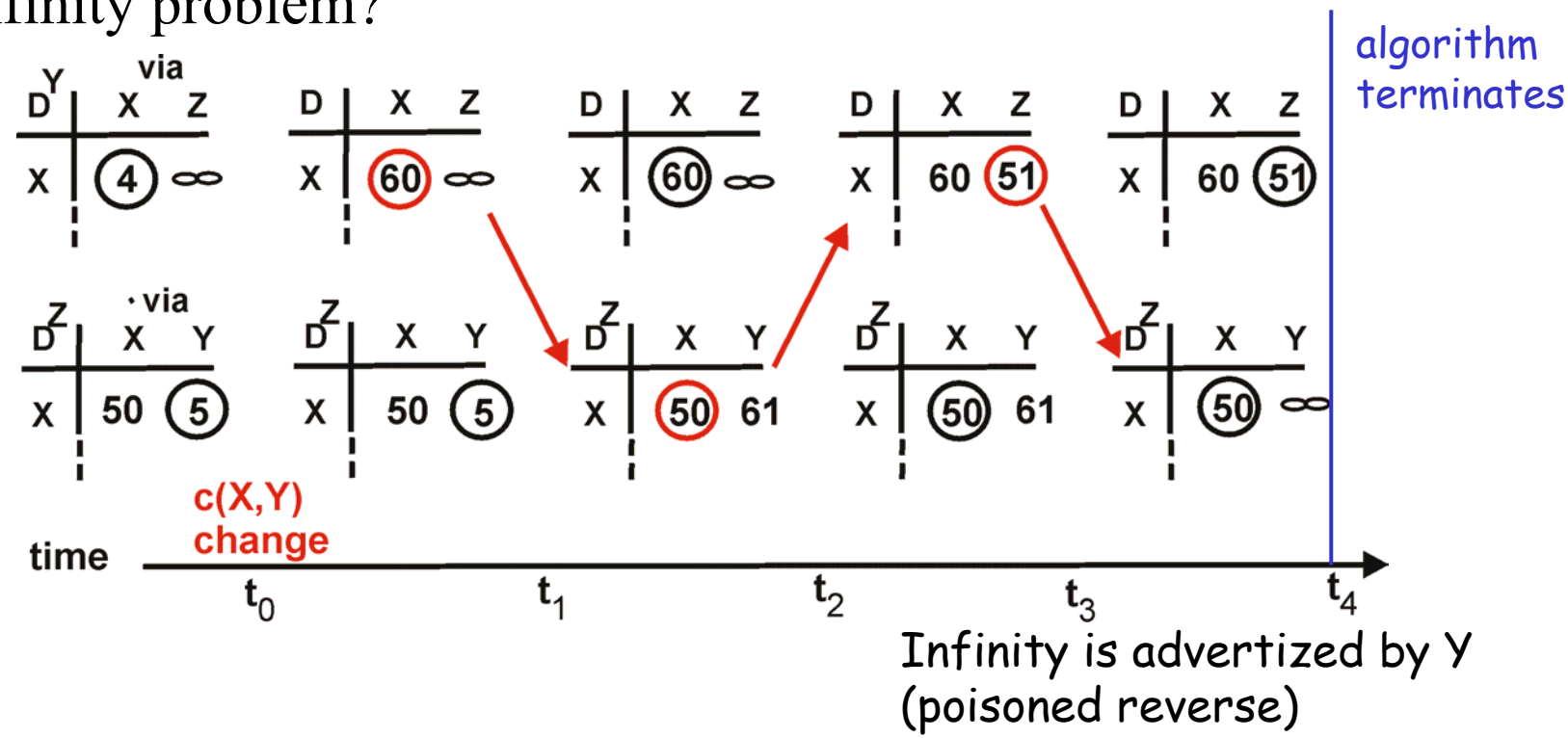
VERY WRONG!! Why is it so? I didn't know that the route from Northeastern Un. to Prudential was going through the section of Hungtinton Ave. from Symphony to Prudential (which is closed)!!

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?





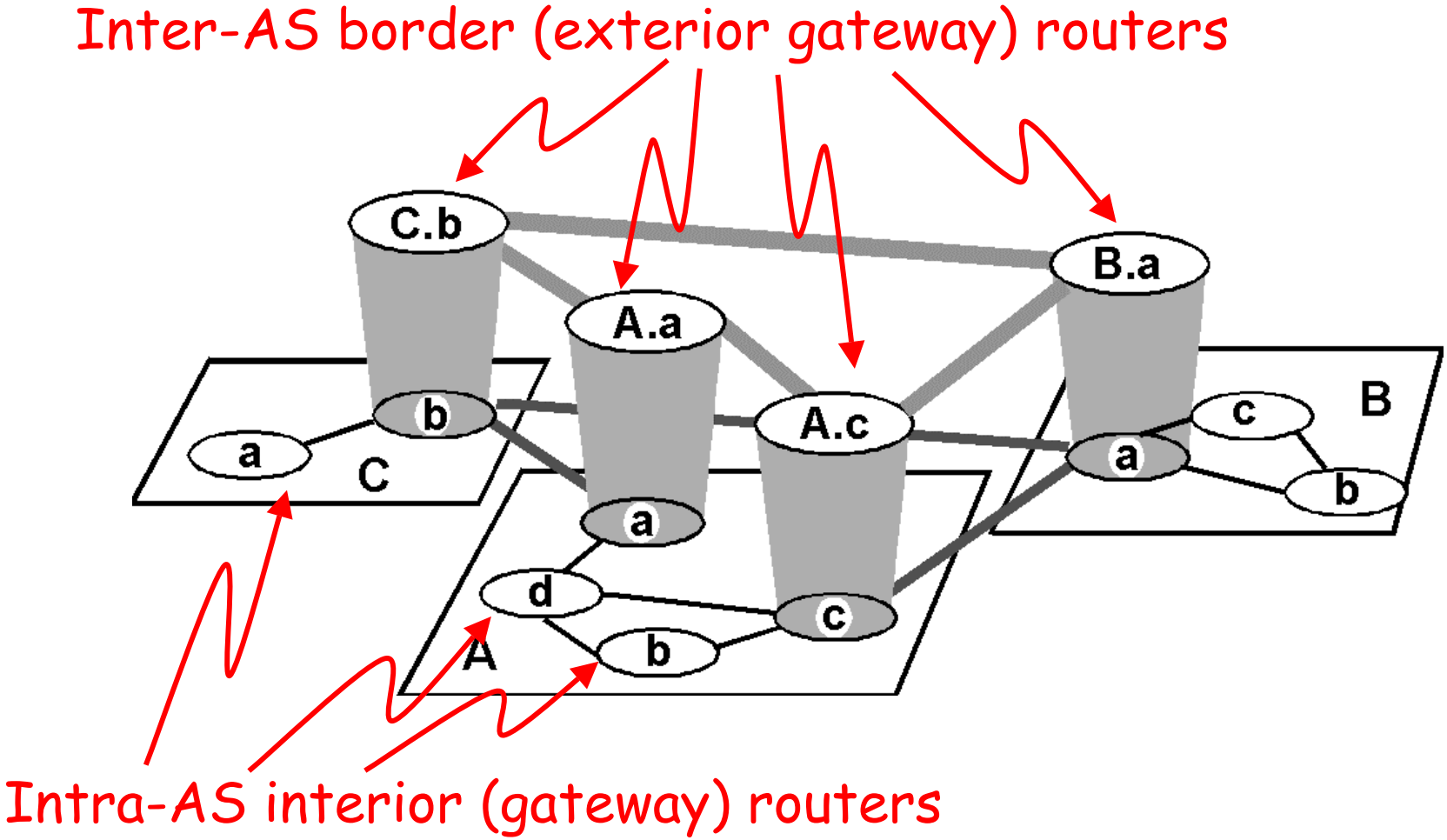
Routing in the Internet

- 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 leats some degree of technical And administrative control) by a single entity, characterized by a given Routing technology.

Internet AS Hierarchy



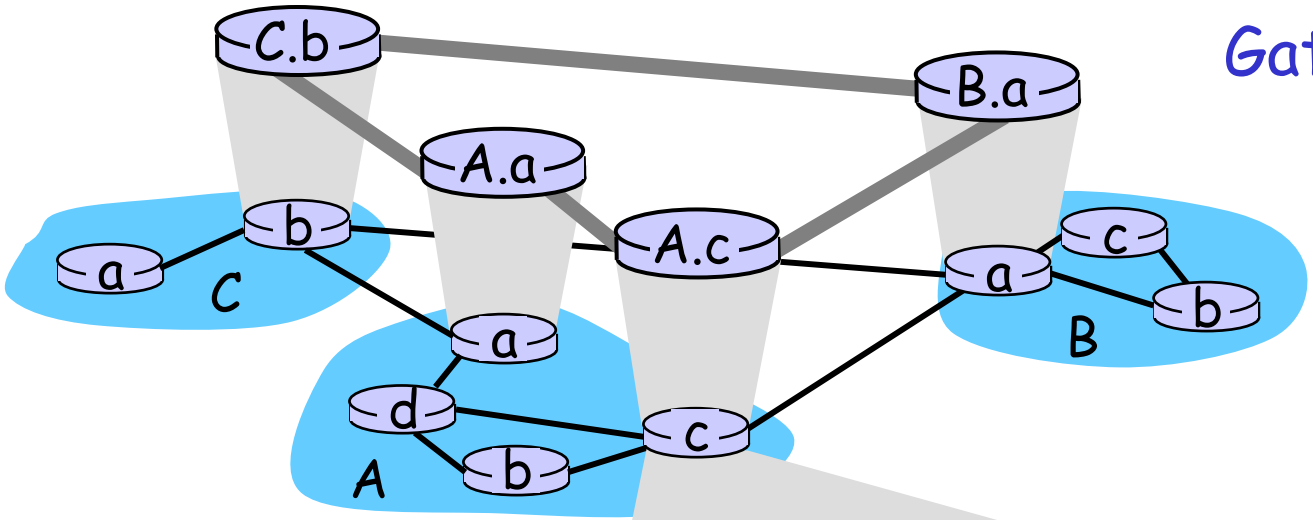
Hierarchical Routing

- 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

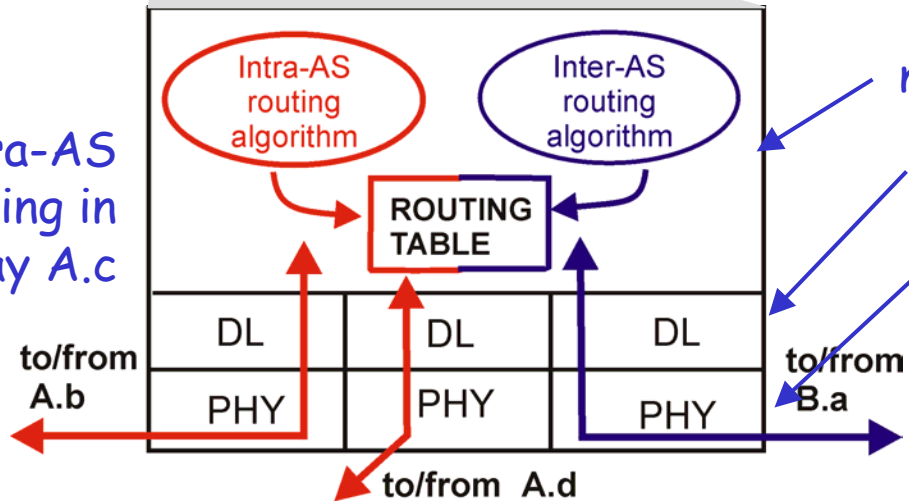
Intra-AS and Inter-AS routing



Gateways:

- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS

inter-AS, intra-AS routing in gateway A.c

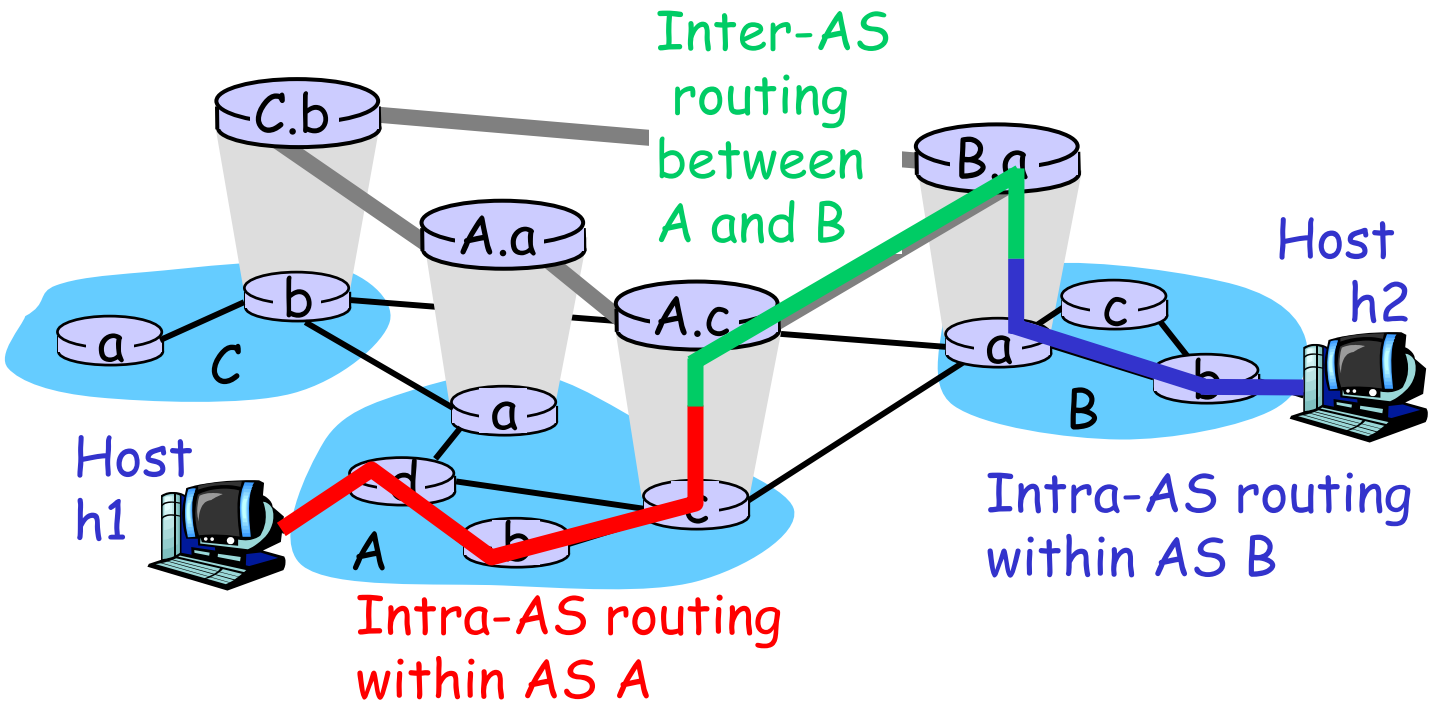


network layer

link layer

physical layer

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)

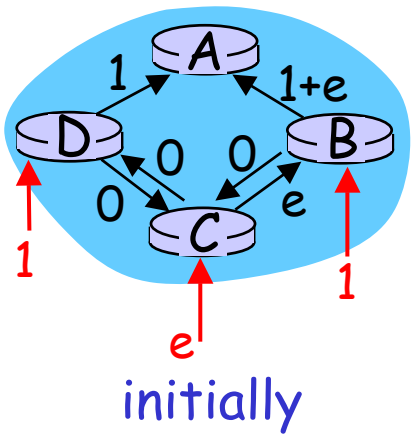
Routing instability

Number of hops simple metric. Why don't we use a more involved metric?, e.g., link cost = amount of carried traffic

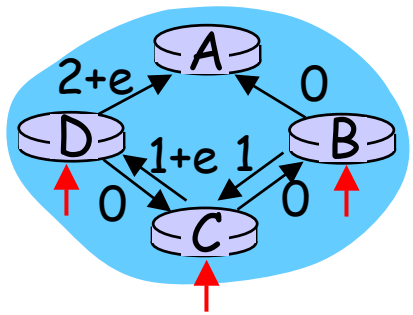
If less traffic goes from A to B then lower delays experienced from A to B (think at what happens if you have to select a register line at the supermarket…)

Oscillations possible:

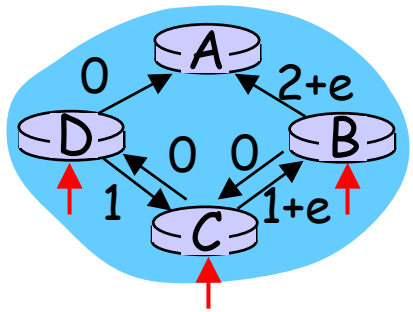
Problem: based on computation more or less packets use the same path→but this changes the edges weight!!



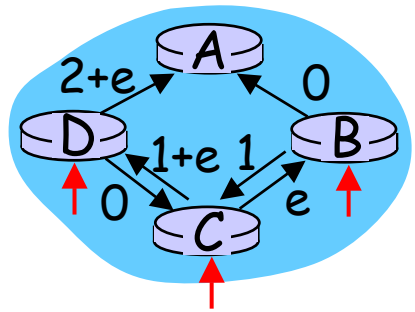
initially



… recompute routing



… recompute



… recompute

Usually metric adopted in routing simply the number of hops

OSPF (Open Shortest Path First)

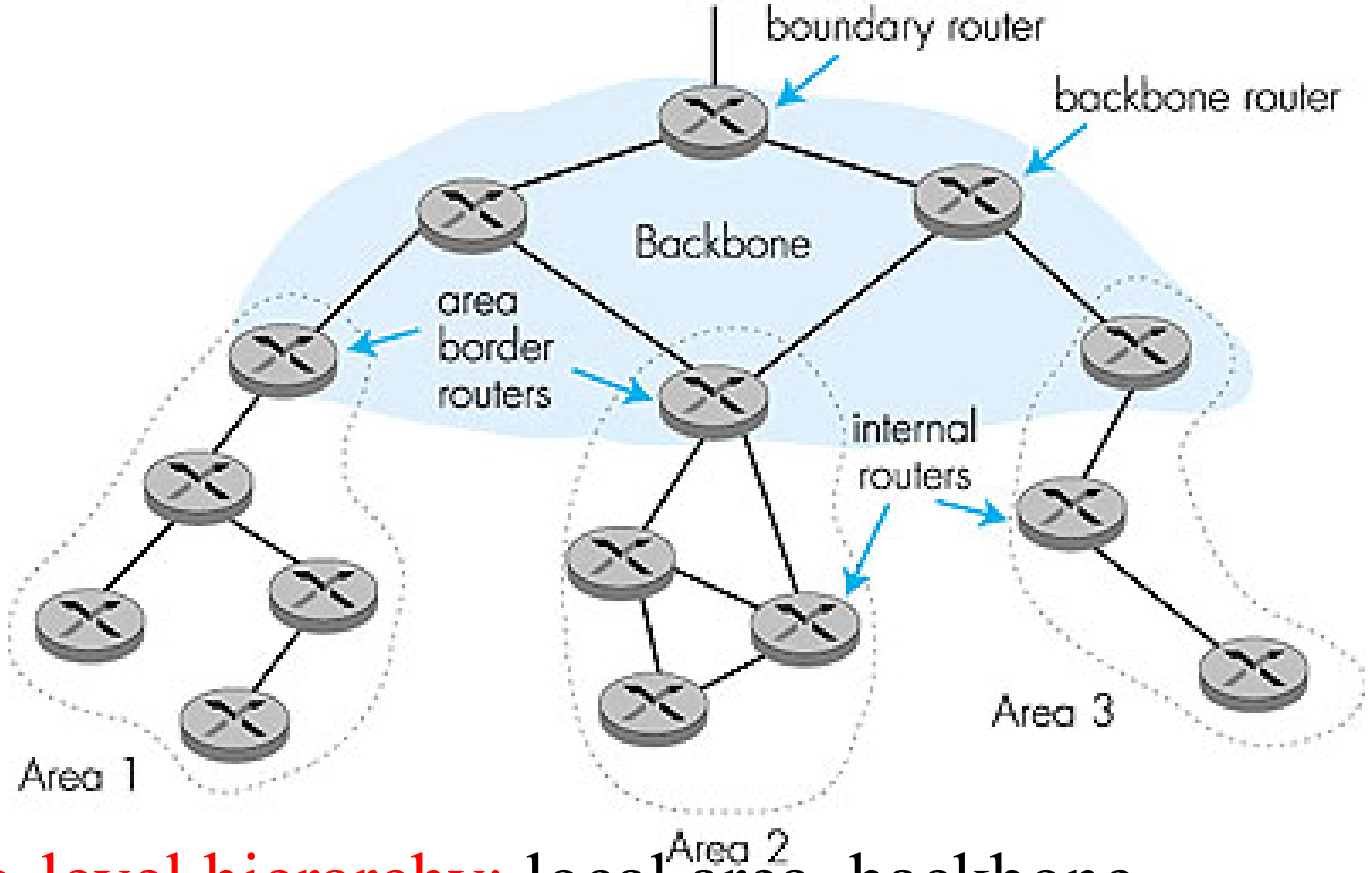
- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm

- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)
  - Carried in OSPF messages directly over IP

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.

Further readings

If you still have doubts:

Asynchronous, distributed Bellman Ford description+
formal proof it converges to shortest paths in:

D.Bertsekas, R. Gallager, “Data Networks”, Prentice Hall 1992,
Pag 404-410

Internet drafts su RIP and OSPF (the latter unreadable):
available on the web