50.012 Networks

Lecture 14: Network Layer Control Plane Overview

2021 Term 6

Assoc. Prof. CHEN Binbin

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Outline

Control Plane

- routing protocols
- link state
- distance vector

Read textbook Section 5.1, 5.2

Network-layer functions

Recall: two network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output

  data plane

- *routing:* determine route taken by packets from source to destination

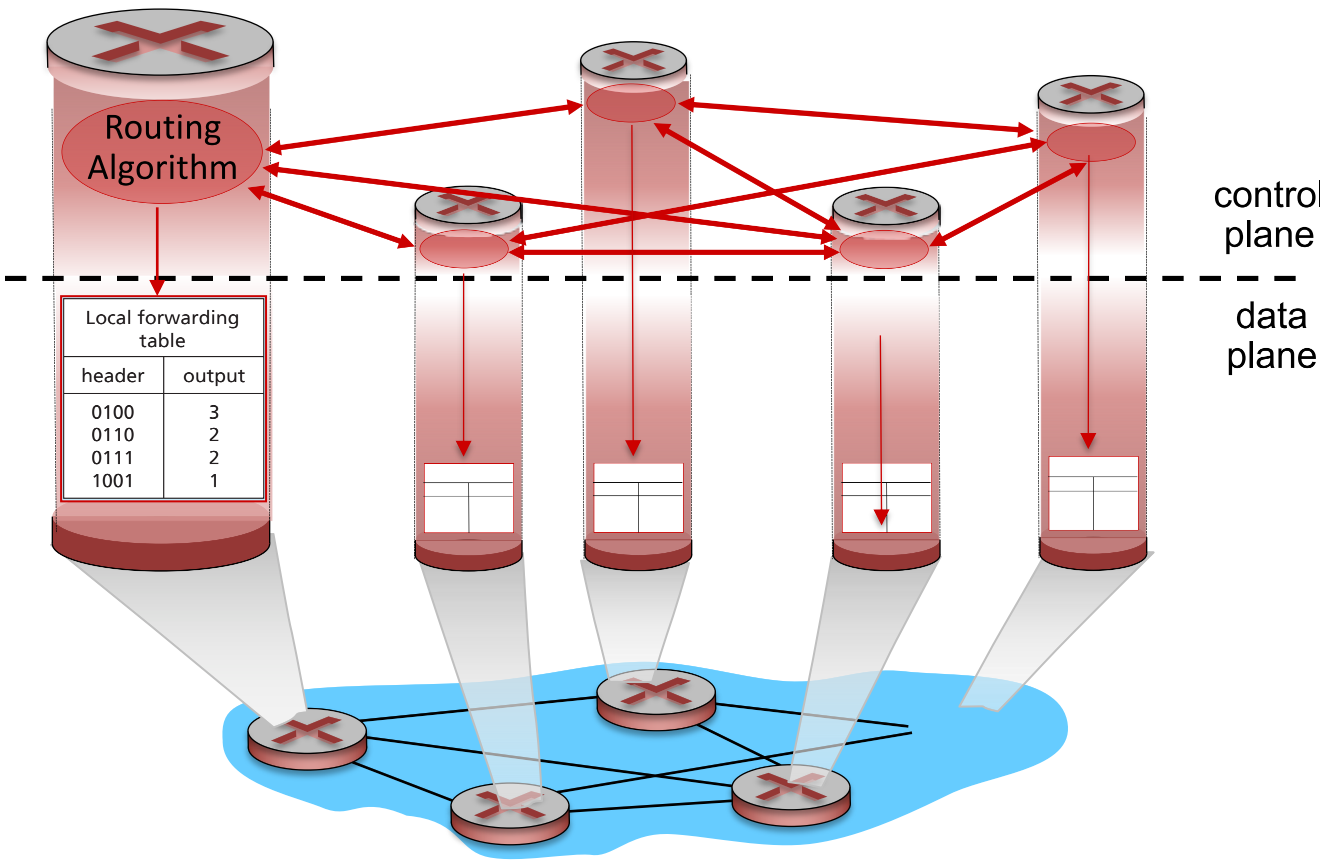
  control plane

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

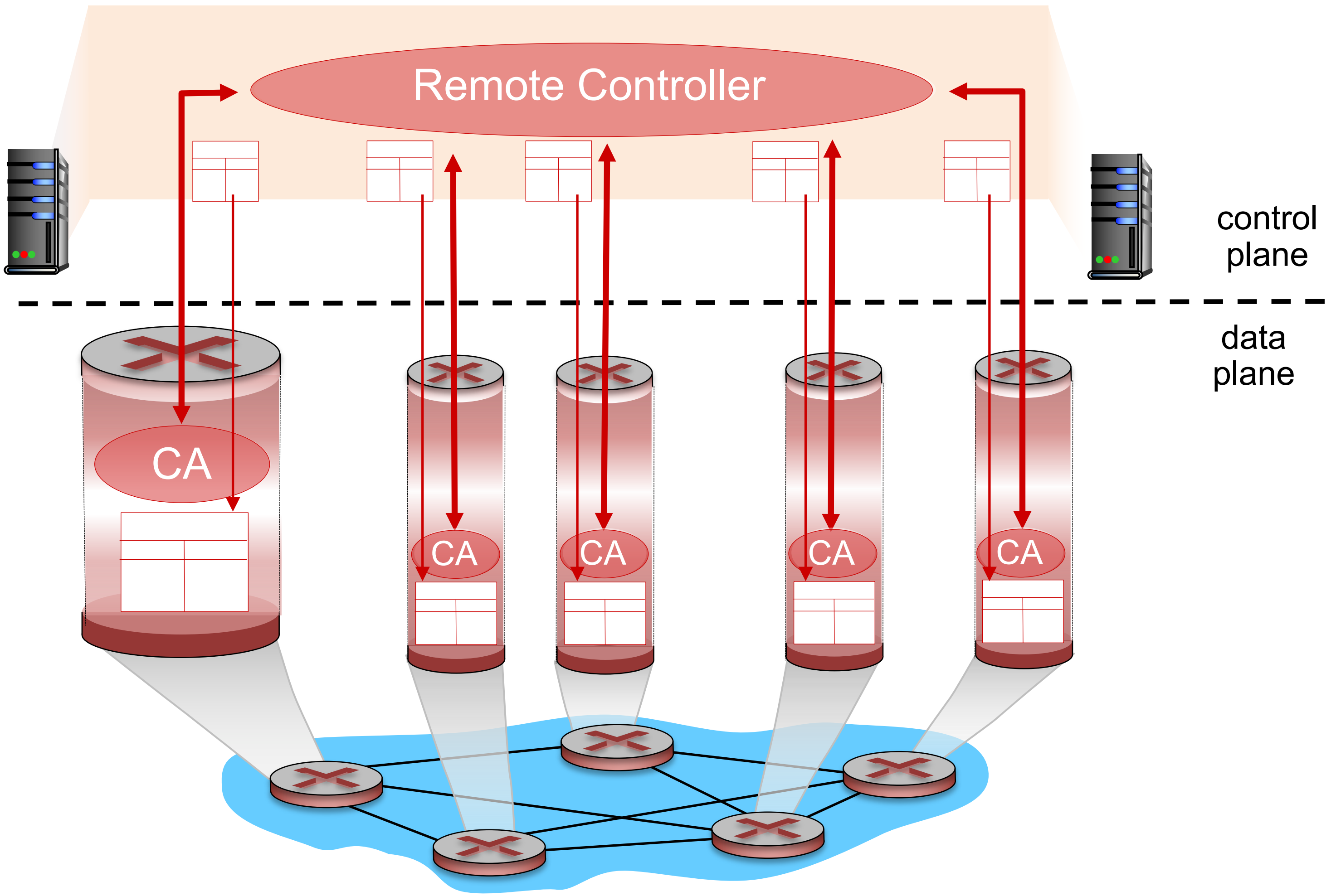
Per-router control plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

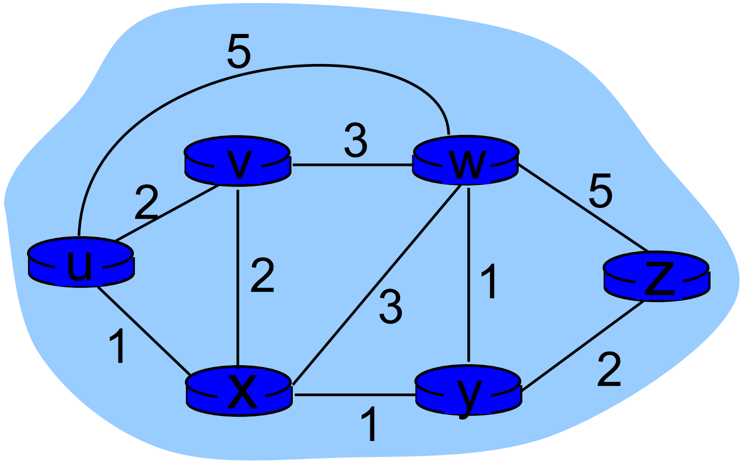


Routing protocols

Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- "good": least "cost", "fastest", "least congested"
- routing: a "top-10" networking challenge!

Graph abstraction of the network



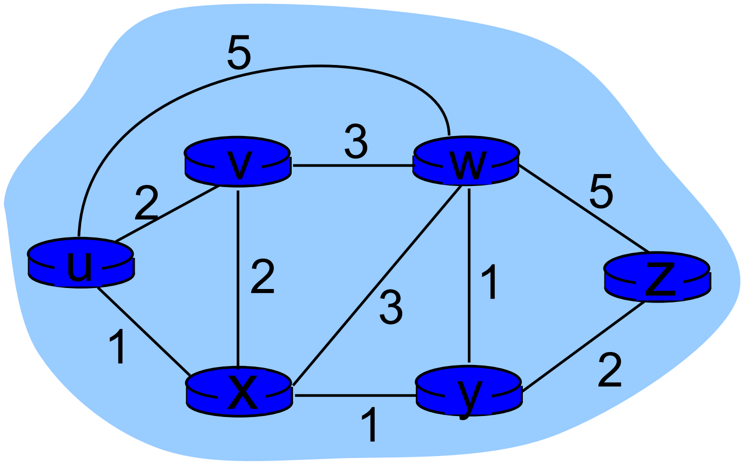
graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph abstraction: costs



$c(x,x')$ = cost of link (x,x')
e.g., $c(w,z) = 5$

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: 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
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

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

Outline

Control Plane

- routing protocols
- link state
- distance vector

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 *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

notation:

- c(x,y): link cost from node x to y; = ∞ 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
- N': set of nodes whose least cost path definitively known

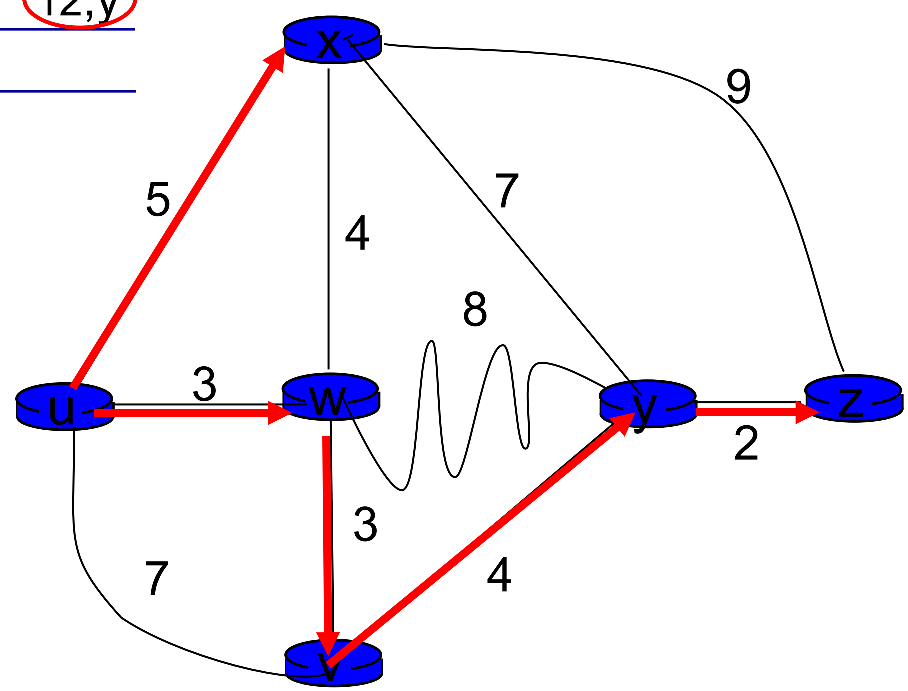
Dijsktra's algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

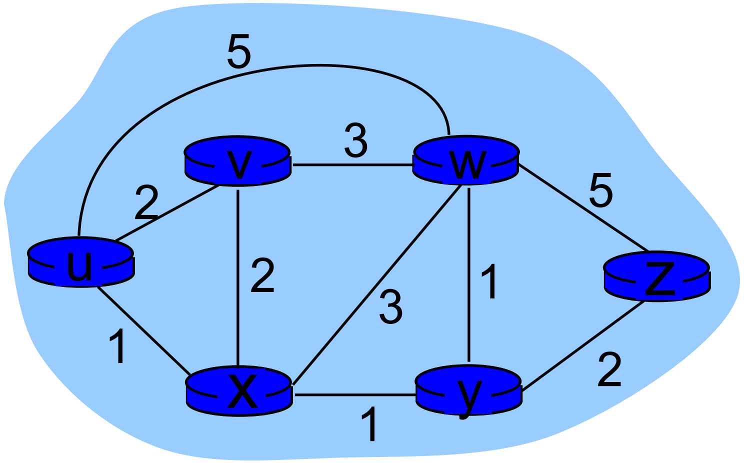


notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Dijkstra's algorithm: another example

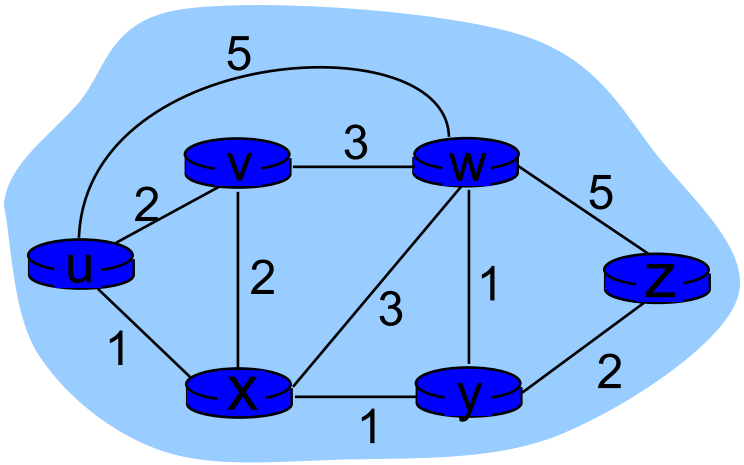
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0						
1						
2						
3						
4						
5						



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Dijkstra's algorithm: another example

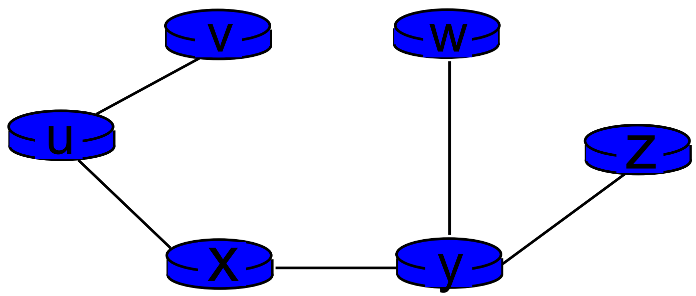
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



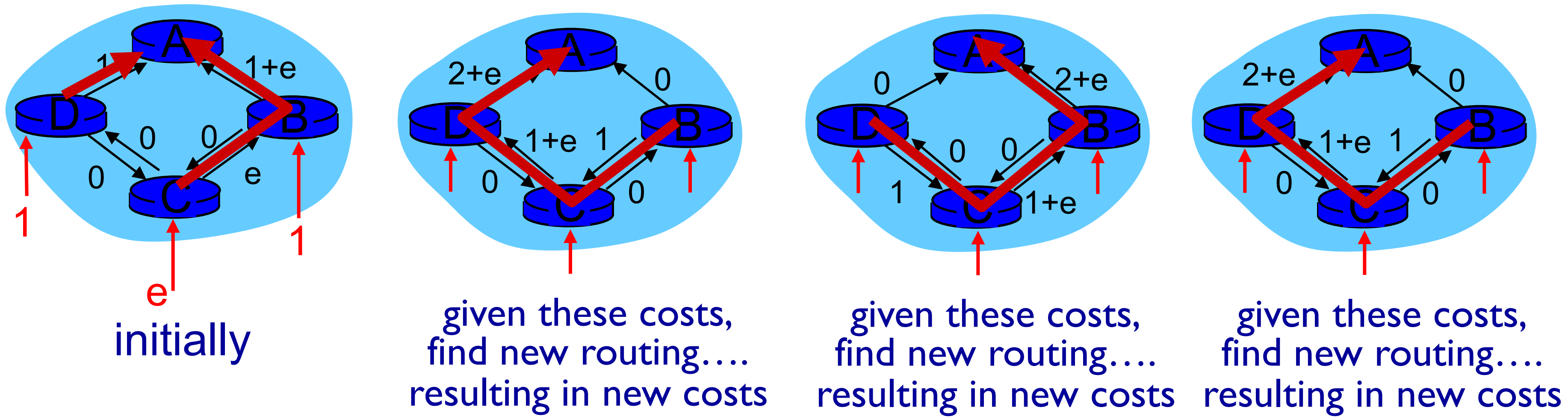
resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Dijkstra's algorithm, discussion

oscillations possible:

- e.g., suppose link cost equals amount of carried traffic:



Outline

Control Plane

- routing protocols
- link state
- distance vector

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

$d_x(y)$:= cost of least-cost path from x to y

then

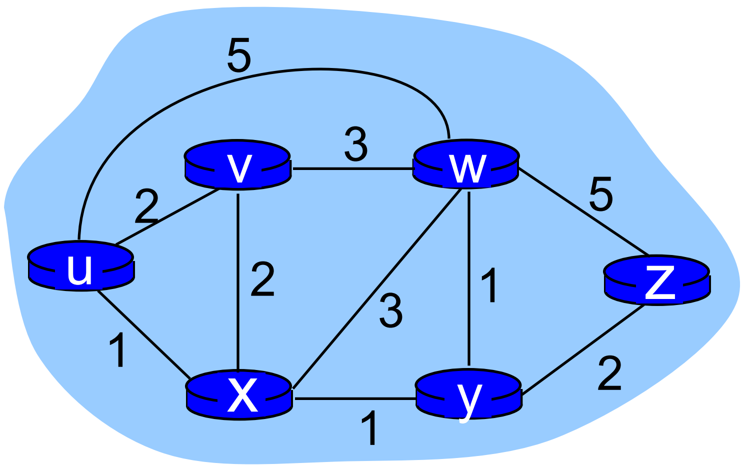
$$d_x(y) = \min_v\{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



u's neighbors: v, x, and w
$d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$
\begin{aligned}
d_u(z) = \min \{ & c(u,v) + d_v(z), \\
& c(u,x) + d_x(z), \\
& c(u,w) + d_w(z) \} \\
= \min \{ & 2 + 5, \\
& 1 + 3, \\
& 5 + 3\} = 4
\end{aligned}
$$

node achieving minimum is next hop in shortest path, used in forwarding table

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains
    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance vector algorithm

iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

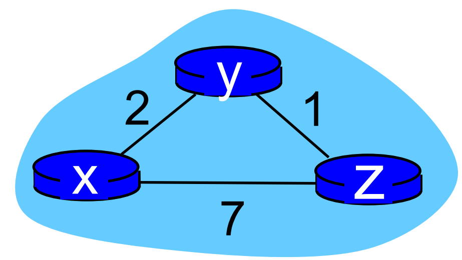
node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

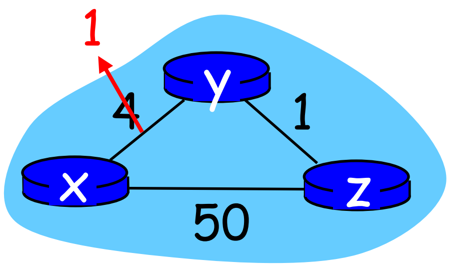
time



Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

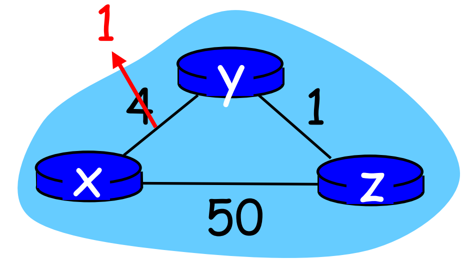
t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance vector: good news travel fast



node x table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

node y table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

time

Distance vector: good news travel fast

1 2

node x table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

node y table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

1

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

node z table

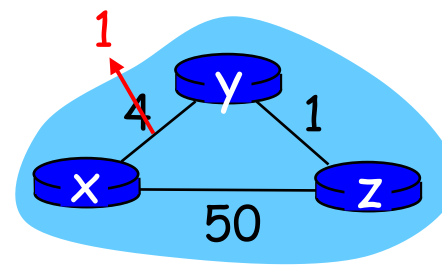
from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

2

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

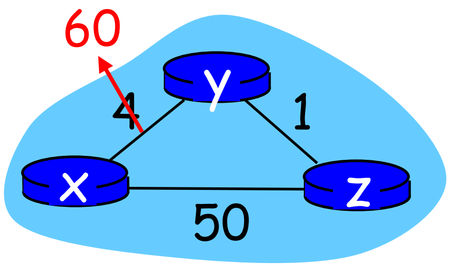
time



Distance vector: link cost changes

link cost changes:

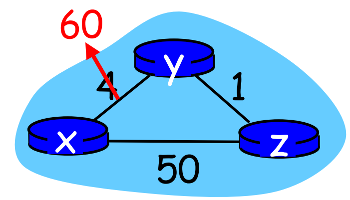
- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text



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?

count to infinity



node x table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

node y table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

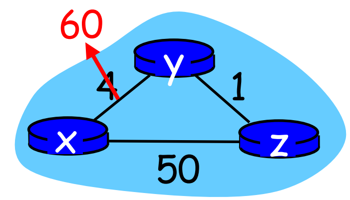
from \ cost to	x	y	z
x			
y			
z			

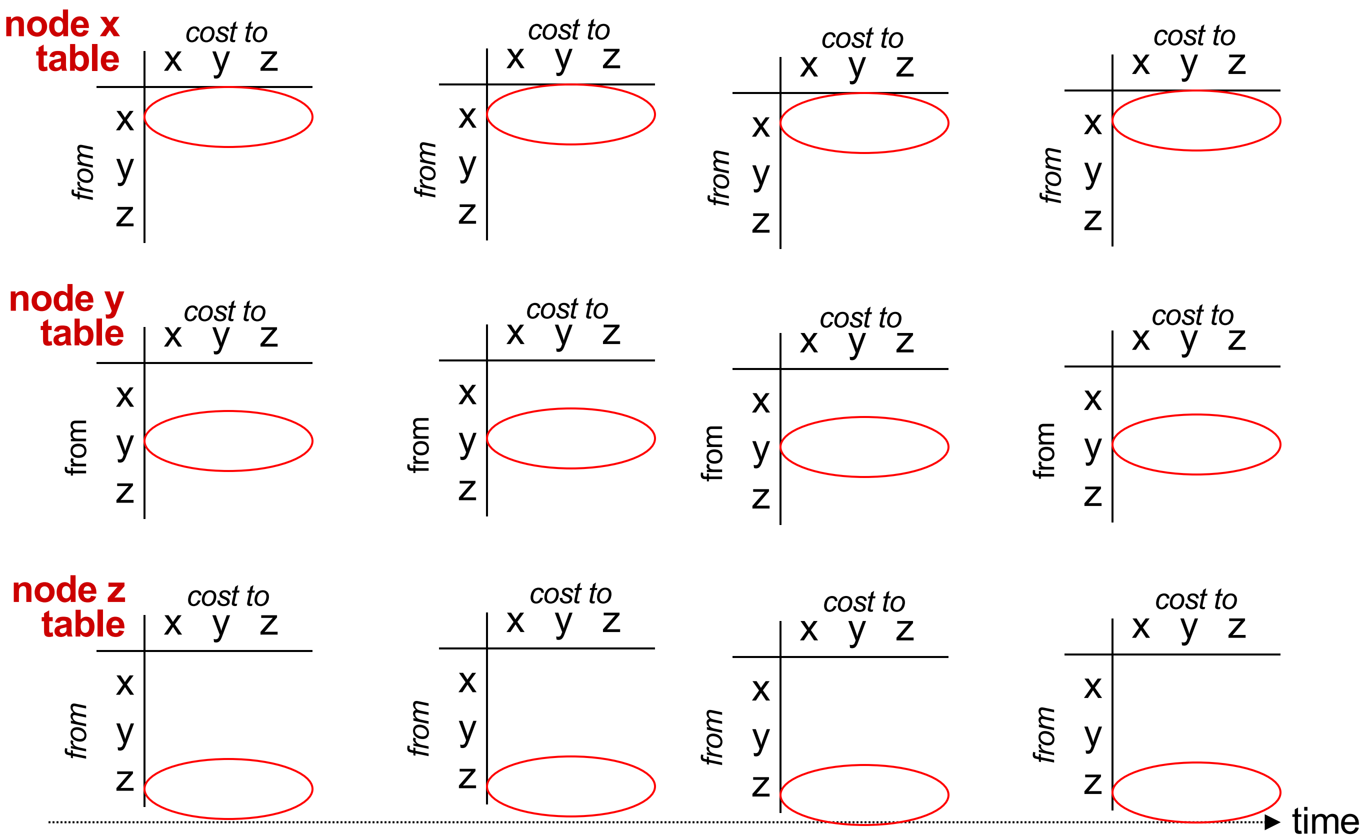
from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			

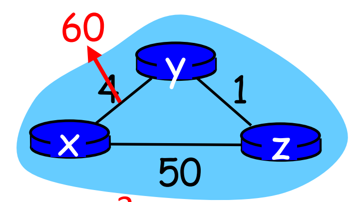
time

count to infinity, continued.





count to infinity



node x table (round 0)

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

51 50

node y table (round 0)

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

6

node z table (round 0)

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

round 1

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

7

2

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

8

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

3

from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

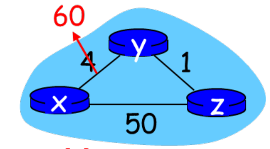
from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

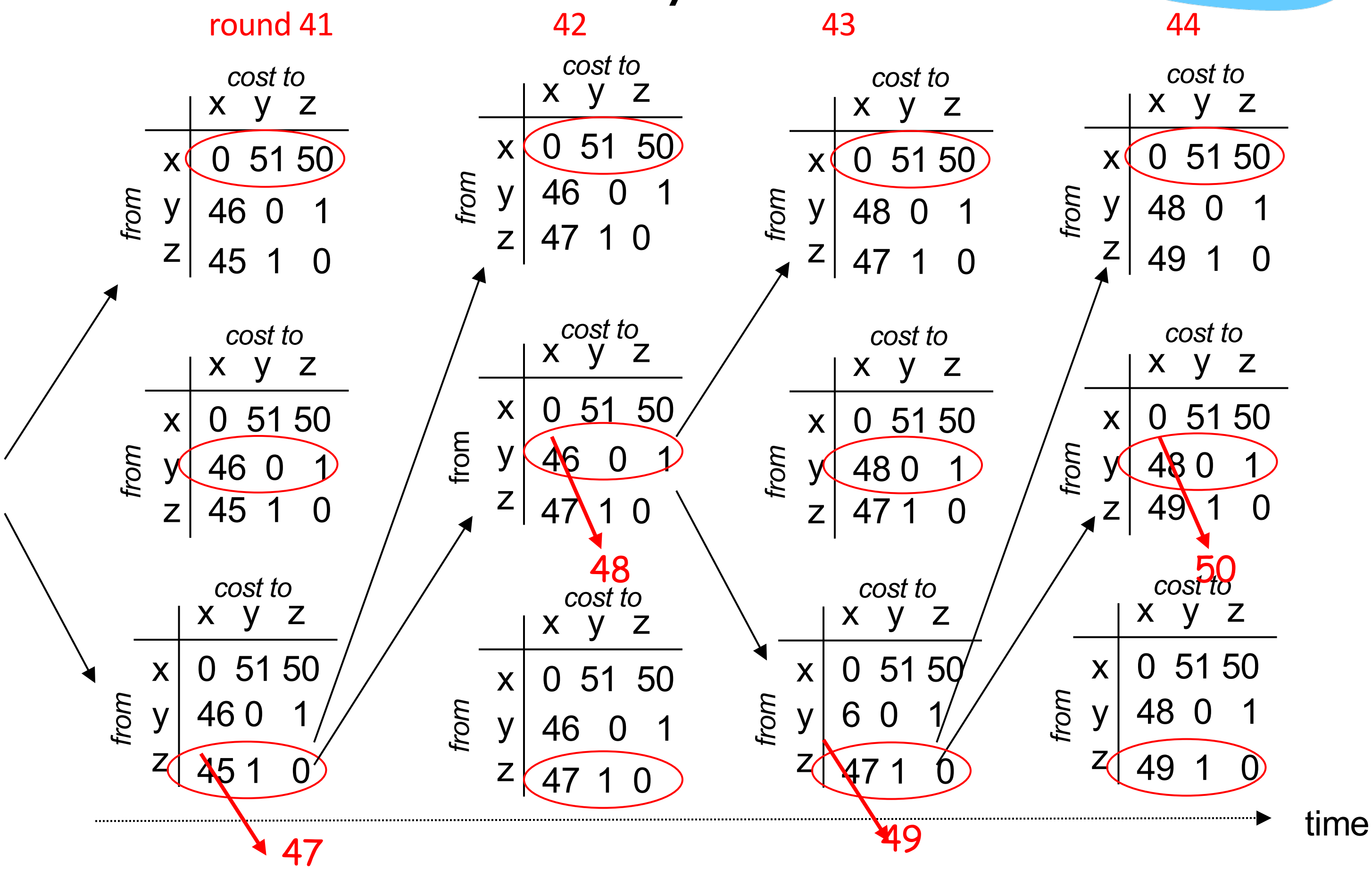
from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

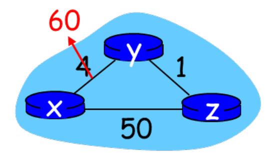
9

time

count to infinity, continued.







count to infinity, continued.

round 45

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

46

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	50	1	0

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	50	1	0

51

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	50	1	0

47

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

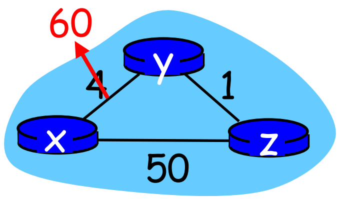
from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

time

50 (z finally decides to go to x directly)

Adding poisoned reverse



node x table

cost to

from	x	y	z
x			
y			
z			

cost to

from	x	y	z
x			
y			
z			

cost to

from	x	y	z
x			
y			
z			

cost to

from	x	y	z
x			
y			
z			

node y table

cost to

from	x	y	z
x			
y			
z	∞	1	0

cost to

from	x	y	z
x			
y			
z			

cost to

from	x	y	z
x			
y			
z			

cost to

from	x	y	z
x			
y			
z			

node z table

cost to

from	x	y	z
x			
y			
z	5	1	0

cost to

from	x	y	z
x			
y			
z			

cost to

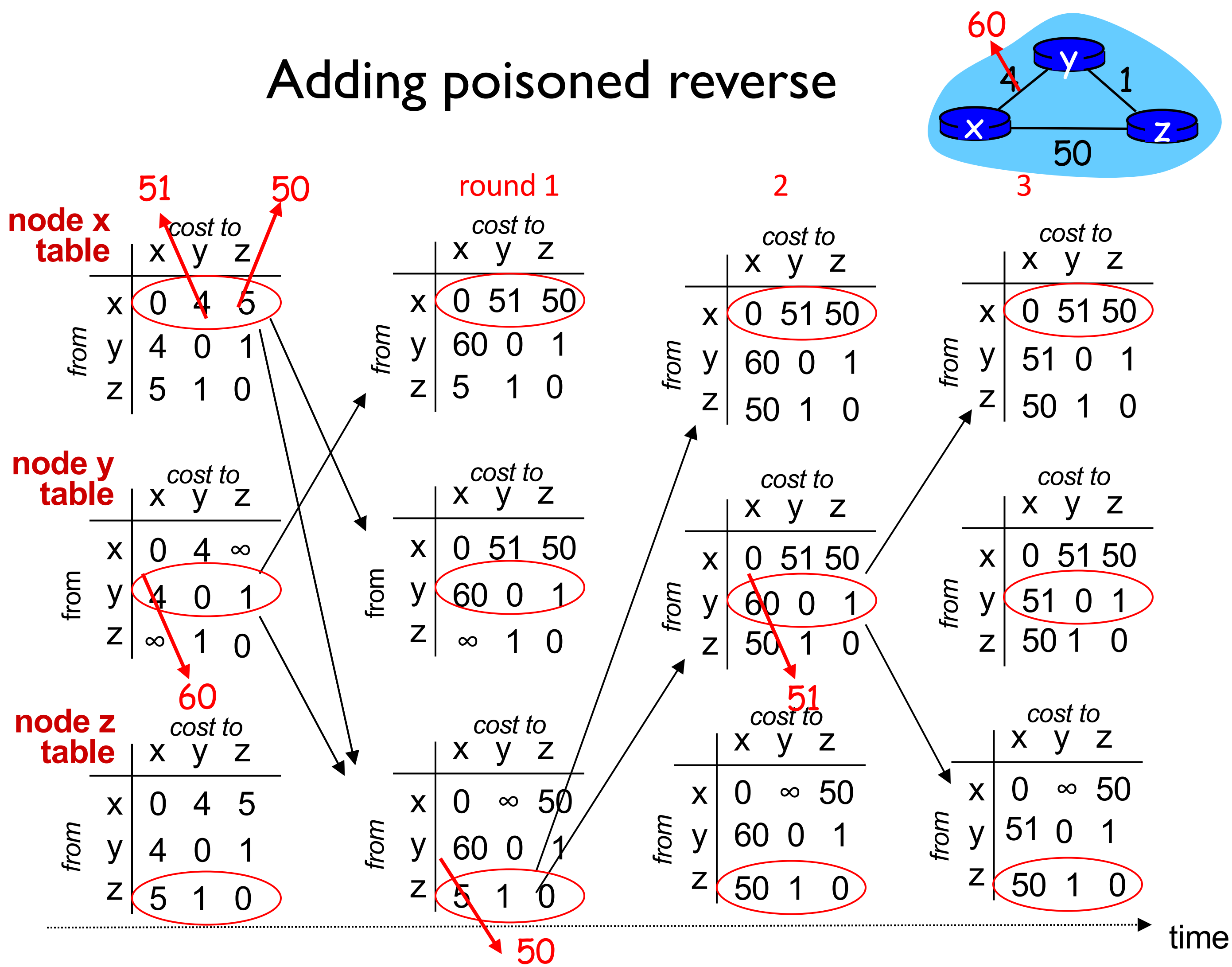
from	x	y	z
x			
y			
z			

cost to

from	x	y	z
x			
y			
z			

time

Adding poisoned reverse



Comparison of LS and DV algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***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 have 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