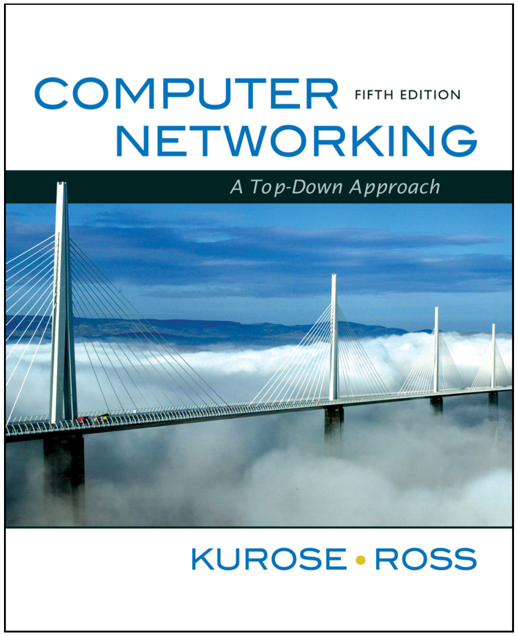
# Chapter 4
# Network Layer



## A note on the use of these ppt slides:

We' re making these slides freely available to all (faculty, students, readers). They' re in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we' d like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

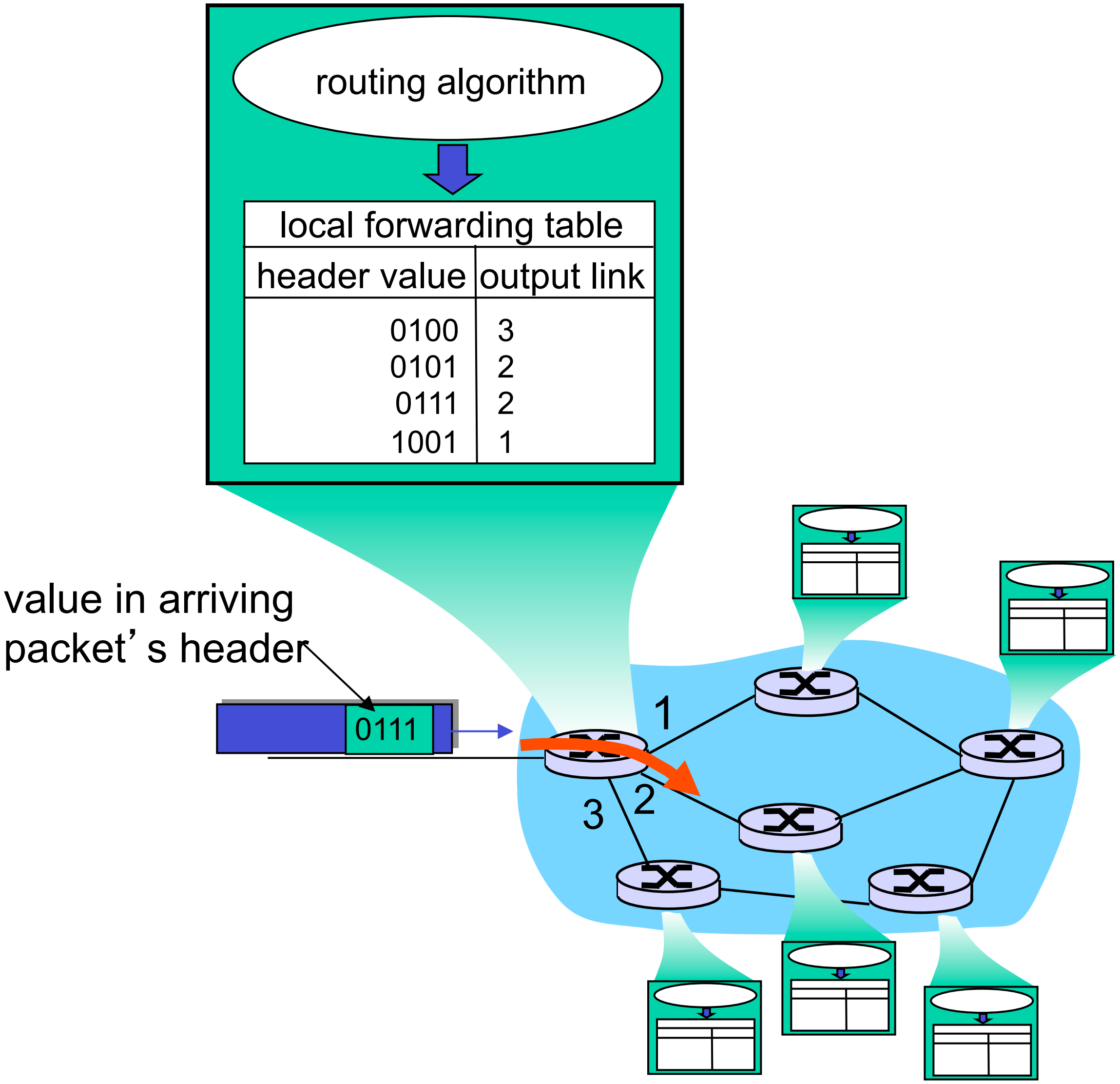
Thanks and enjoy! JFK/KWR

*Computer Networking: A Top Down Approach*
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

# Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

# Interplay between routing, forwarding



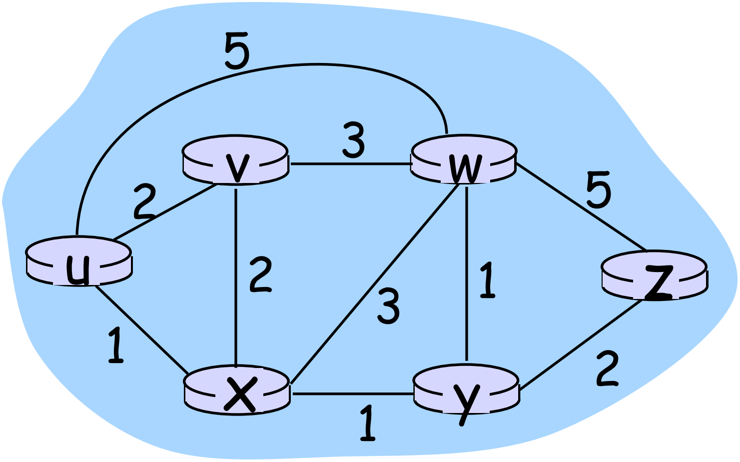
routing algorithm

local forwarding table

| header value | output link |
|---|---|
| 0100 | 3 |
| 0101 | 2 |
| 0111 | 2 |
| 1001 | 1 |

value in arriving packet's header

# Graph abstraction



Graph: G = (N,E)

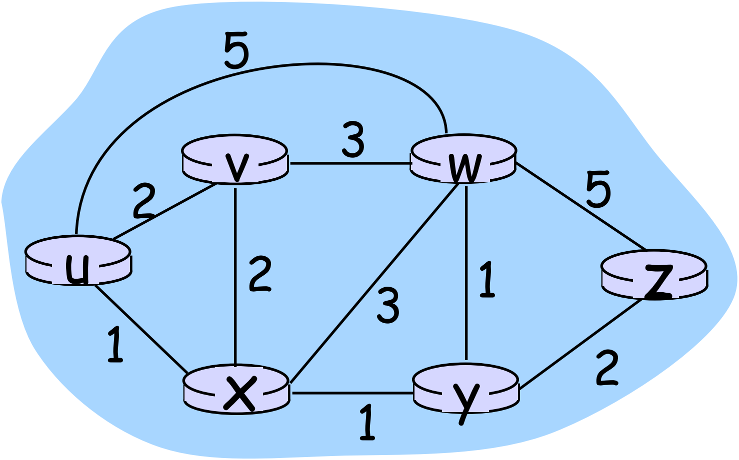
N = set of routers = { u, v, w, x, y, z }

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

Remark: Graph abstraction is useful in other network contexts

Example: 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 $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What’s the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

# 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
- "distance vector" algorithms

**Static or dynamic?**

Static:

- routes change slowly over time

Dynamic:

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

# Chapter 4: Network Layer

4. 1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

- Datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 Routing algorithms

- Link state
- Distance Vector
- Hierarchical routing

4.6 Routing in the Internet

- RIP
- OSPF
- BGP

4.7 Broadcast and multicast routing

# 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

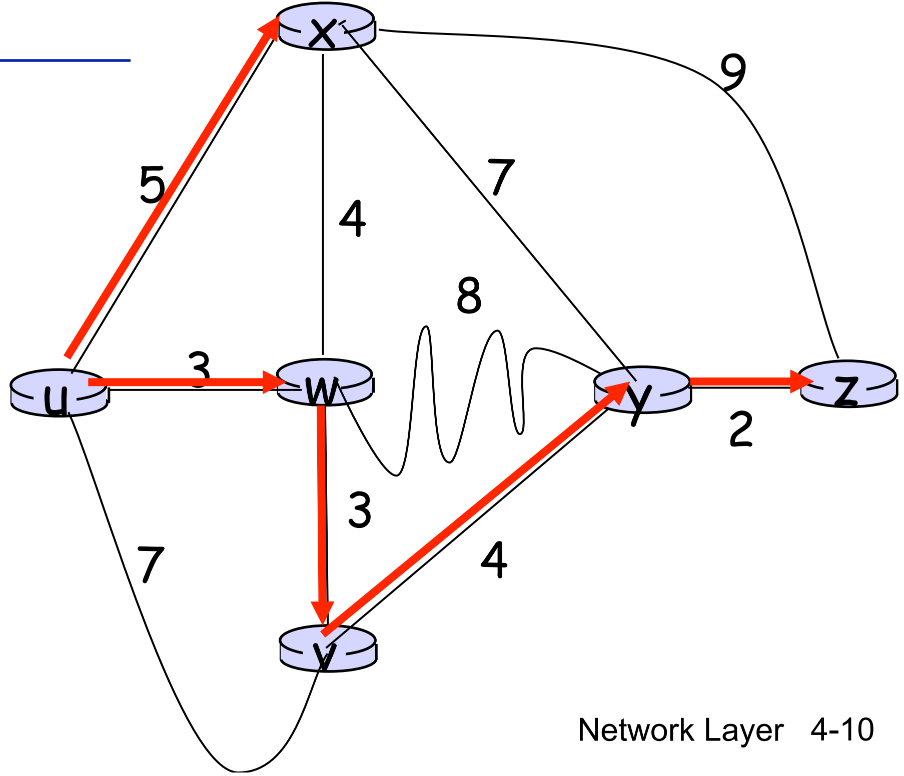
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

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

| | | D(v) | D(w) | D(x) | D(y) | D(z) |
|---|---|---|---|---|---|---|
| Step | N' | p(v) | p(w) | p(x) | p(y) | 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 | | | | | |

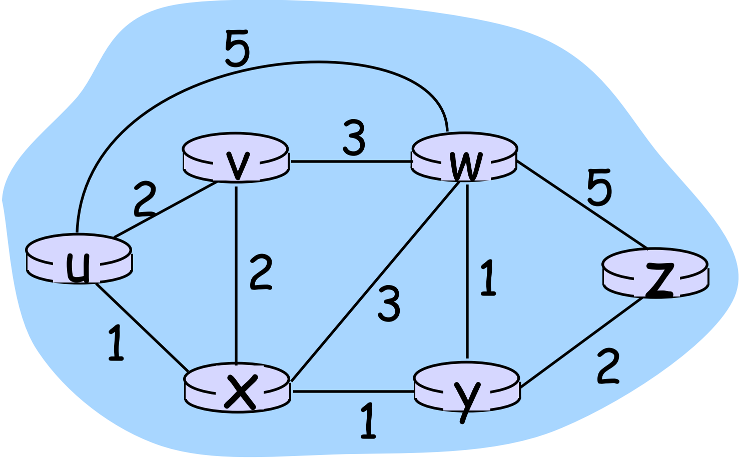
## Notes:

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



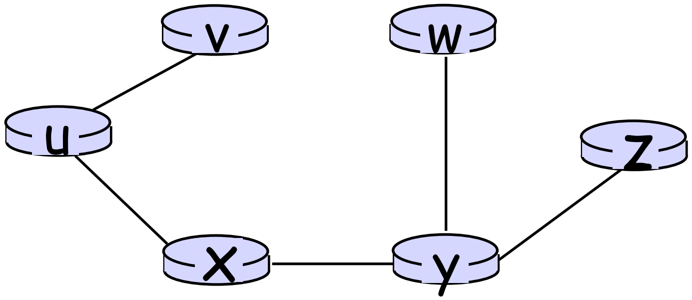
# Dijkstra's algorithm: another example

| Step | N' | D(v),p(v) | D(w),p(w) | D(x),p(x) | D(y),p(y) | D(z),p(z) |
|---|---|---|---|---|---|---|
| 0 | 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 | | | | | |



# 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) |

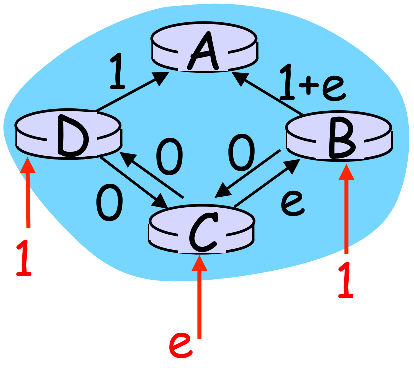
# Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

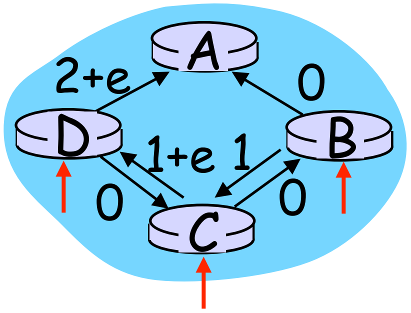
- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

Oscillations possible:

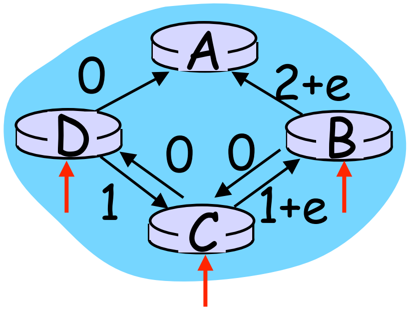
- e.g., link cost = amount of carried traffic



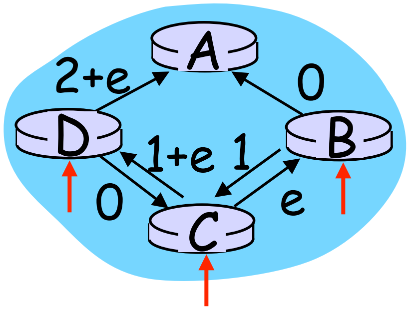
initially



… recompute routing



… recompute



… recompute

# Chapter 4: Network Layer

4. 1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What’s inside a router

4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6

4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing

4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP

4.7 Broadcast and multicast routing

# Distance Vector Algorithm

## Bellman-Ford Equation (dynamic programming)

Define

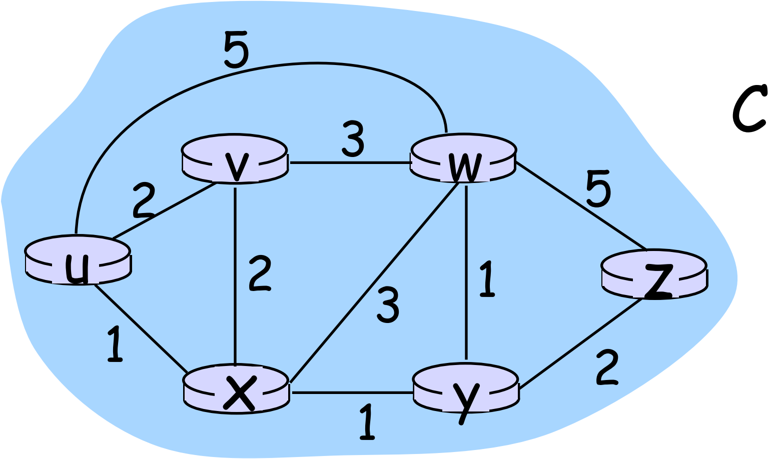
$d_x(y)$ := cost of least-cost path from x to y

Then

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

where min is taken over all neighbors v of x

# Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

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

Node that achieves minimum is next hop in shortest path ➜ 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 (4)

## Basic 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:

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

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

# Distance Vector Algorithm (5)

**Iterative, asynchronous:**

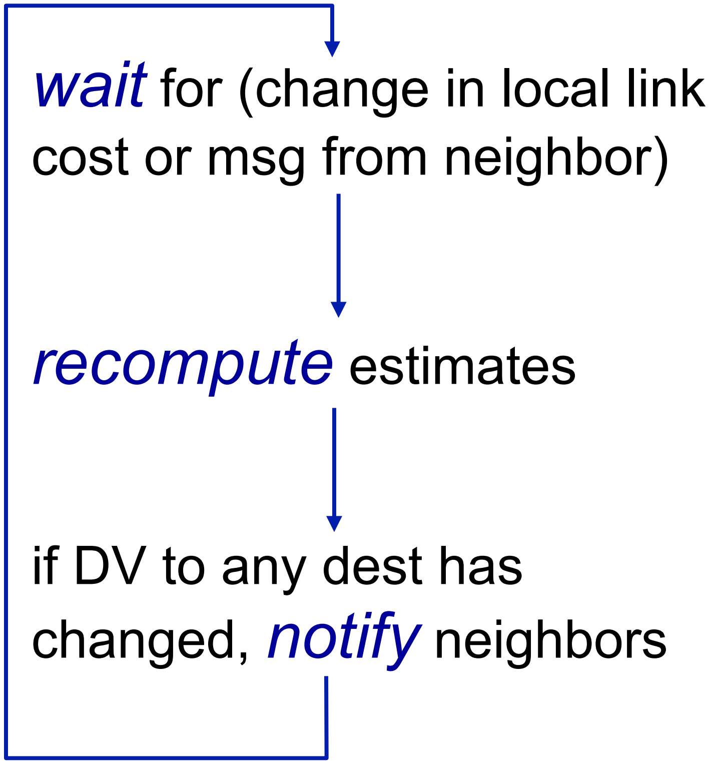
each local iteration caused by:

- 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**

cost to

from

|   | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | ∞ | ∞ | ∞ |
| z | ∞ | ∞ | ∞ |

cost to

from

|   | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

**node y table**

cost to

from

|   | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

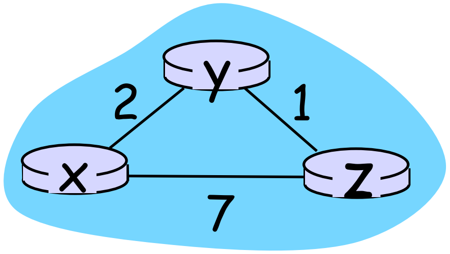
**node z table**

cost to

from

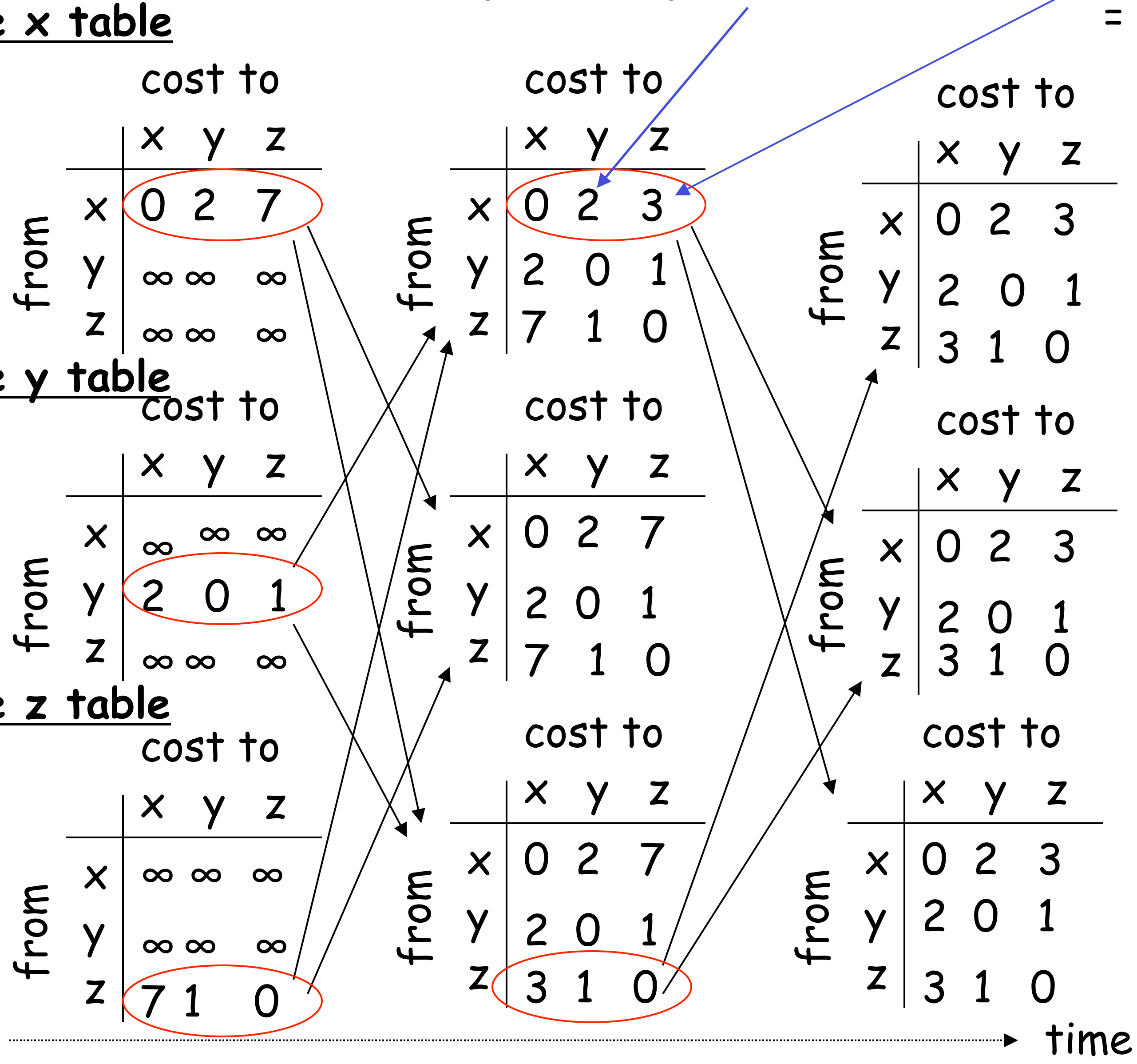
|   | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 7 | 1 | 0 |

time



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

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



**node x table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | ∞ | ∞ | ∞ |
| z | ∞ | ∞ | ∞ |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node y table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node z table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 7 | 1 | 0 |

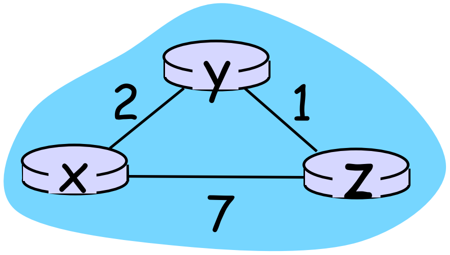
cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

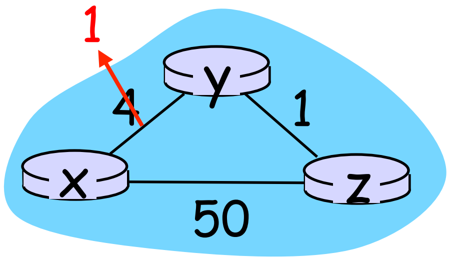
time



# 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



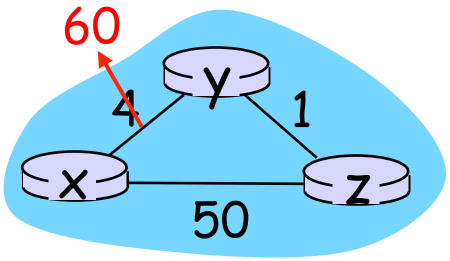
"good news travels fast"

$t_0$: *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*.

# Distance Vector: link cost changes



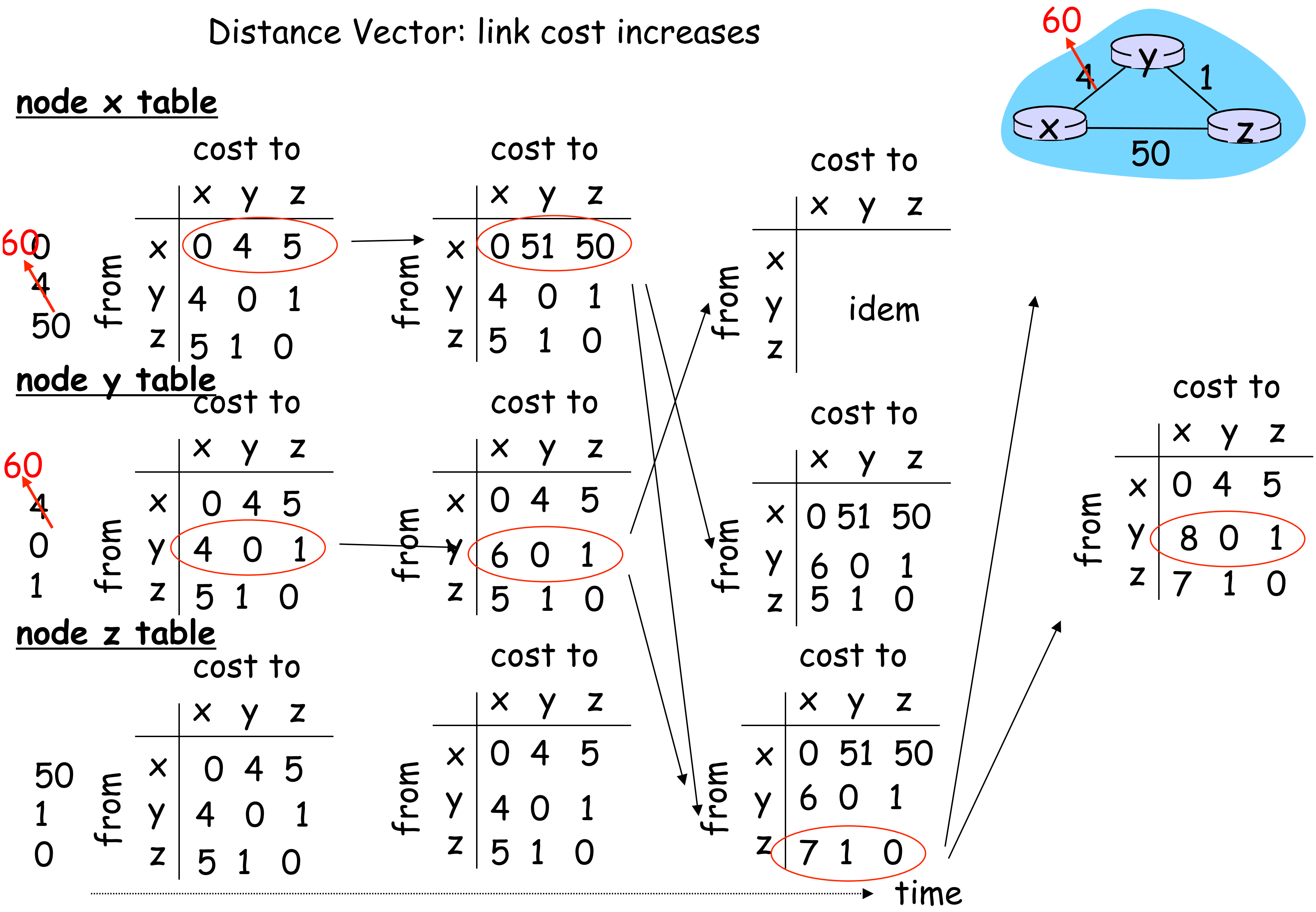
Link cost changes:

- good news travels fast
- 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?

# Distance Vector: link cost increases



**node x table**

0
~~4~~ 60
50

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 51 | 50 |
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | idem | | |
| y | | | |
| z | | | |

**node y table**

~~4~~ 60
0
1

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 6 | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 51 | 50 |
| y | 6 | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 8 | 0 | 1 |
| z | 7 | 1 | 0 |

**node z table**

50
1
0

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

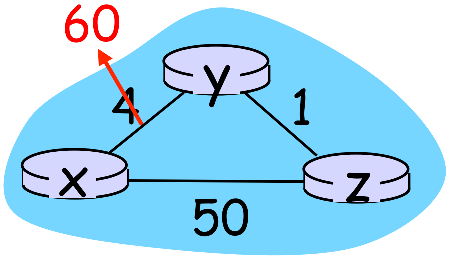
cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 51 | 50 |
| y | 6 | 0 | 1 |
| z | 7 | 1 | 0 |

# Same with poison reverse!



**node x table**

0
~~4~~ 60
50

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | ∞ | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 51 | 50 |
| y | ∞ | 0 | 1 |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | | | |
| y | | idem | |
| z | | | |

**node y table**

~~4~~ 60
0
1

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | ∞ | ∞ |
| y | 4 | 0 | 1 |
| z | ∞ | ∞ | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | ∞ | ∞ |
| y | 60 | 0 | 1 |
| z | ∞ | ∞ | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | | | |
| y | | idem | |
| z | | | |

**node z table**

50
1
0

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 4 | 0 | ∞ |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 5 |
| y | 4 | 0 | ∞ |
| z | 5 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | ∞ | ∞ |
| y | 60 | 0 | ∞ |
| z | 50 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 51 | 50 |
| y | 51 | 0 | 1 |
| z | 50 | ∞ | 0 |

time

# 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 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