COMP 3234B
Computer and Communication Networks

2nd semester 2023-2024
Network Layer (III)

Prof. C Wu

Department of Computer Science
The University of Hong Kong

Roadmap

Network layer

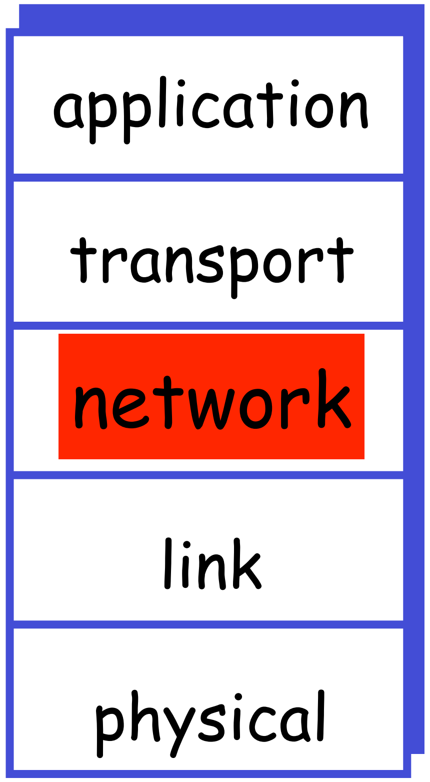
- Principles behind network-layer services (ILO1)

  forwarding vs. routing

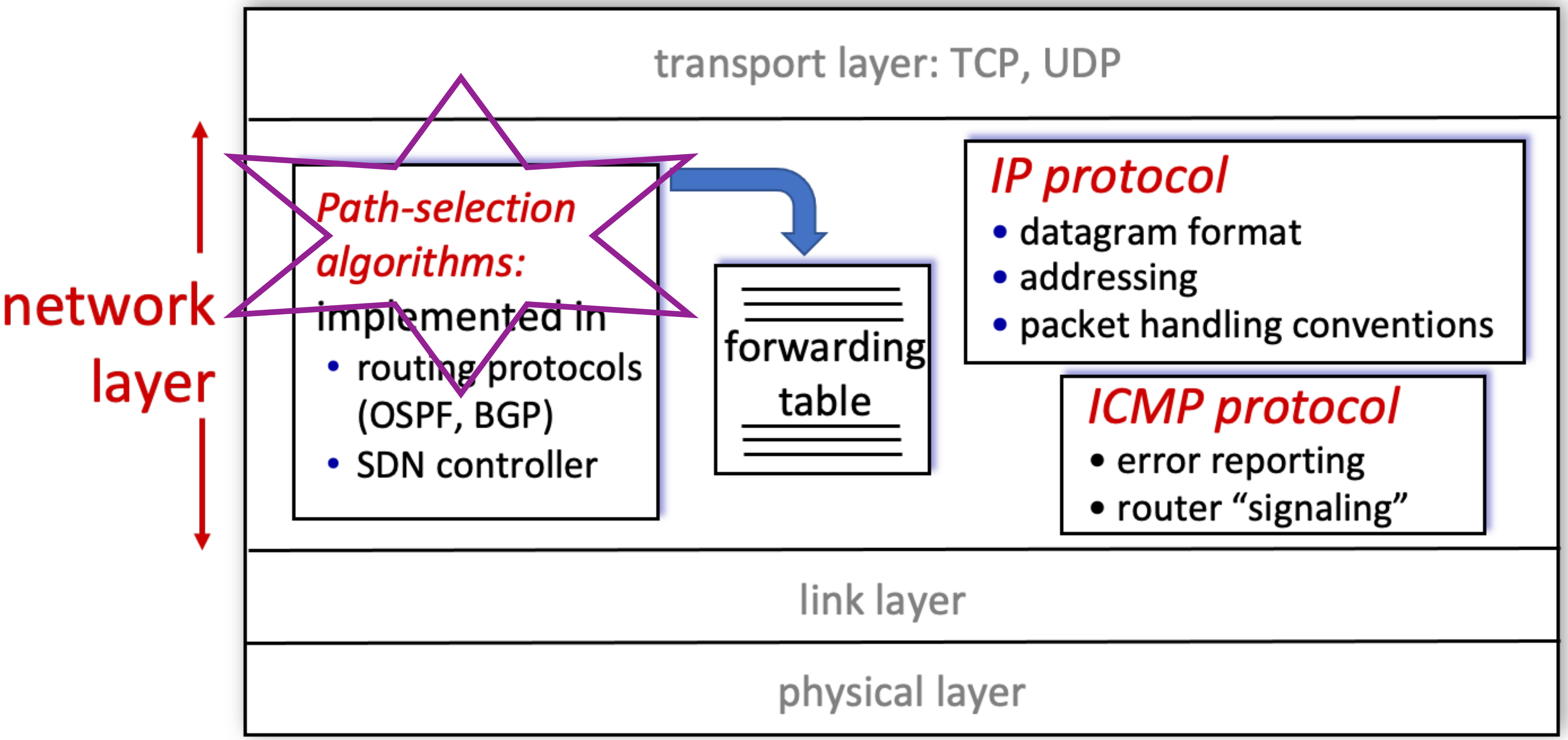
  network service models
- Router (ILO1)
- IP (ILO2,5)

  DHCP

  NAT
- ICMP(ILO2)
- Routing algorithms (ILO3)
- Routing in the Internet (ILO2,3)

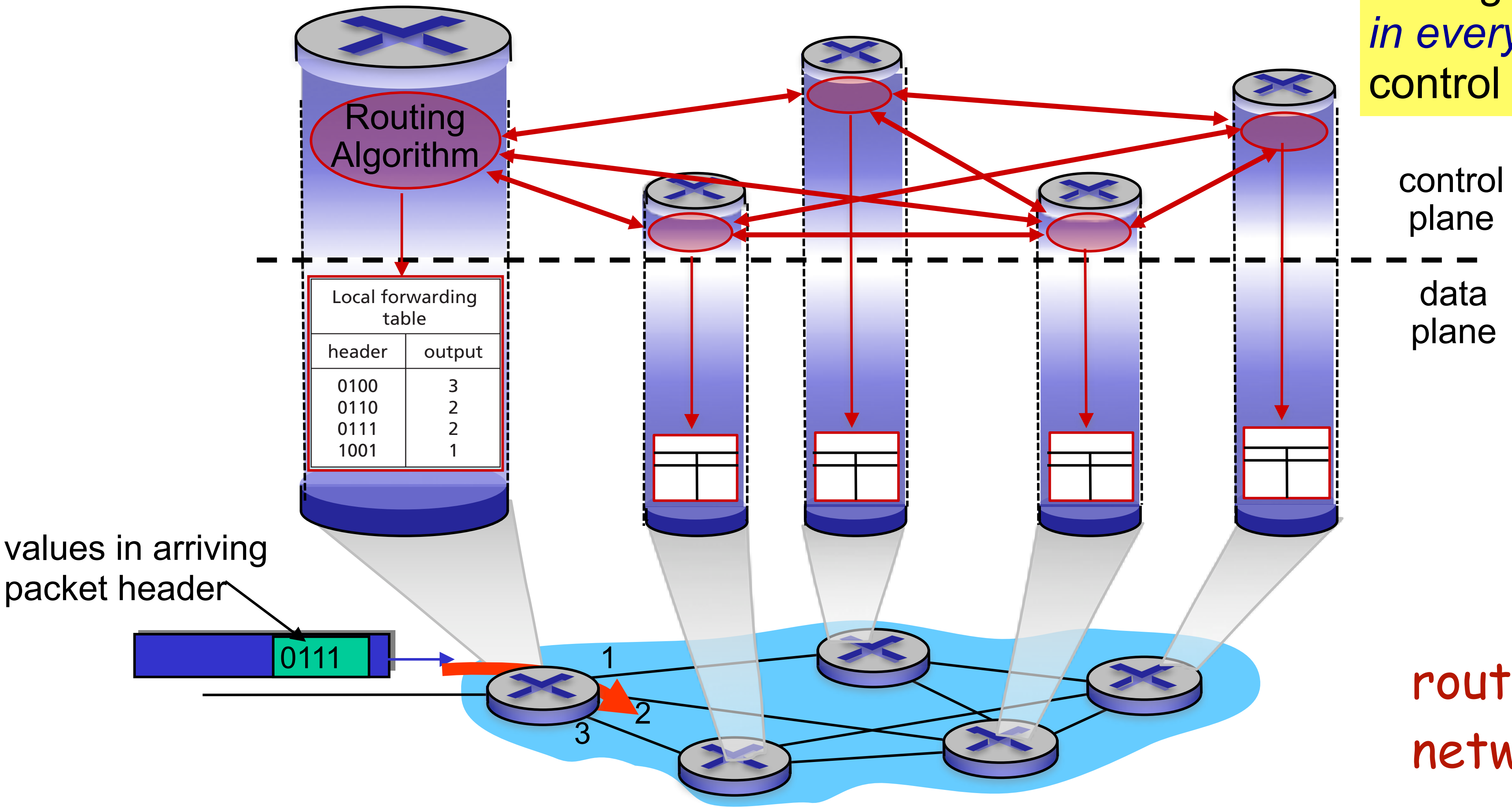


Internet network layer



Routing protocols

- Decide the *good* paths (aka routes) taken by datagrams from sending host to receiving host, through the network of routers
  - decide the forwarding tables at routers
  - path: sequence of routers that datagrams traverse, going from source host to destination host
  - "good": "least cost", "fastest", "least congested"



Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

routing algorithm components *in every router* interact in the control plane

routing: a "top-10" networking problem

Graph abstraction of the network

Graph abstraction

- Undirected graph: G = (N,E)

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

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

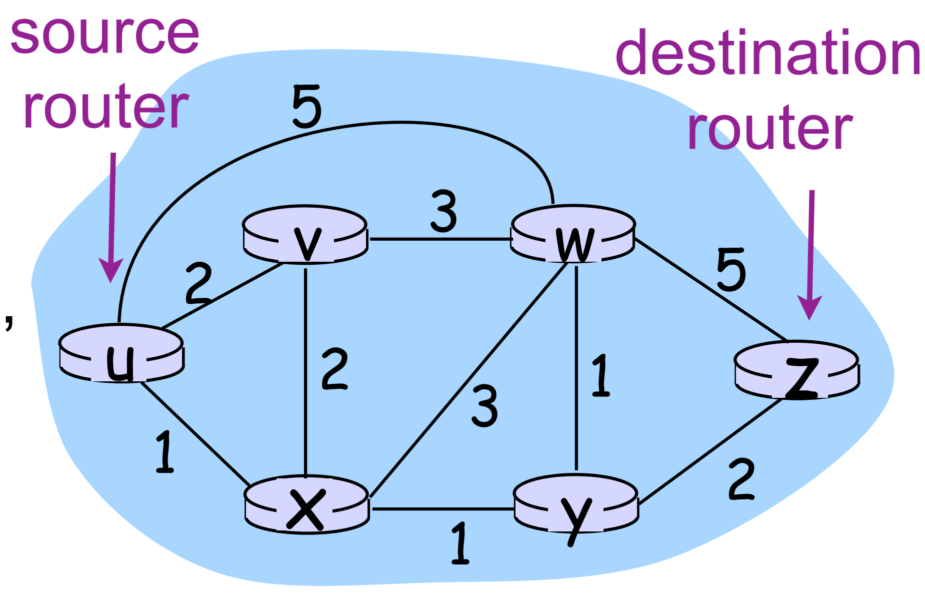
- Link cost: the value associated with a link (related to bandwidth, congestion, physical length, delay, etc.)

  c(x_1, x_2) = cost of link (x_1, x_2), e.g., c(w,z) = 5

  c(x_1, x_2) = ∞ if (x_1, x_2) does not belong to E

- Path: a sequence of nodes, (x_1, x_2, x_3,…, x_p)

- Cost of path (x_1, x_2, x_3,…, x_p): c(x_1,x_2) + c(x_2,x_3) + … + c(x_{p-1},x_p)



source router

destination router

What's the least-cost path between u and z ?

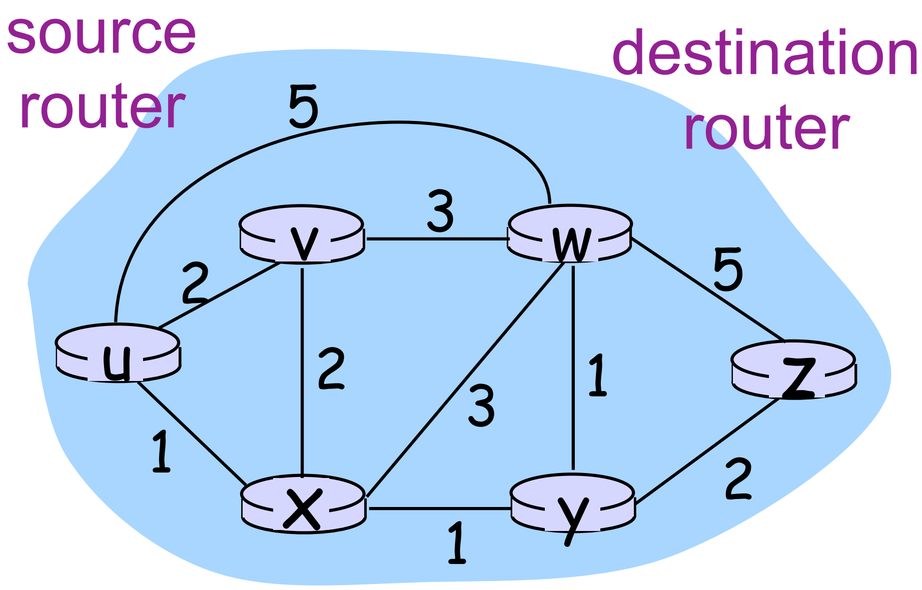
Routing algorithm

- Given a set of routers with links connecting the routers, finds a least-cost path from source router to destination router

  (finds the shortest path when link costs represent length)

Routing algorithm classification

- ☐ Centralized or decentralized routing algorithm



- Centralized:

  algorithm input: complete global knowledge about the network, including node connectivity (topology), link costs

  carried out at each router

  link-state (LS) algorithms

- Decentralized:

  with (local) knowledge of connectivity to neighbor routers, link costs to neighbors

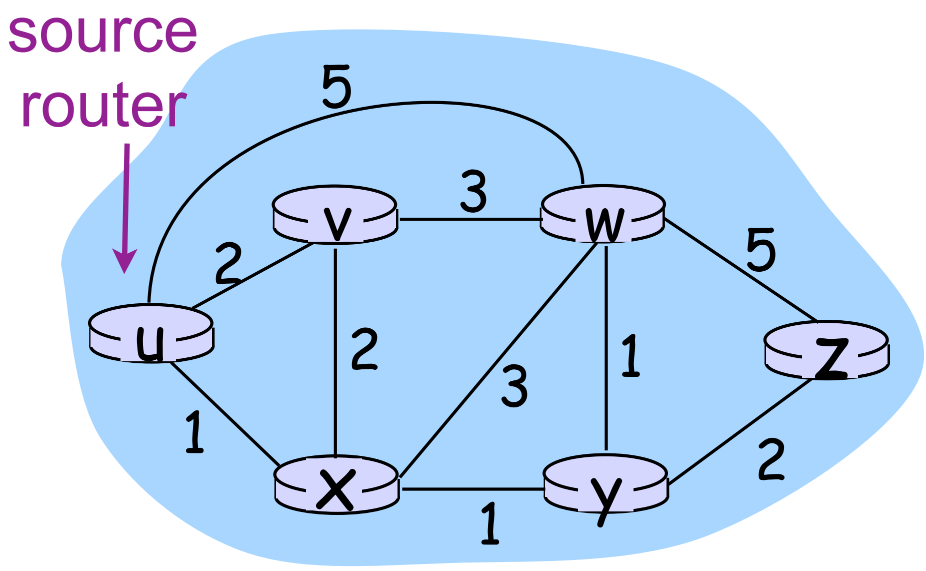
  algorithm carried out at each router by *iterative* process of computation and exchange of information with neighbor

  distance-vector (DV) algorithms

Link-state routing algorithm

- Dijkstra's algorithm
  - Input: network topology, link costs of all links
  - Output: least-cost paths from one node ("source router") to all other nodes (routers)

    derives forwarding table for that router
  - Iterative algorithm executed at one node: after k iterations, the source derives least-cost paths to k destinations with the smallest path costs



each node broadcasts its identity and costs of neighboring links to all other nodes in the network ("link state broadcast")

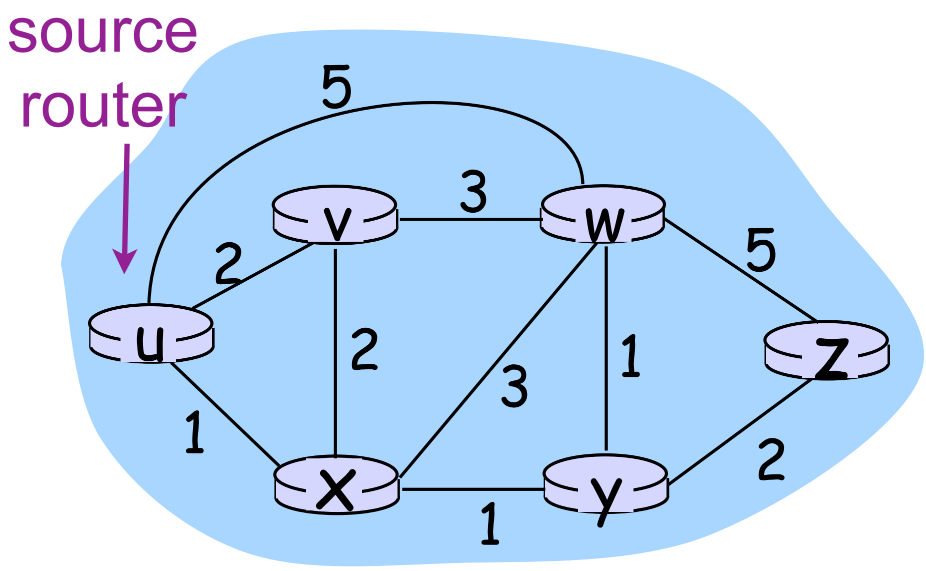
=>all nodes have same and complete view of the network

Dijkstra's algorithm

```
Initialization:
  N' = {u}
  for all nodes i
    if i is a neighbor of u
      then $D(i) = c(u,i)$, $p(i) = u$
    else $D(i) = \infty$

Loop
  find j not in N' such that D(j) is minimum
  add j to N'
  update D(i) for each neighbor i of j and not in
  N' :
      $D(i) = \min( D(i), D(j) + c(j,i) )$; update p(i)
  /* new cost to i is either old cost to i or known
   least path cost to j plus cost from j to i */
until N' = N
```

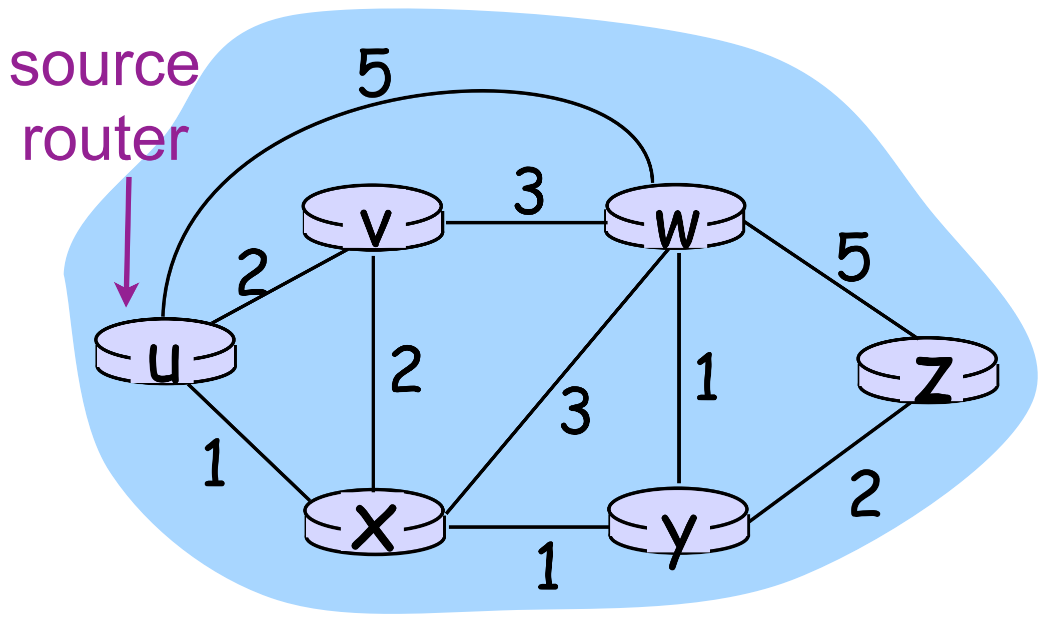
# of loops=
# of nodes in the network
excluding the source

- 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 destination v
  - p(v): predecessor node along the current least-cost path from source to v
  - N': set of nodes whose least-cost path already known



An example of Dijkstra’s algorithm

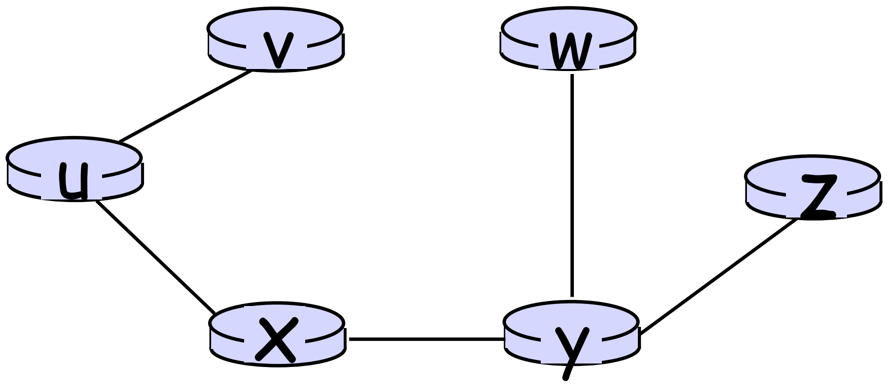
- Example



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

An example of Dijkstra's algorithm (cont'd)

Resulting least-cost-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)

next-hop node along the least-cost path towards the destination

Discussions on Dijkstra's algorithm

- Algorithm complexity

  with n nodes (routers excluding the source)

  - 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)

Discussions on Dijkstra's algorithm (cont'd)

- Potential problem: routing oscillations
  - Example scenario:

    link cost = amount of carried traffic (reflecting delay on the link)

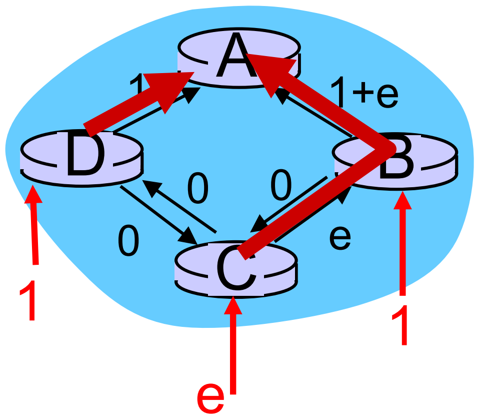
    link costs could be asymmetric: c(u,v)=c(v,u) only if traffic on both directions is the same

    node D originates a unit of traffic destined to A;

    node B originates a unit of traffic destined to A;

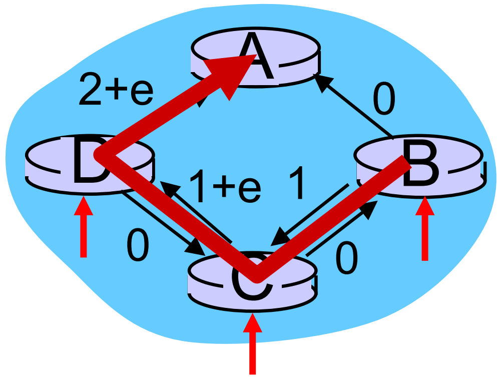
    node C originates traffic of amount e to A.

  - Possible solution: have each router run the algorithm at different times

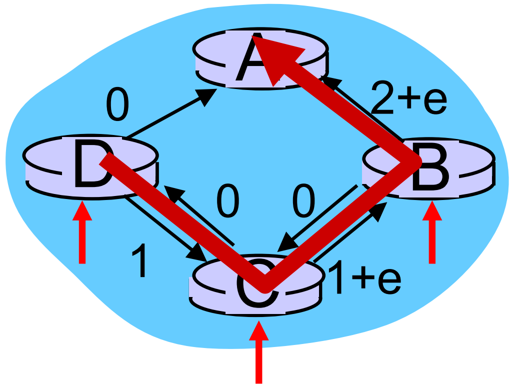


initially

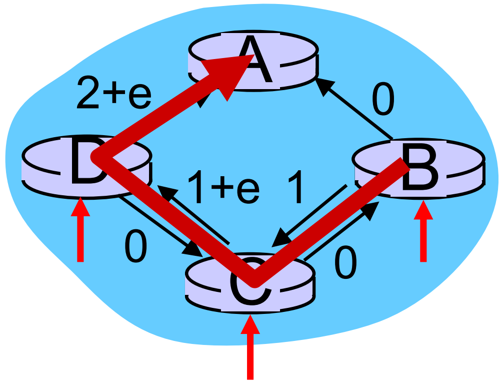
(new costs shown after initial routing)



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs

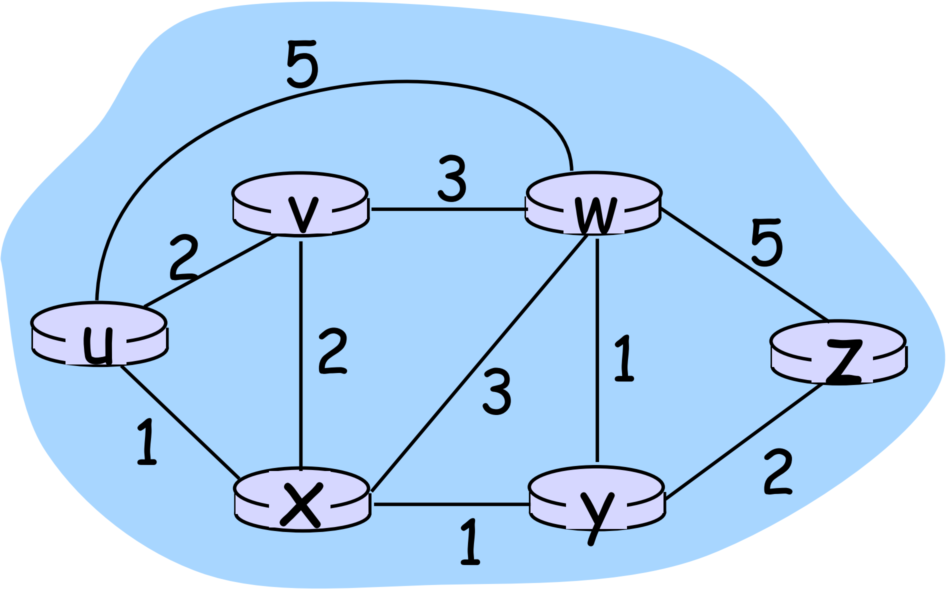
Distance-vector routing algorithm

- Bellman-Ford algorithm
  - Input: connectivity/link costs to neighbors
  - Output: least-cost paths from each node to all other nodes

    derives forwarding table for each router
  - Iterative, asynchronous, distributed algorithm executed by all nodes together:

    each node receives updates from neighbors, recomputes, and distributes its new calculation result to neighbors

    algorithm terminates (least-cost path from each node to each other node derived) when no more update is exchanged between neighbors



each node updates information to neighbors only

Bellman-Ford equation

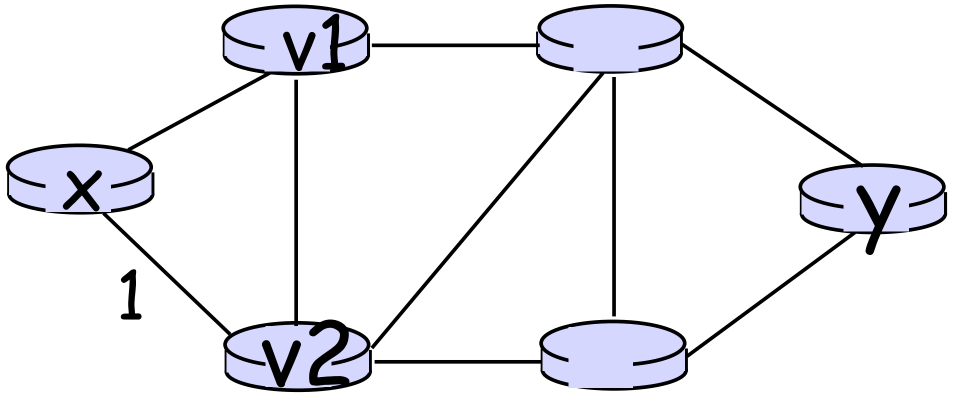
- Bellman-Ford equation
  - Define an important relationship among the costs of least-cost paths

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

Then

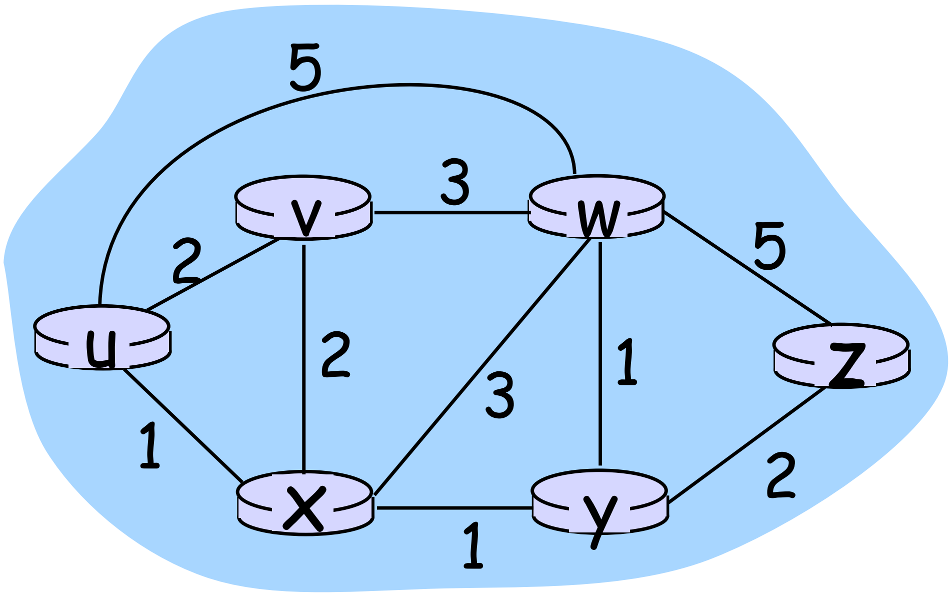
$$\boxed{d_x(y) = \min_v \{c(x,v) + d_v(y) \}}$$

where min is taken over all neighbors v of x



Bellman-Ford equation (cont'd)

- Example



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

Bellman-Ford equation says:

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

$$\begin{aligned} = \min\{ & 2 + 5, \\ & 1 + 3, \\ & 5 + 3\} = 4 \end{aligned}$$

Node that achieves minimum is next hop in the least-cost path
➜ decides the entry in u's forwarding table

Bellman-Ford algorithm basics

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

  For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

- Basic idea of Bellman-Ford algorithm
  - From time-to-time, each node sends its updated distance vector (DV) to neighbors
  - When a node x receives new DV estimate from neighbor v, it updates its own DV using Bellman-Ford equation:

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

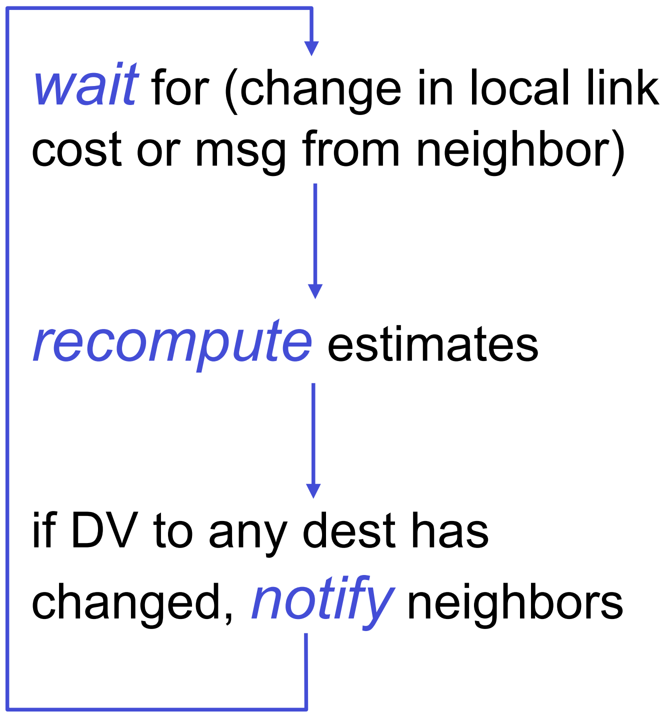
  - all nodes continue to exchange their DVs in an asynchronous fashion; each least cost estimate $D_x(y)$ converges to the actual least cost $d_x(y)$

Bellman-Ford algorithm

At each node, x:

```
Initialization:
  for all destinations y in N:
    $D_x(y) = c(x,y)$
  for each neighbor w
    $D_w(y) = \infty$ for all destinations y in N
  for each neighbor w
    send distance vector $\mathbf{D}_x = [D_x(y): y \in N]$ to w

Loop
  wait (until I see a link cost change to some neighbor w or
        until I receive a distance vector from some neighbor w)

  for each y in N:
    $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$

  if $D_x(y)$ changed for any destination y
    send distance vector $\mathbf{D}_x = [D_x(y): y \in N]$ to all neighbors

forever
```

Each node:



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

recompute estimates

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

An example of Bellman-Ford algorithm

- Example

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

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

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	∞	∞	∞

cost to

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

node z table

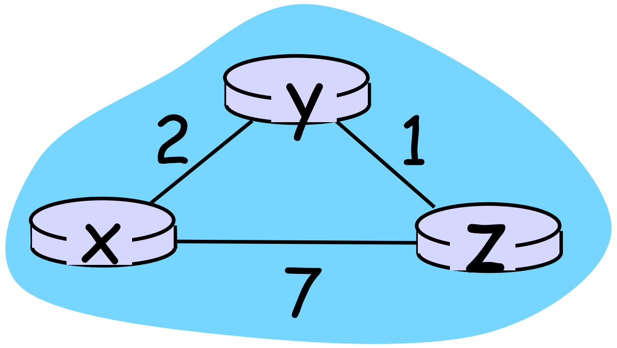
cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

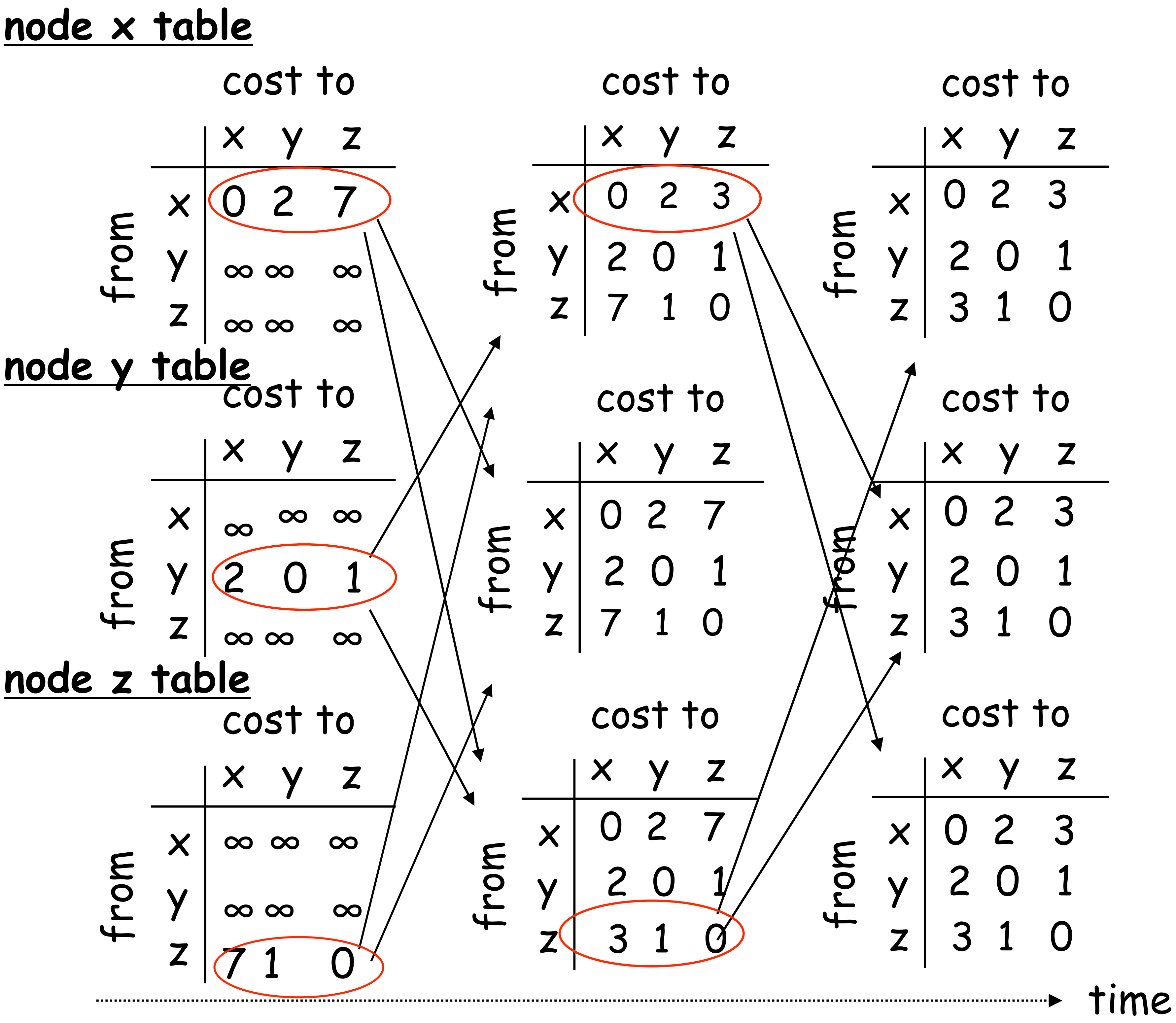
cost to

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

time



An example of Bellman-Ford algorithm (cont'd)



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

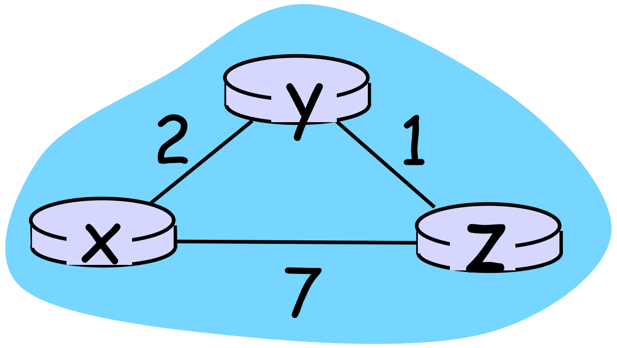
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

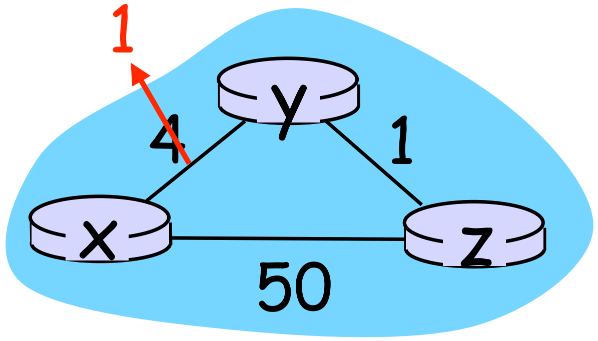
from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

cost to

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

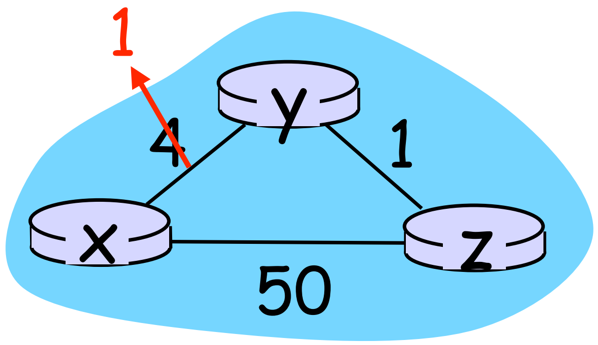


Discussions on Bellman-Ford algorithm I



- Link cost change: scenario 1
  - At time t_0, x and y detect the link-cost change (4 → 1), updates their DV, and inform neighbors.

DV table evolution



node x table

		cost to		
		x	y	z
from	x	0	~~4~~ 1	~~5~~ 2
	y	4	0	1
	z	5	1	0

node y table

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

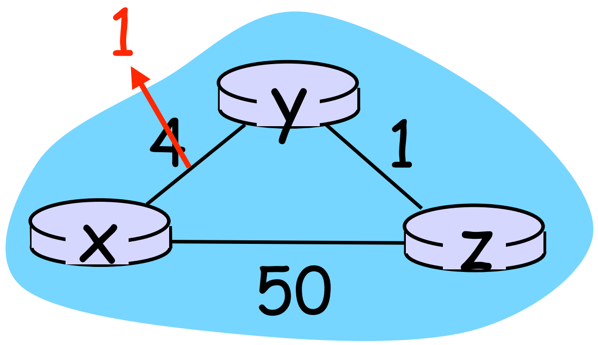
node z table

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

time

t_0 t_1 t_2

DV table evolution



node x table

		cost to		
		x	y	z
from	x	0	~~4~~ 1	~~5~~ 2
	y	4	0	1
	z	5	1	0

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

node y table

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

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

node z table

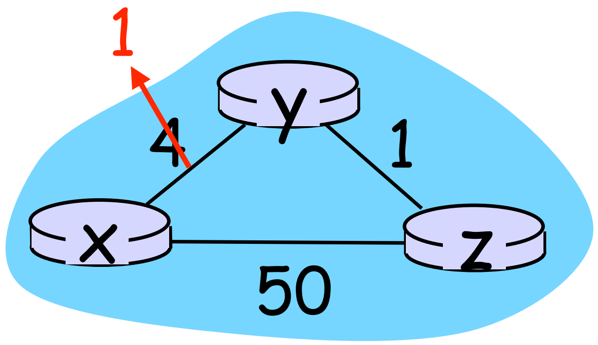
		cost to		
		x	y	z
from	x	0	4	5
	y	4	0	1
	z	5	1	0

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

time

t_0 t_1 t_2

DV table evolution



node x table

cost to

from	x	y	z
x	0	~~4~~ 1	~~5~~ 2
y	4	0	1
z	5	1	0

cost to

from	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

cost to

from	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

node y table

cost to

from	x	y	z
x	0	4	5
y	~~4~~ 1	0	1
z	5	1	0

cost to

from	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

cost to

from	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

node z table

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	1	2
y	1	0	1
z	2	1	0

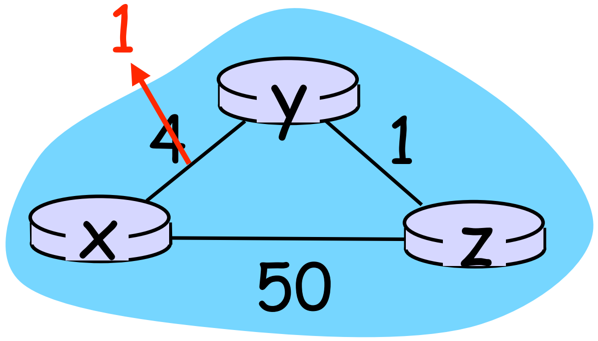
cost to

from	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

time

t_0 t_1 t_2

Discussions on Bellman-Ford algorithm I (cont'd)

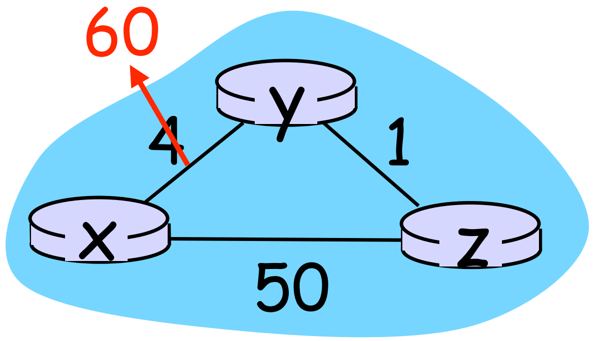


- Link cost change: scenario 1
  - At time t_0, x and y detect the link-cost change (4 → 1), updates their DV, and inform neighbors.
  - At time t_1, z receives the updates, computes a new least cost to x (5->2) and informs its neighbors.
  - At time t_2, x and y receive z's update and update their DV tables. x and y's least costs do not change and hence do not send any further message to z.

Good news travels fast!

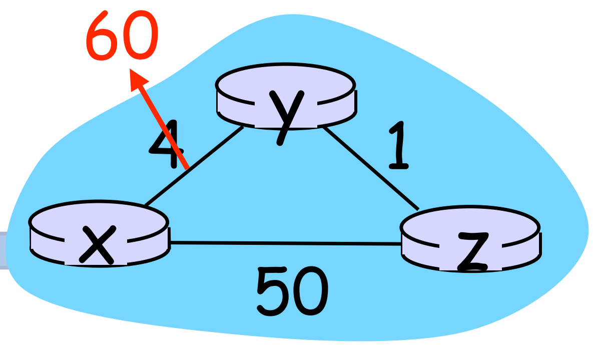
Discussions on Bellman-Ford algorithm II

- Link cost change: scenario 2
  - At time t_0, x and y detect the link-cost change (4 → 60), update their DV, and notify neighbors



$$D_x(y)=\min\{c(x,y)+D_y(y),\ c(x,z)+D_z(y)\}=\min\{60+0,50+1\}=51$$

$$D_y(x)=\min\{c(y,x)+D_x(x),\ c(y,z)+D_z(x)\}=\min\{60+0,1+5\}=6$$



node x table

cost to

from	x	y	z
x	0	~~4~~ 51	~~5~~ 50
y	4	0	1
z	5	1	0

node y table

cost to

from	x	y	z
x	0	4	5
y	~~4~~ 6	0	1
z	5	1	0

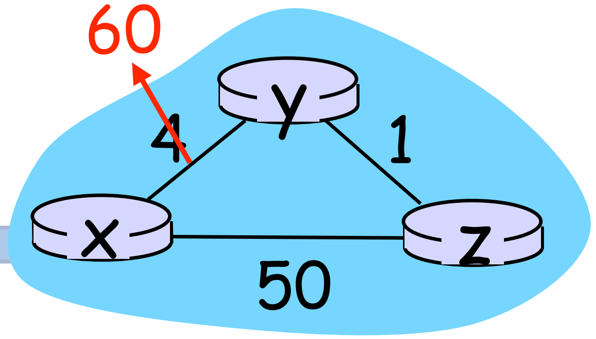
node z table

cost to

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

t_0 t_1 t_2 t_3 t_4 time

$$D_z(x)=\min\{c(z,x)+D_x(x),\ c(z,y)+D_y(x)\}=\min\{50+0,1+6\}=7$$



node x table

cost to

from	x	y	z
x	0	4 → 51	5 → 50
y	4	0	1
z	5	1	0

cost to

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

node y table

cost to

from	x	y	z
x	0	4	5
y	4 → 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

node z table

cost to

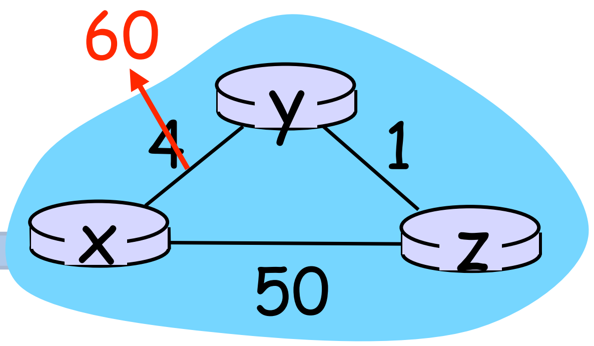
from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

cost to

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

t_0 t_1 t_2 t_3 t_4 time

$$D_y(x)=\min\{c(y,x)+D_x(x),\ c(y,z)+D_z(x)\} =\min\{60+0,1+7\}=8$$



node x table

	cost to		
from	x	y	z
x	0	~~4~~ 51	~~5~~ 50
y	4	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	51	50
y	6	0	1
z	7	1	0

node y table

	cost to		
from	x	y	z
x	0	4	5
y	~~4~~ 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	51	50
y	8	0	1
z	7	1	0

node z table

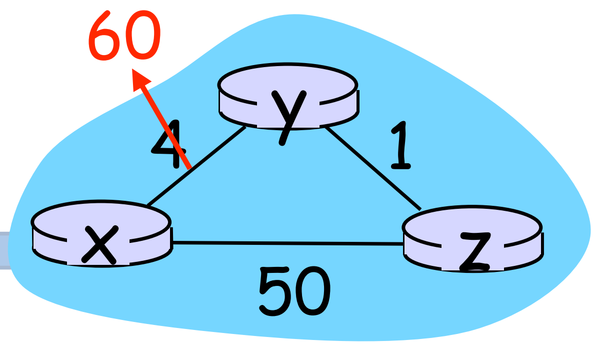
	cost to		
from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

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

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

t_0 t_1 t_2 t_3 t_4 time

$D_z(x)=\min\{c(z,x)+D_x(x),$
$c(z,y)+D_y(x)\}$
$=\min\{50+0,1+8\}=9$



node x table

cost to

from	x	y	z
x	0	~~4~~ 51	~~5~~ 50
y	4	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	51	50
y	6	0	1
z	7	1	0

cost to

from	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

node y table

cost to

from	x	y	z
x	0	4	5
y	~~4~~ 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	51	50
y	8	0	1
z	7	1	0

cost to

from	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

node z table

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

cost to

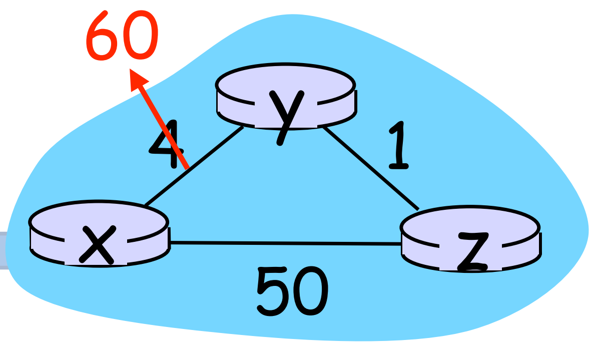
from	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

cost to

from	x	y	z
x	0	51	50
y	8	0	1
z	9	1	0

t_0 t_1 t_2 t_3 t_4 time

$$D_y(x)=\min\{c(y,x)+D_x(x),\ c(y,z)+D_z(x)\} =\min\{60+0,1+9\}=10$$



node x table

cost to

from	x	y	z
x	0	4 → 51	5 → 50
y	4	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	51	50
y	6	0	1
z	7	1	0

cost to

from	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

cost to

from	x	y	z
x	0	51	50
y	8	0	1
z	9	1	0

node y table

cost to

from	x	y	z
x	0	4	5
y	4 → 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	51	50
y	8	0	1
z	7	1	0

cost to

from	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

cost to

from	x	y	z
x	0	51	50
y	10	0	1
z	9	1	0

....

node z table

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

cost to

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

cost to

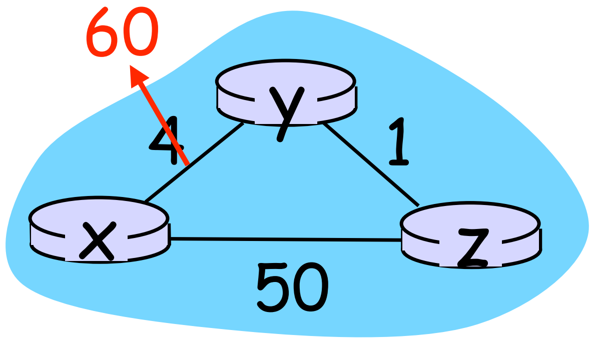
from	x	y	z
x	0	51	50
y	8	0	1
z	9	1	0

cost to

from	x	y	z
x	0	51	50
y	8	0	1
z	9	1	0

t_0 t_1 t_2 t_3 t_4 time

Discussions on Bellman-Ford algorithm II (cont'd)



- Link cost change: scenario 2
  - At time t_0, x and y detect the link-cost change (4 → 60), update their DV, and notify neighbors
  - At time t_1, z receives updated DVs, computes a new least cost to x of $D_z(x) = \min\{50+0, 1+6\} = 7$, and informs neighbors of its new DV
  - At time t_2, y receives z's update, recomputes $D_y(x) = 8$, and sends it to neighbors
  - At time t_3, z receives y's update, recomputes $D_z(x) = 9$, and sends it to neighbors
  - ...

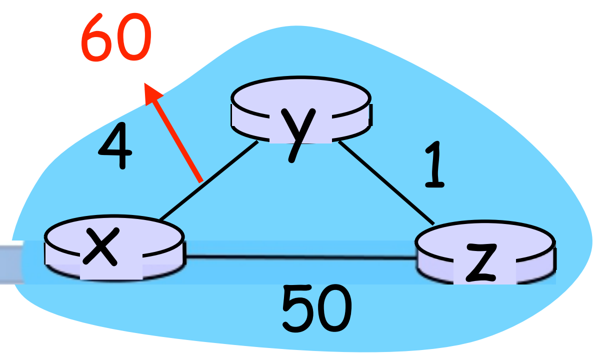
44 iterations before algorithm terminates!

Bad news travels slowly => "count to infinity" problem!

But it does not solve general count-to-infinity problem

The problem in this scenario can be avoided by 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)

Poisoned reverse



node x table

t_0

cost to

from	x	y	z
x	0	~~4~~ 51	~~5~~ 50
y	4	0	1
z	5	1	0

t_1

cost to

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

t_2

cost to

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

t_3

cost to

from	x	y	z
x	0	51	50
y	60	0	1
z	7	1	0

t_4

cost to

from	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

t_5

cost to

from	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node y table

t_0

cost to

from	x	y	z
x	0	4	5
y	~~4~~ 6	0	1
z	5	1	0

t_1

cost to

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

t_2

cost to

from	x	y	z
x	0	51	50
y	60	0	1
z	∞	1	0

t_3

cost to

from	x	y	z
x	0	51	50
y	60	0	1
z	∞	1	0

t_4

cost to

from	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

t_5

cost to

from	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node z table

t_0

cost to

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

t_1

cost to

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

t_2

cost to

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

t_3

cost to

from	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

t_4

cost to

from	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

t_5

cost to

from	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

z routes to x through y; suppose it tells y that Dz(x) = ∞ in next round

Comparison of LS and DV algorithms

Message complexity

LS: with n nodes, $O(n^2)$ msgs

DV: exchange between neighbors only; convergence time varies

Convergence

LS: $O(n^2)$ algorithm requires $O(n^2)$ msgs; may have oscillations

DV: convergence time varies; may existing routing loops (count-to-infinity problem)

Both algorithms used in routing protocols in the Internet

LS algorithms used in OSPF, IS-IS.

DV algorithms used in RIP, IGRP.

- Required reading:
  - *Computer Networking: A Top Down Approach* (8th Edition)
    Ch 5.2

- Acknowledgement:
  - Some materials are extracted from the slides created by Prof. Jim F. Kurose and Prof. Keith W. Ross for the textbook.