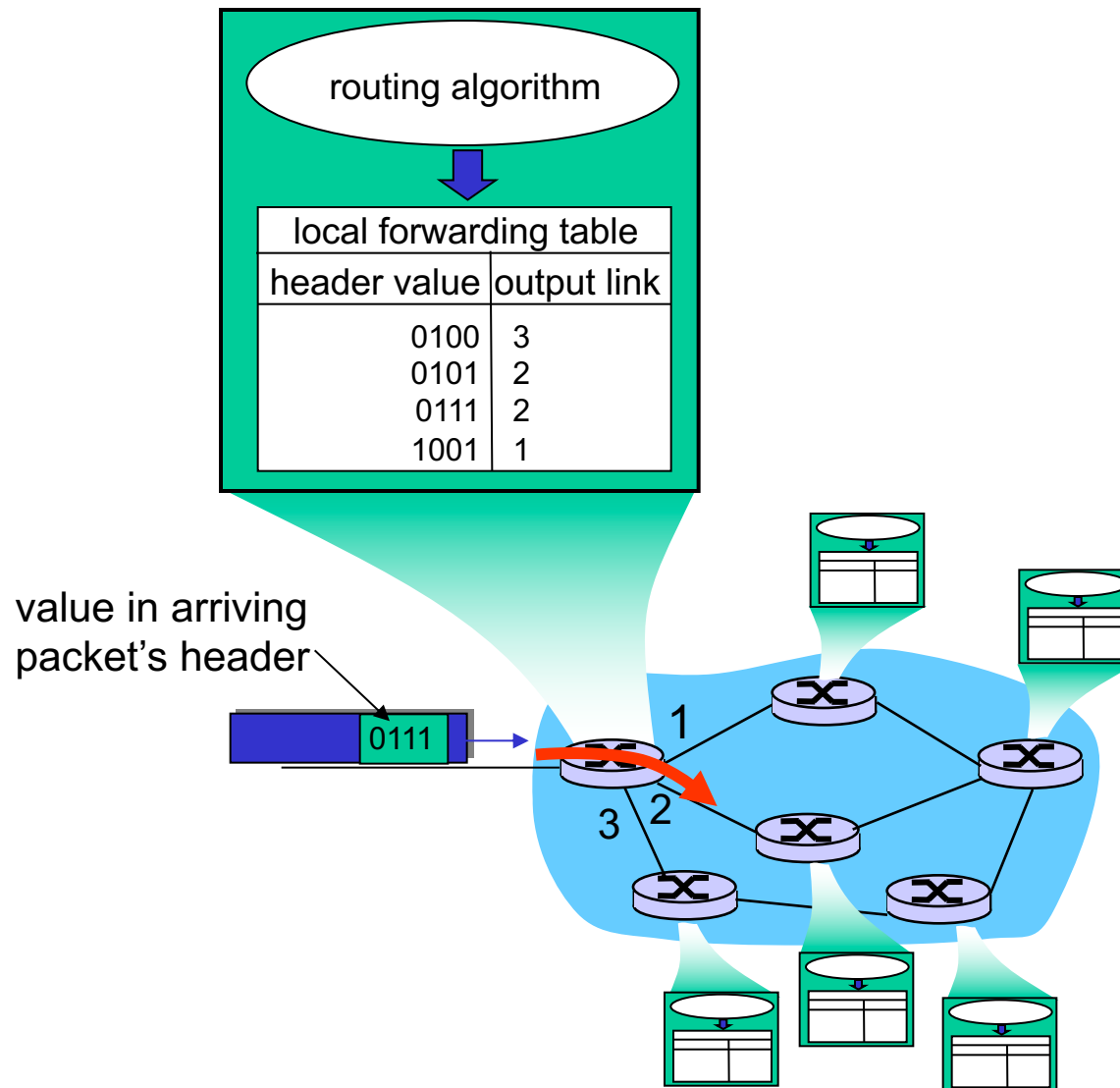


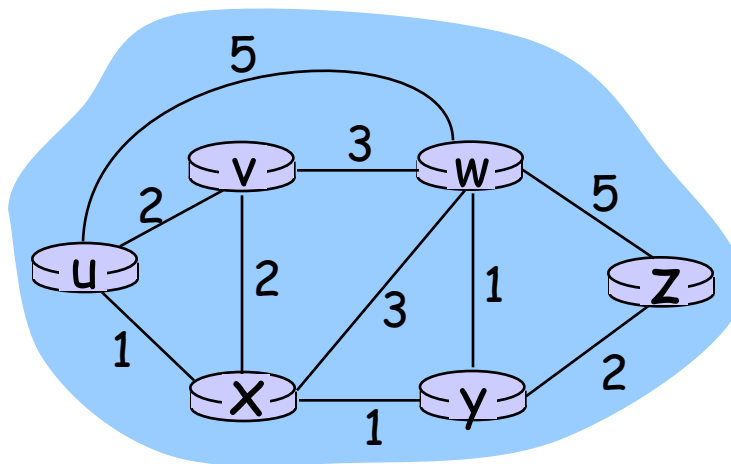
Chapter 4: Network Layer

- ❑ 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



Graph abstraction



Graph: $G = (N,E)$

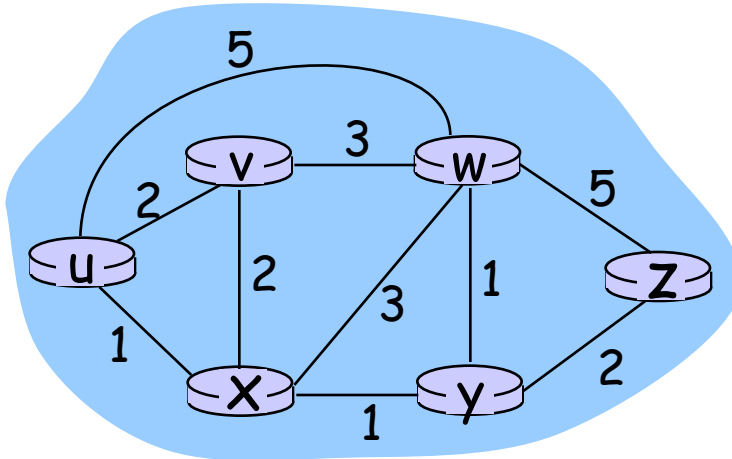
$N =$ set of routers = $\{ u, v, w, x, y, z \}$

$E =$ 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

$$\text{cost}(u, x, y, z) = c(u, x) + c(x, y) + c(y, z) = 1 + 1 + 2 = 4$$

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

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

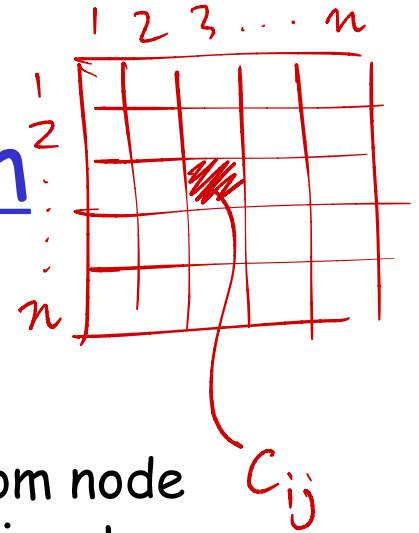
Dynamic:

- ❑ 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

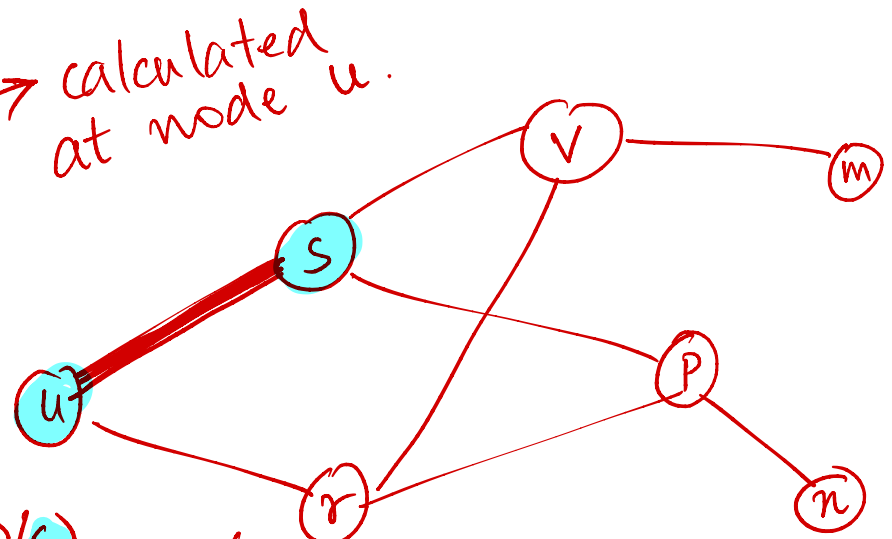
Notation:

- $c(x,y)$: link cost from node x to y; $= \infty$ 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

set of nodes which we know to optimal path

- 1 **Initialization:**
- 2 $N' = \{u\}$
- 3 for all nodes v
- 4 if v adjacent to u
- 5 then $D(v) = c(u,v)$
- 6 else $D(v) = \infty$
- 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'**



$$D(s) = c(u,s) = 3$$

$$D(r) = c(u,r) = 10$$

$$D(v) = \infty$$

$$D(p) = \infty$$

$$D(m) = \infty$$

$$D(n) = \infty$$

$$D(v) = \min \{ D(v), D(x) + c(x,v) \}$$

at node u.

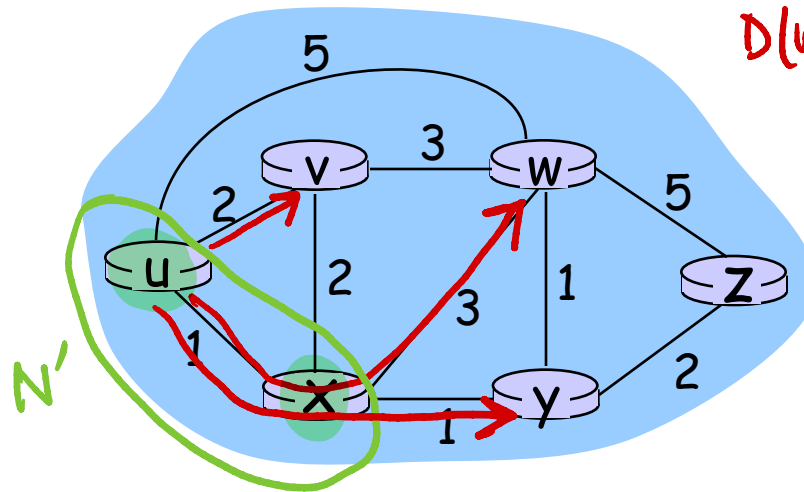
$$= \min \{ 2, 1+2 \}$$

Dijkstra's algorithm: example

$$= \min \{ 2, 3 \}$$

$$= 2$$

| 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 | u,x | 2,u | 4,x | 1,u | 2,x | ∞ |
| 2 | u,x,y | | 3,y | | | |



$$D(w) = \min \{ 5, 1+3 \} = 4$$

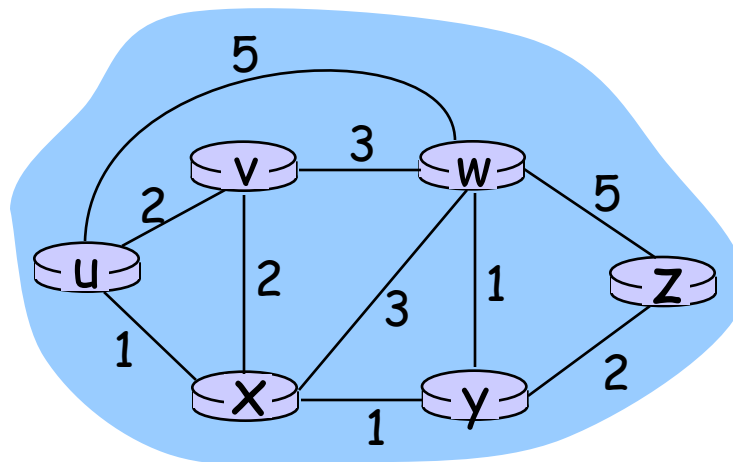
$$D(y) = \min \{ \infty, 1+1 \} = 2$$

$$D(z) = \min \{ \infty, 1+\infty \} = \infty$$

$$D(w) = \min \{ 4, 2+1 \} = 3$$

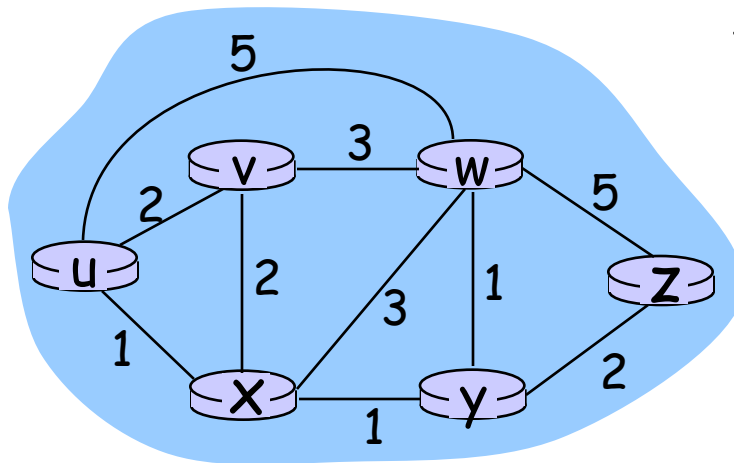
Dijkstra's algorithm: example

| 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 | ∞ |



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 |

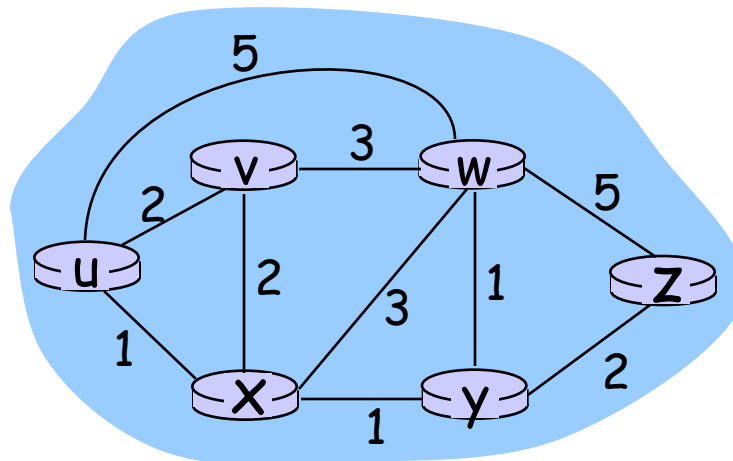


$$D(v) = \min \{ 2, 2 + \infty \} = 2$$

$$D(z) = \min \{ \infty, 2 + 2 \} = 4$$

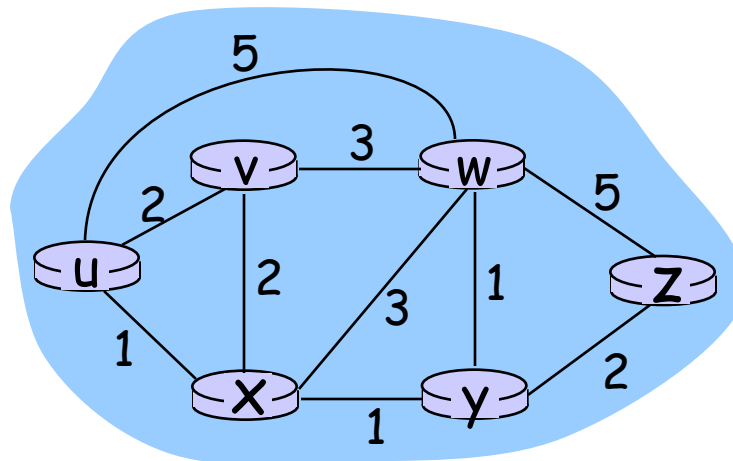
Dijkstra's algorithm: example

| 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 |



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 |

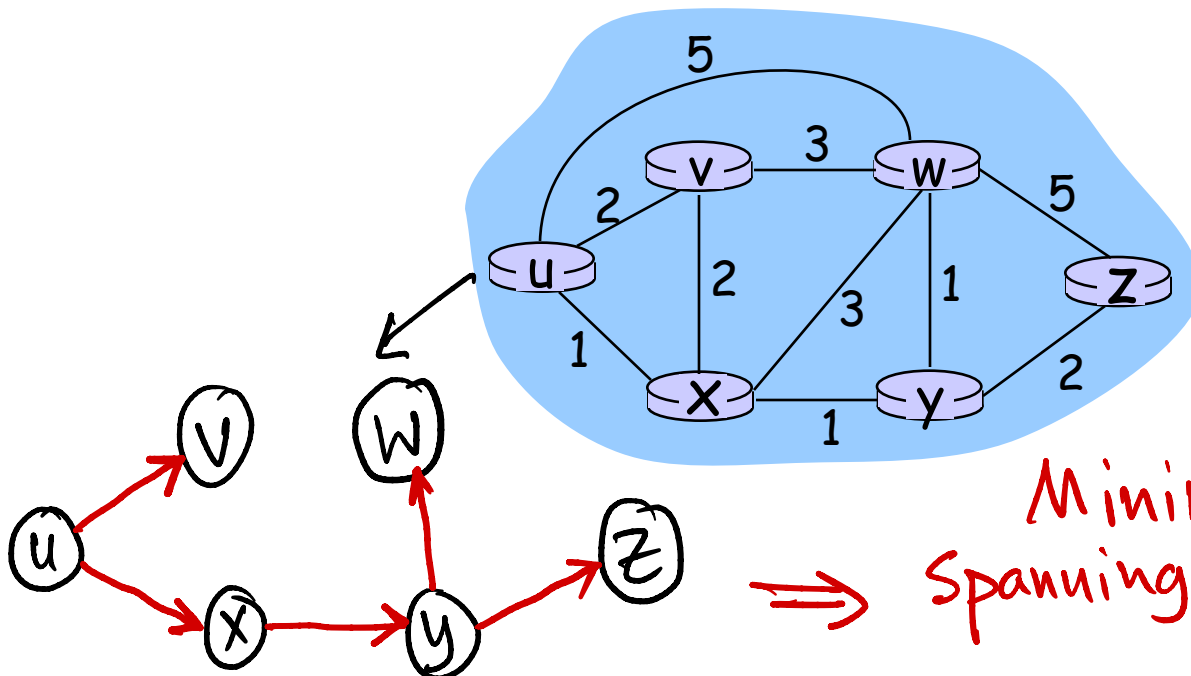


Dijkstra's algorithm: example

| 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 | 1, u | 2, x | ∞ |
| 2 | uxy | 2, u | 3, y | 4, x | 2, x | 4, y |
| 3 | uxyv | 2, u | 3, y | 3, y | 4, y | 4, y |
| 4 | uxyvw | 2, u | 3, y | 3, y | 4, y | 4, y |
| 5 | uxyvwz | 2, u | 3, y | 3, y | 4, y | 4, y |

want to find least-cost path to node z.

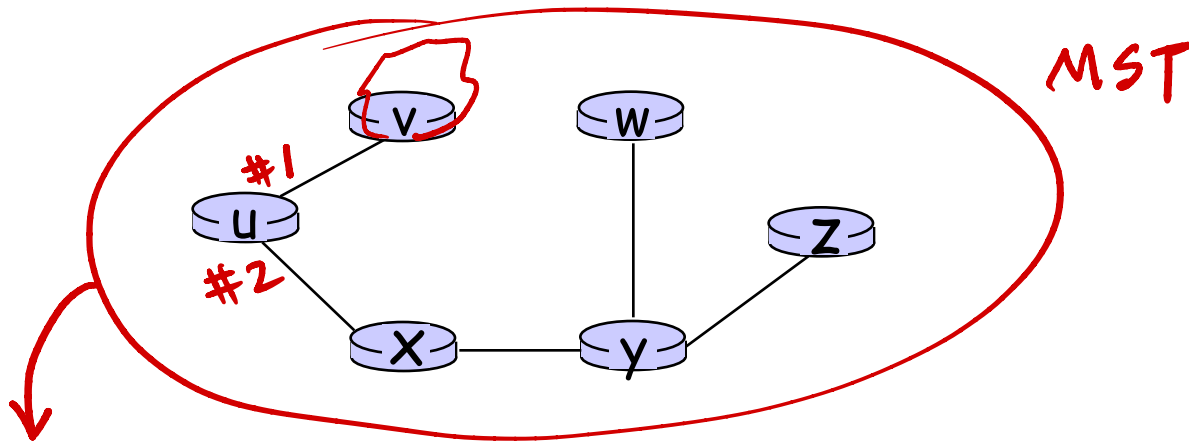
$$D(z) = 4$$



Minimum Spanning Tree

Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

| destination | link | interface |
|-------------|-------|-----------|
| v | (u,v) | #1 |
| x | (u,x) | #2 |
| y | (u,x) | #2 |
| w | (u,x) | #2 |
| z | (u,x) | #2 |

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: $O(n \log n)$

In reality, link cost = $f(\text{traffic on link})$

Does that lead to a problem?

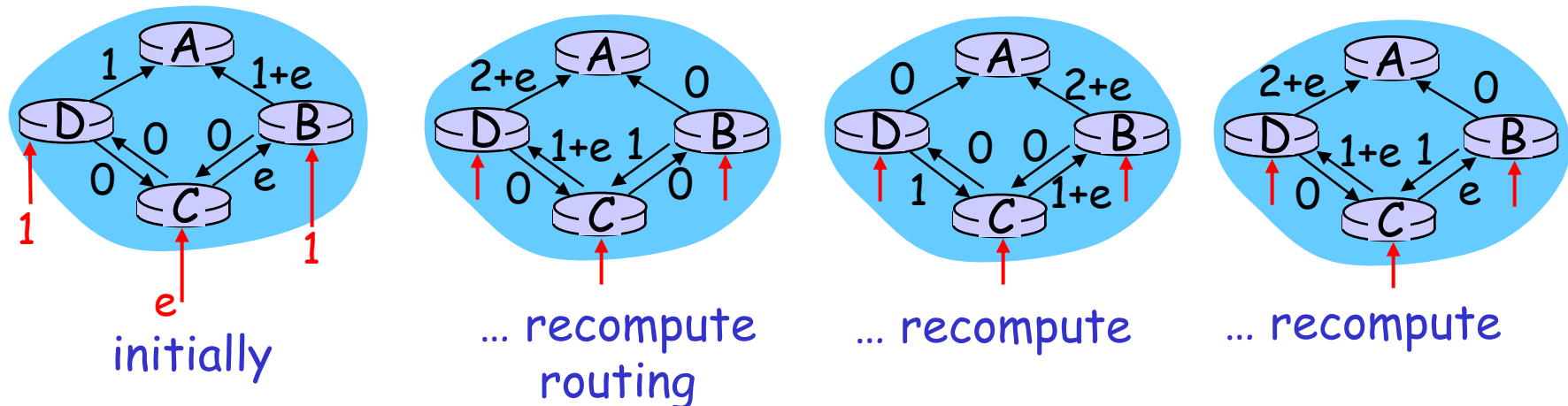
Dijkstra's algorithm, discussion

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: $O(n \log n)$

Oscillations possible:

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



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

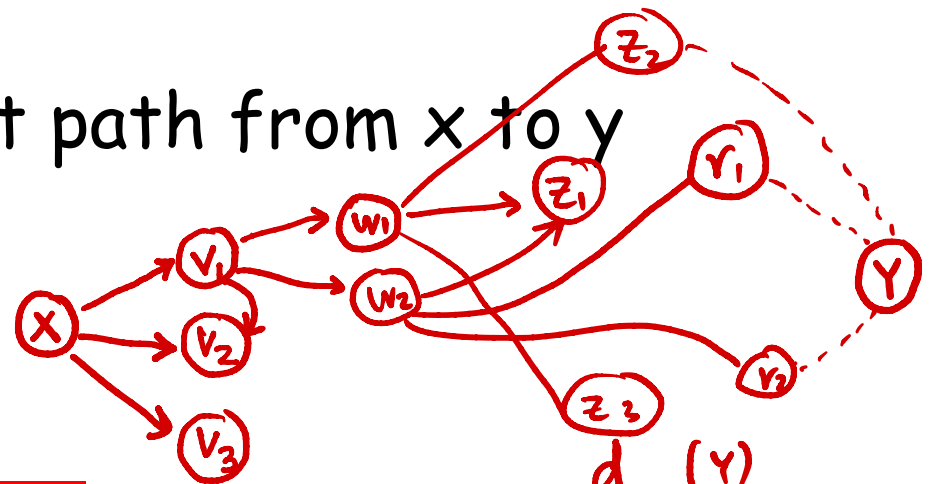
Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

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

Then



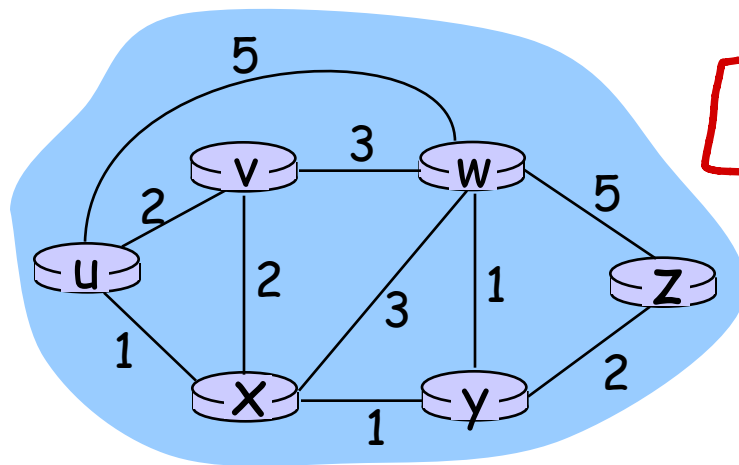
$$d_x(y) = \min_v \{ c(x, v) + d_v(y) \}$$

where min is taken over all neighbors v of x

$$d_x(y) = \min \left\{ \underbrace{c(x, v_1) + d_{v_1}(y)}_{\text{cost to Y via } v_1}, \underbrace{c(x, v_2) + d_{v_2}(y)}_{\text{cost to Y via } v_2}, \underbrace{c(x, v_3) + d_{v_3}(y)}_{\text{via } v_3} \right\}$$

Network Layer 4-81

Bellman-Ford example



Ground truth

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

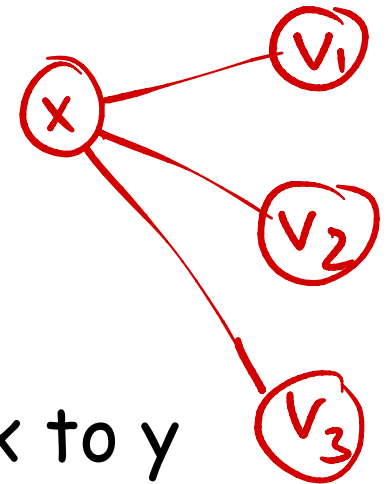
B-F equation says:

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

Node that achieves minimum is next hop in shortest path → forwarding table

$D_x(y), y \in N$
 $D_{v_1}, D_{v_2}, D_{v_3}$

Distance Vector Algorithm



$$D_{v_1} = D_{v_1}(y), y \in N$$

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $D_x = [D_x(y): y \in N]$
- Node x knows cost to each neighbor v : $c(x,v)$
- Node x maintains $D_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
 - For each neighbor v , x maintains $D_v = [D_v(y): y \in N]$

Distance vector algorithm (4)

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

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

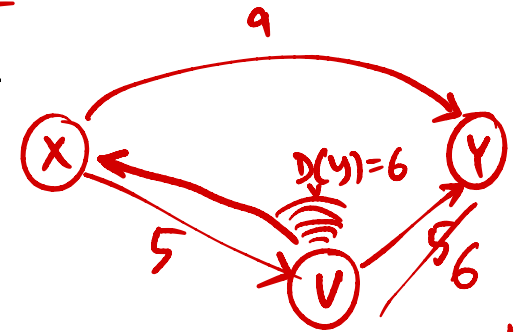
- Under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

$d_x(y)$
true optimal

$D_x(y)$
algo's estimate

Distance Vector Algorithm (5)

$$D_x(v) = 5$$
$$D_x(y) = 9$$



Iterative, asynchronous:

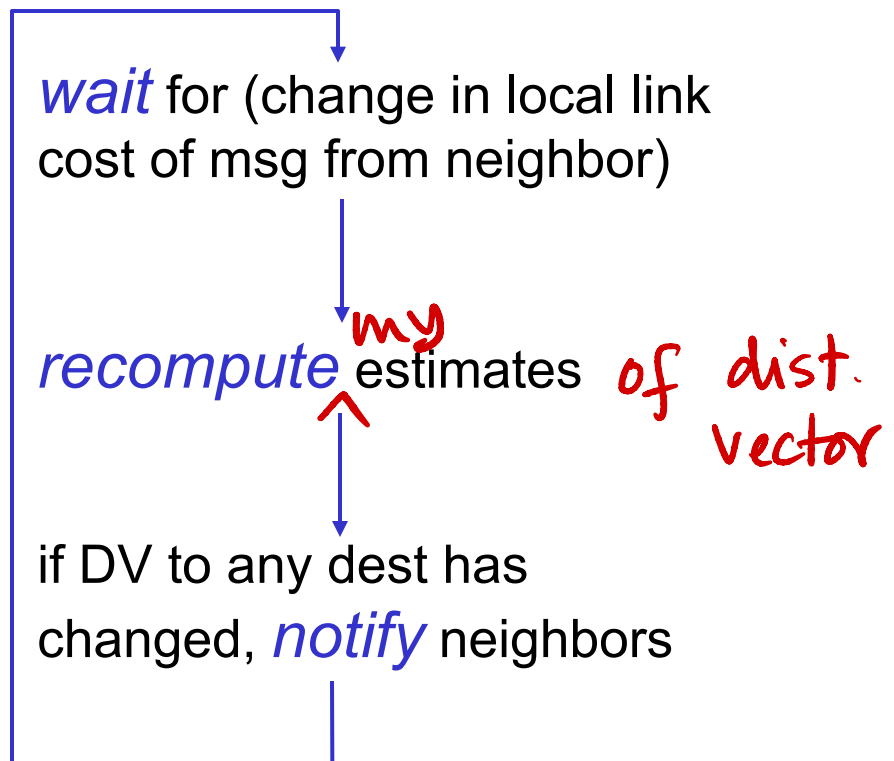
each local iteration caused by:

- 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:



Does this mean a network flood every time a link cost changes?

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

$$= \min\{2+0, 7+1\} = 2$$

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

$$= \min\{2+1, 7+0\} = 3$$

node x table

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

node y table

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

node z table

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

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

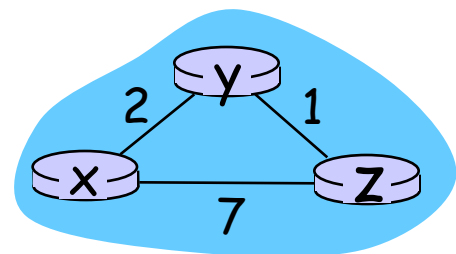
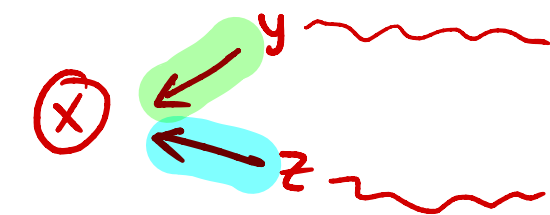
| | | cost to | | |
|------|---|---------|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 7 |
| | y | 2 | 0 | 1 |
| | z | 7 | 1 | 0 |

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

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

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

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



nodes

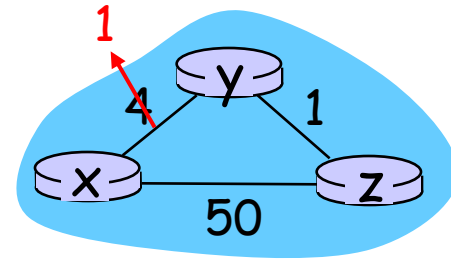
time

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



$$D_Y(x) = \cancel{4} + 1$$

At time t_0 , y detects the link-cost change, updates its DV, and informs its neighbors.

$$D_Z(x) = \cancel{50} + 2$$

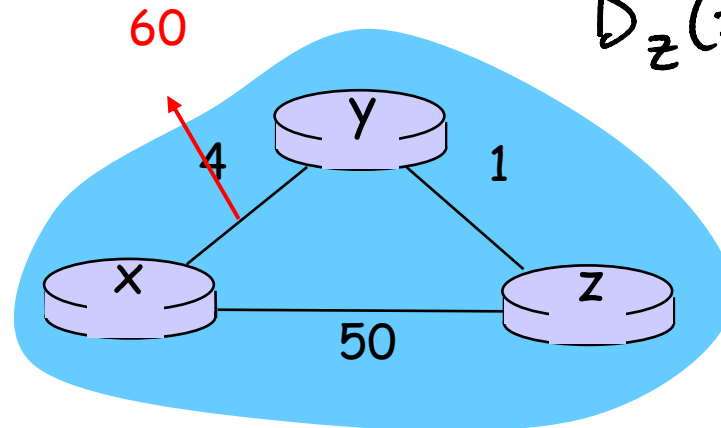
“good
news
travels
fast”

At time t_1 , z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2 , y receives z 's update and updates its distance table. y 's least costs do not change and hence y does *not* send any message to z .

Distance Vector: link cost changes

$$D_Y(x) = \min \{ 60, 1 + D_Z(x) \}$$



$$\begin{aligned} D_Z(x) &= \min \{ 50, 1 + D_Y(x) \} \\ &= \min \{ 50, 1 + 4 \} \\ &= 5 \end{aligned}$$

New DV update
from Y is

$$D_Y(x) = 6$$

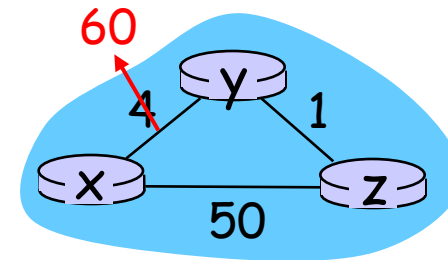
Z sends update
 $D_Z(x) = 7$

What happens now ?

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?

Tradeoffs

What will you recommend ?

Link State?
Distance Vector?

There is no right answer

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?

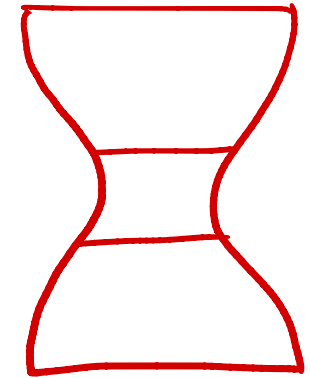
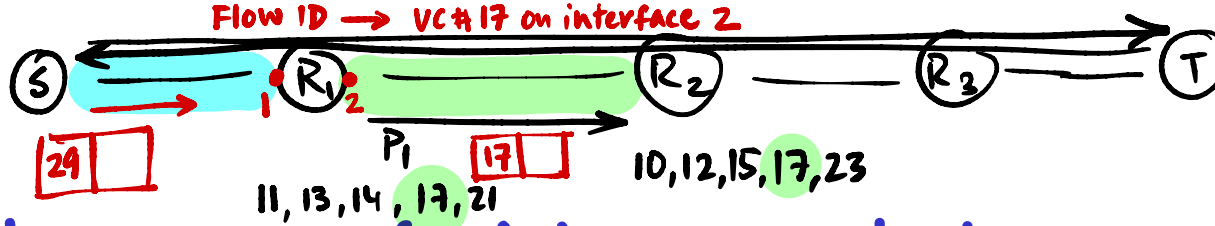
LS:

- 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

VC#



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