Lecture 18: Network Layer
Link State and Distance Vector Routing

COMP 332, Fall 2018

Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

1. Announcements
   - homework 7
     - written due Wed., programming due next Wed.

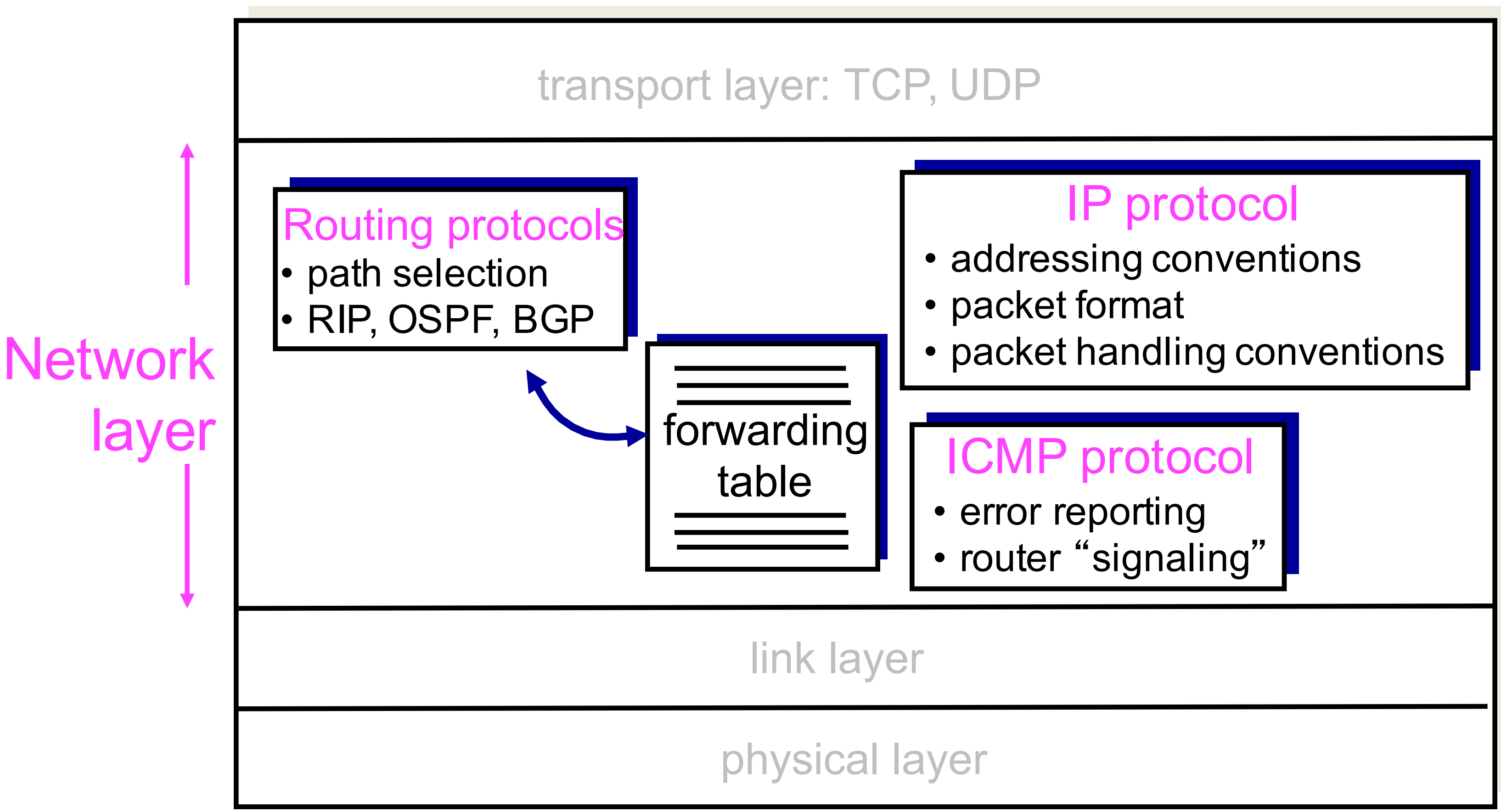
2. Control plane
   - overview
   - link state routing
   - distance vector routing

Control Plane
OVERVIEW

Internet's network layer

Network layer functions on hosts and routers

– control plane vs. data plane



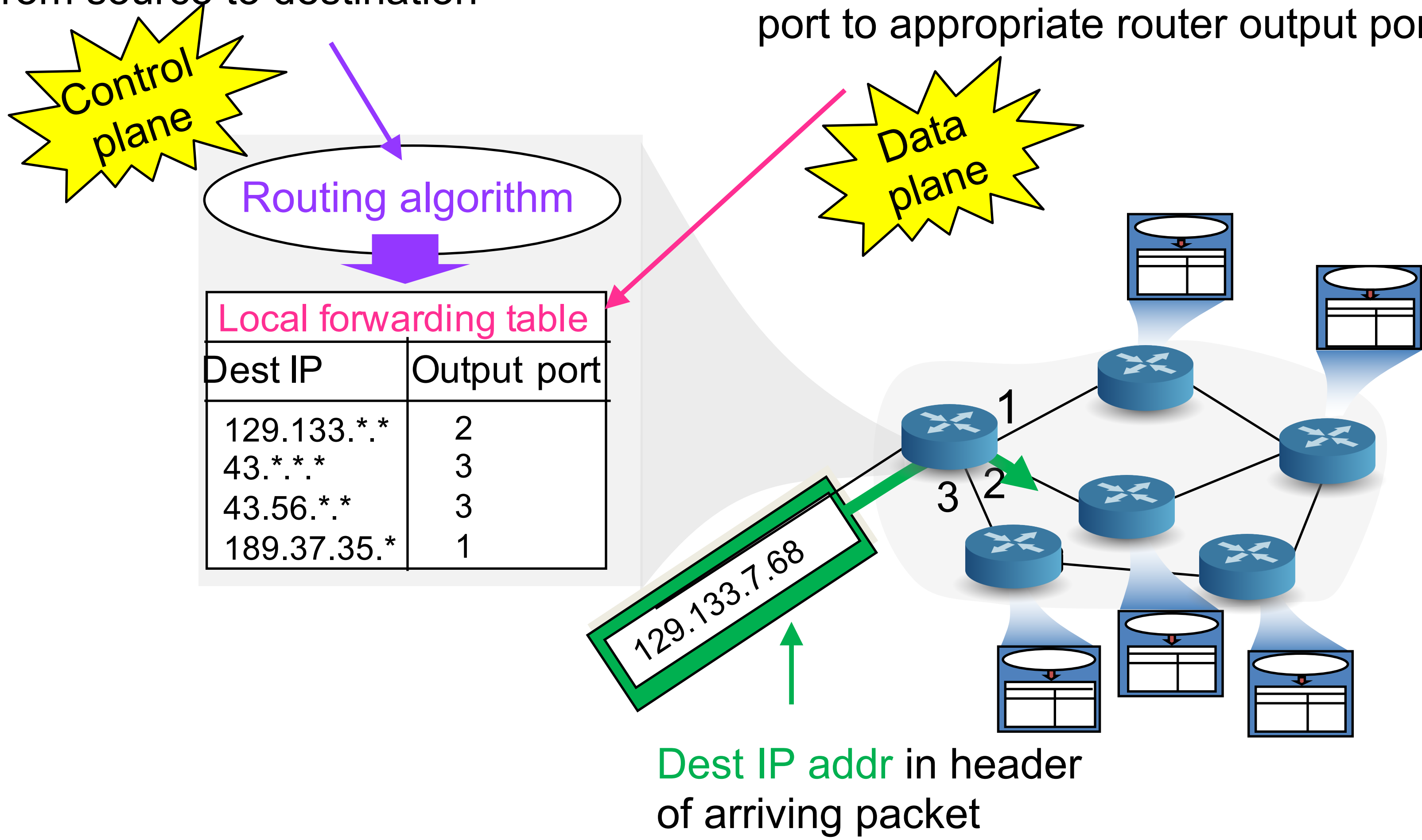
Control vs. data plane functions

Routing (slower time scale)

- determine route taken by packets from source to destination

Forwarding (faster time scale)

- move packets from router's input port to appropriate router output port



Dest IP	Output port
129.133.*.*	2
43.*.*.*	3
43.56.*.*	3
189.37.35.*	1

Routing protocols

Goal

- determine “good” path from sending hosts to receiving host, through network of routers

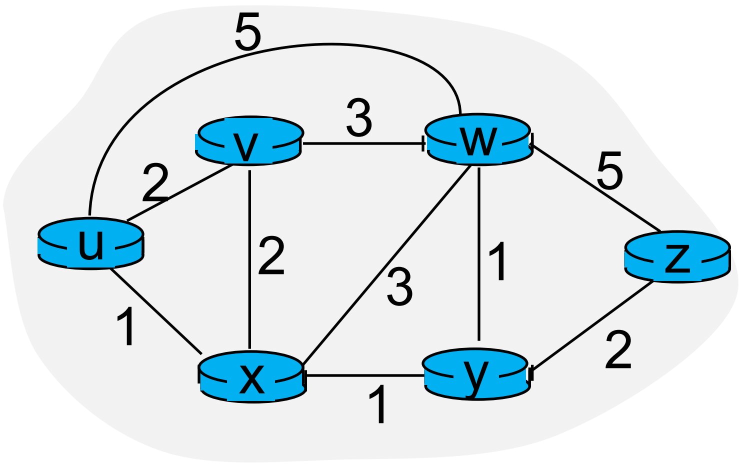
Path

- sequence of routers packets will traverse in going from given initial source host to given final destination host

“Good”

- least “cost”, “fastest”, “least congested”, …
- correctness constraints
  - no loops
  - no dead-ends

Abstract network as a graph

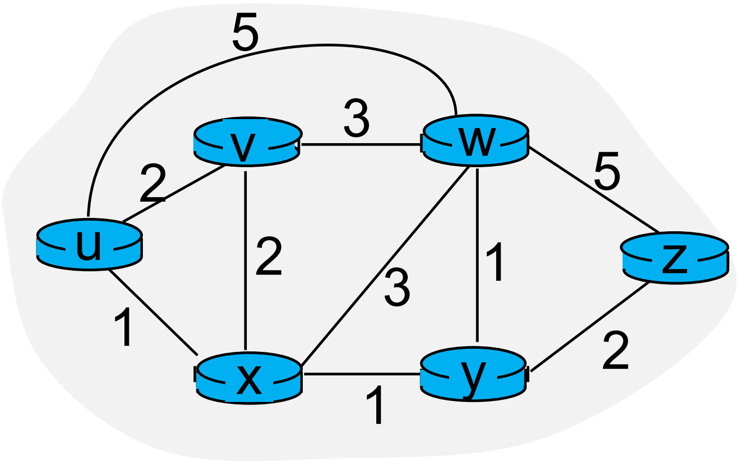


Graph: G = (N,E)

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) }

Link costs



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

Q: how to set cost?
Cost could always be 1, related to bandwidth, inversely related to congestion, …

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

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

Routing algorithm: algorithm that finds least-cost path

Classifying routing algorithms

Global vs. decentralized info

- global link state algorithms
  - all routers have complete topology, link cost info
- decentralized distance vector algorithms
  - router knows physically-connected neighbors, link costs to neighbors
  - iterative computation
  - exchange info with neighbors
- both link state and distance vector algorithms used on Internet
  - first cover abstractly and then talk about specific Internet protocols
  - OSPF, BGP, RIP, …

Static vs. dynamic topology

- static: routes change slowly over time
- dynamic: routes change more quickly
  - periodic update in response to link cost changes

Control Plane

LINK STATE ROUTING

Dijkstra's algorithm

Link state: i.e., network topology, link costs

- known to all nodes, accomplished via link state broadcast
  - msg sent to every other node in network
- all nodes have same global info

Computes least cost paths

- from one "source" node to all other nodes
- obtain forwarding table for that node

Given path, put 1st hop router for each dst in forwarding table

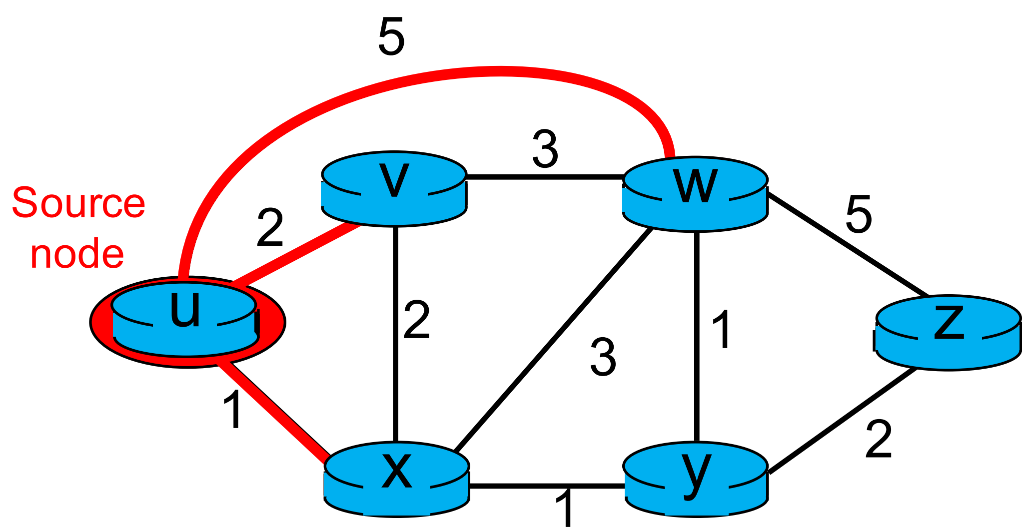
Iterative

- after k iterations, know least cost path to k destinations
  - if n nodes, loop n times

Dijkstra's algorithm

$c(x,y)$: link cost from node x to y
$D(v)$: current cost from source u to dst node v
$p(v)$: predecessor node along path from source u to v
N': set of nodes whose least cost path definitively known

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						
2						
3						
4						
5						



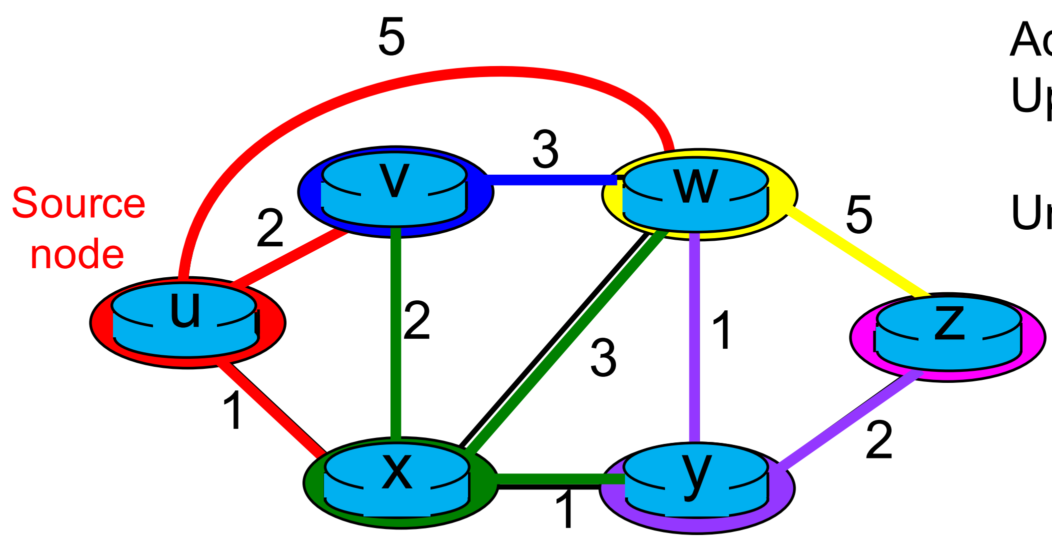
Initialization

N' = {u}
for all nodes v
    if v adjacent to u
        then $D(v) = c(u,v)$
    else $D(v) = \infty$

Dijkstra's algorithm

c(x,y): link cost from node x to y
D(v): current cost from source u to dst node v
p(v): predecessor node along path from source u to v
N': set of nodes whose least cost path definitively known

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					



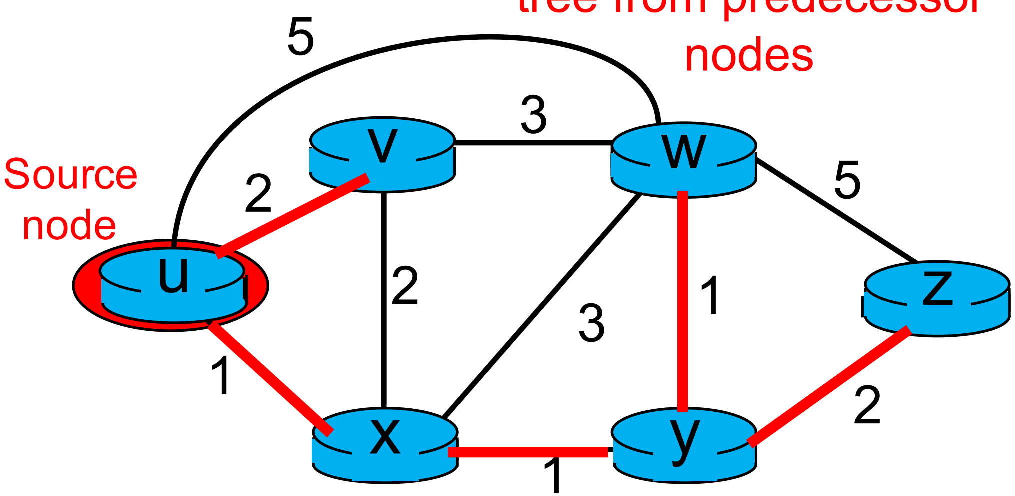
Loop

Find w ∉ N' s.t. D(w) is min
Add w to N'
Update D(v) for all neighbors v ∉ N' of w
D(v) = min(D(v), D(w)+c(w,v))
Until all nodes in N'

Dijkstra's algorithm

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					

1. Build shortest path tree from predecessor nodes



2. Build forwarding table at u

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

Algorithm complexity with n nodes

Each iteration: need to check all nodes not in N'

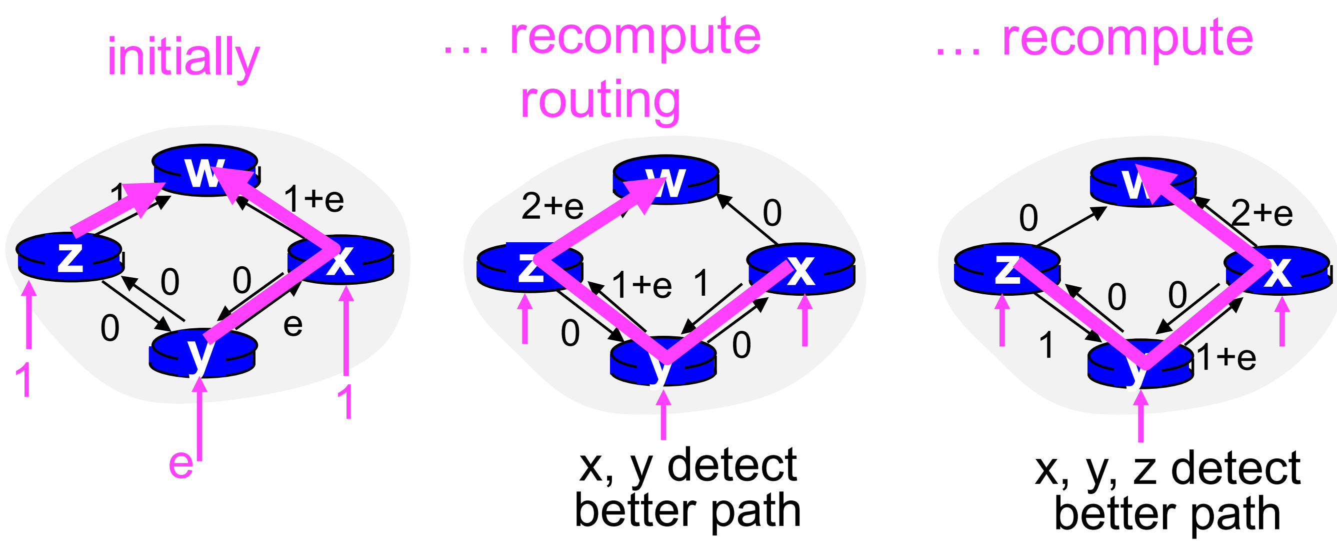
– n(n+1)/2 comparisons: $O(n^2)$, more efficient implementations possible

Network is dynamic

– link goes down: link state broadcast
– router goes down: remove link and all nodes recompute

Oscillations possible

– when congestion or delay-based link cost



Need to prevent routers from synchronizing computations:
Have routers randomize when they send out link advertisements

Control Plane

DISTANCE VECTOR ROUTING

Distance vector routing

Distance vector (DV)

- vector of best known costs from router to each dst and link to use

Each node x maintains

- Link cost from x to each neighbor v
  - $c(x,v)$
- x's own DV
  - $D_x(y)$: estimate of least cost path from x to node y
  - $D_x = [D_x(y): y \in N]$
- DV for each nbr v
  - $D_v(y)$: estimate of least cost path from neighbor v to node y
  - $D_v = [D_v(y): y \in N]$

Each node periodically sends its own DV to neighbors

- rather than link state costs

Bellman-Ford equation to update DV estimates

Uses dynamic programming

- break problem into simpler sub-problems
- solve each sub-problem once and store solution

Bellman-Ford equation

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

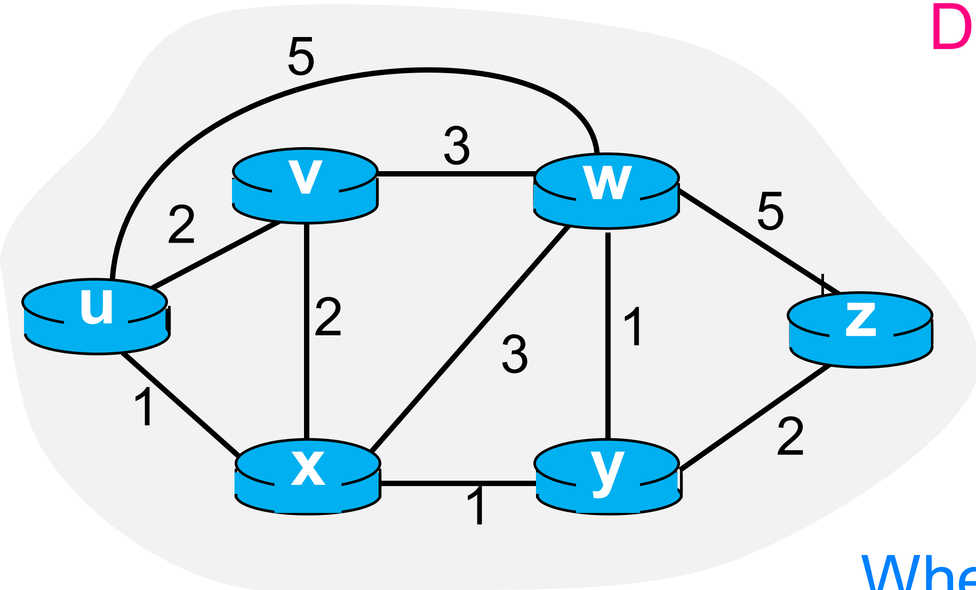
$$D_x(y) = \min_v \{ c(x,v) + D_v(y) \} \text{ for each node } y \in N$$

- $\min_v$: *min* taken over all nbrs v of x
- $c(x,v)$: cost to nbr v
- $D_v(y)$: cost from nbr v to dst y

When x receives new DV estimate from neighbor

- x updates its own DV using B-F equation

Example: compute min cost path from u to z



Bellman-Ford equation

$$D_u(z) = \min \{ c(u,v) + D_v(z), \; c(u,x) + D_x(z), \; c(u,w) + D_w(z) \}$$

$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\}$$

$$= 4$$

Where

$D_v(z) = 5$, $D_x(z) = 3$, $D_w(z) = 3$

Node achieving minimum is next hop in shortest path

- put in forwarding table

Distance vector algorithm run at each node x

Initialization

For all dst $y \in N$

if y is nbr of x

$D_x(y) = c(x, y)$

else

$D_x(y) = \infty$

For each nbr w and dst $y \in N$

$D_w(y) = \infty$

Send x's DV to all nbrs w

$D_x = [D_x(y) : y \in N]$

Loop

x waits for change in local link cost or DV msg from neighbor

recompute estimates

$D_x(y) = min\ v \{ c(x,v) + D_v(y) \}$

if x's DV to any dst has changed, *notify* neighbors

Q: when does loop terminate?
When no more 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

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

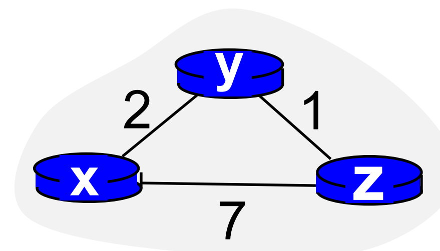
cost to (from)

	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Node z

cost to (from)

	x	y	z
x	∞	∞	∞
y		∞∞∞	∞
z	7	1	0



time