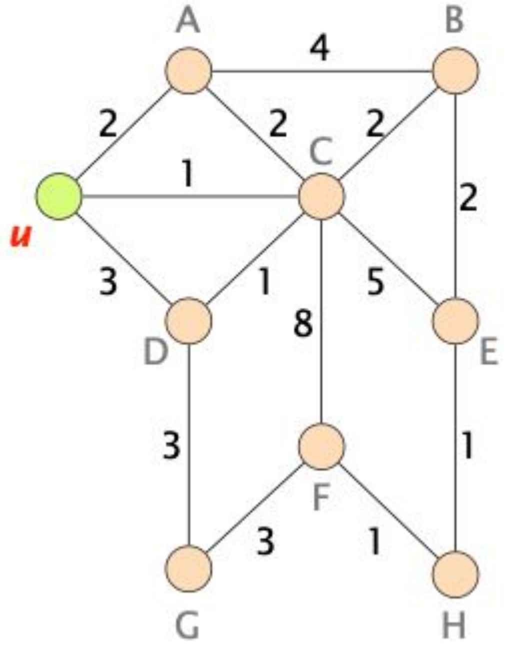
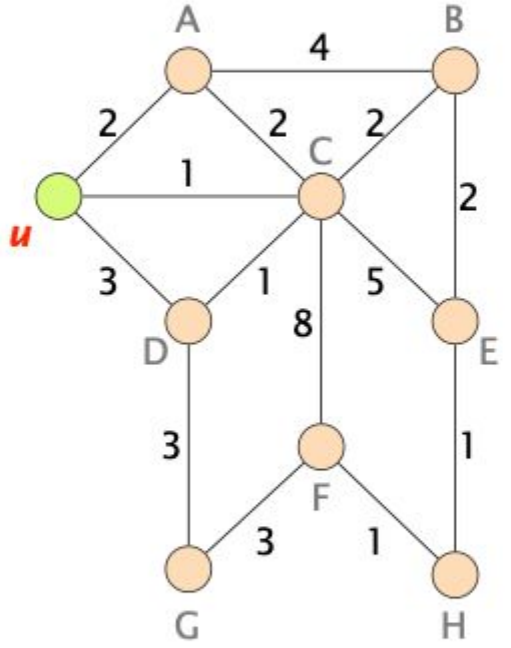
Dijkstra’s Example

Starting from node u, (i) manually compute Dijkstra’s algorithm, and then (ii) list the obtained shortest-paths from u to each of the other nodes. The algorithm follows the one discussed in the lecture. If several nodes could next be added to node set S, select the node that comes first in the alphabet.



Dijkstra's Example

Starting from node u, (i) manually compute Dijkstra's algorithm, and then (ii) list the obtained shortest-paths from u to each of the other nodes. The algorithm follows the one discussed in the lecture. If several nodes could next be added to node set S, select the node that comes first in the alphabet.



Node	Path	$\sum$(weights)
A	u - A	2
B	u - C - B	3
C	u - C	1
D	u - C - D	2
E	u - C - B - E	5
F	u - C - B - E - H - F	7
G	u - C - D - G	5
H	u - C - B - E - H	6

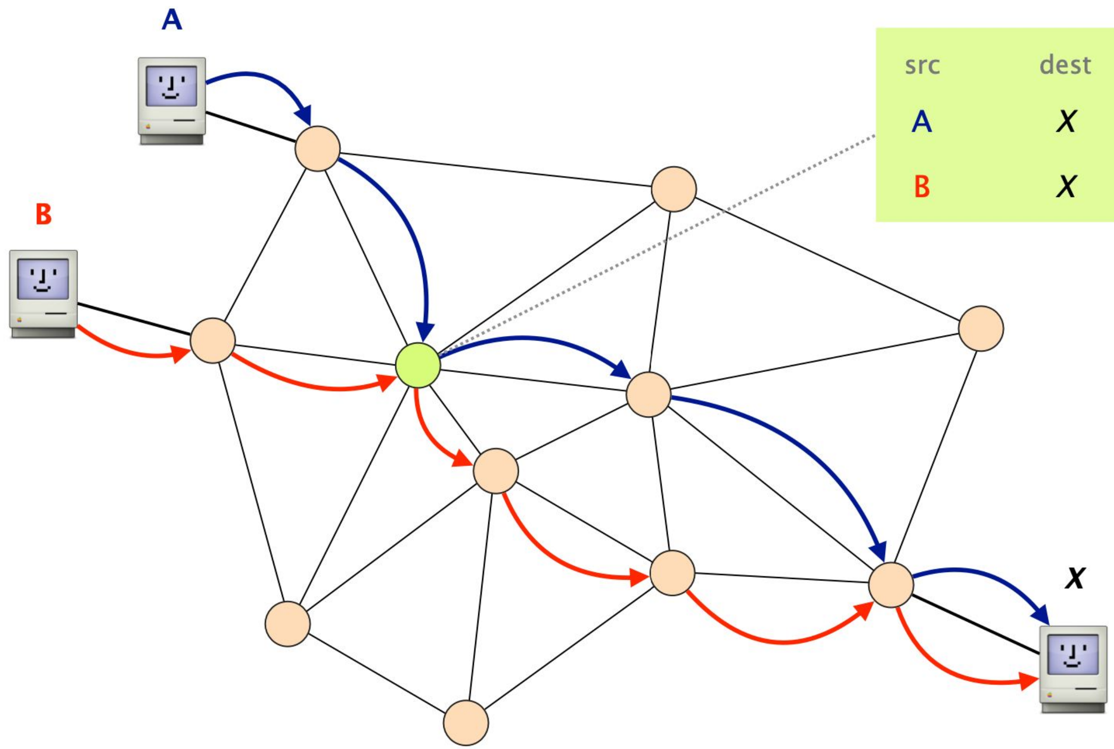
Forwarding vs Routing

	forwarding	routing
goal	directing packet to an outgoing link	computing the paths packets will follow
scope	local	network-wide
implem.	hardware usually	software usually
timescale	nanoseconds	milliseconds (hopefully)

Forwarding Depends on Destination, but Can Also Consider Other Criteria

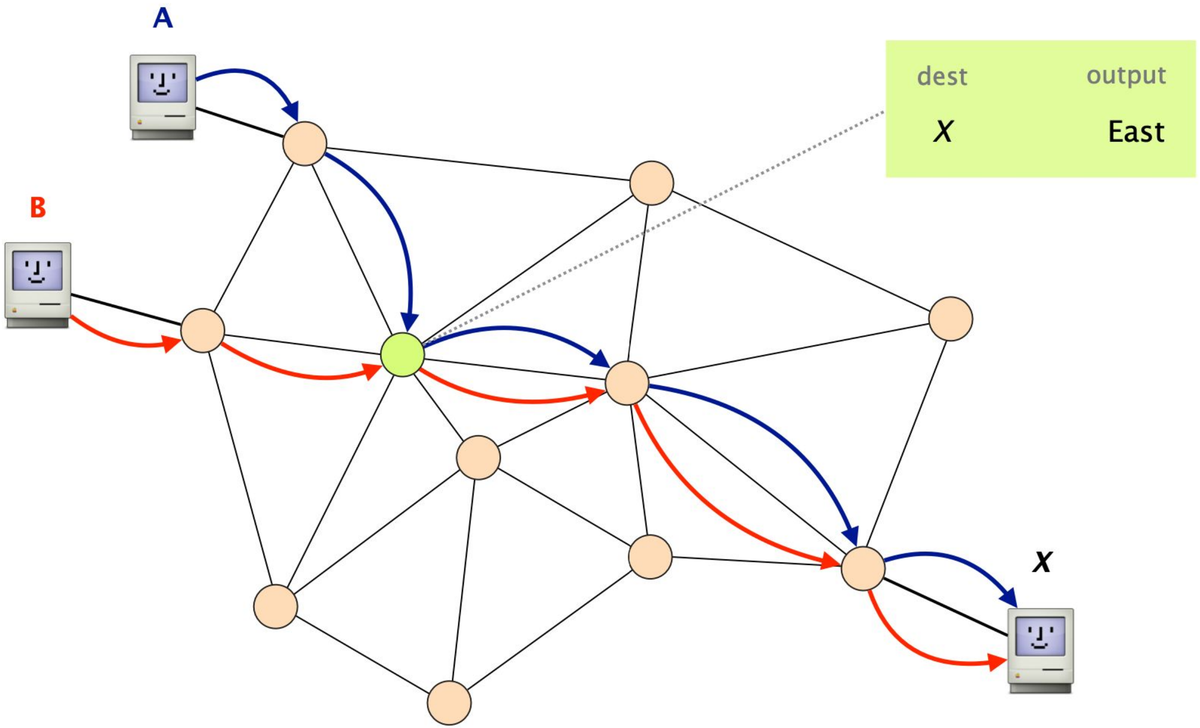
- Destination
- Source
- Input port
- Any other header field

Forwarding on Both Source and Destination - Paths from Different Sources can Differ

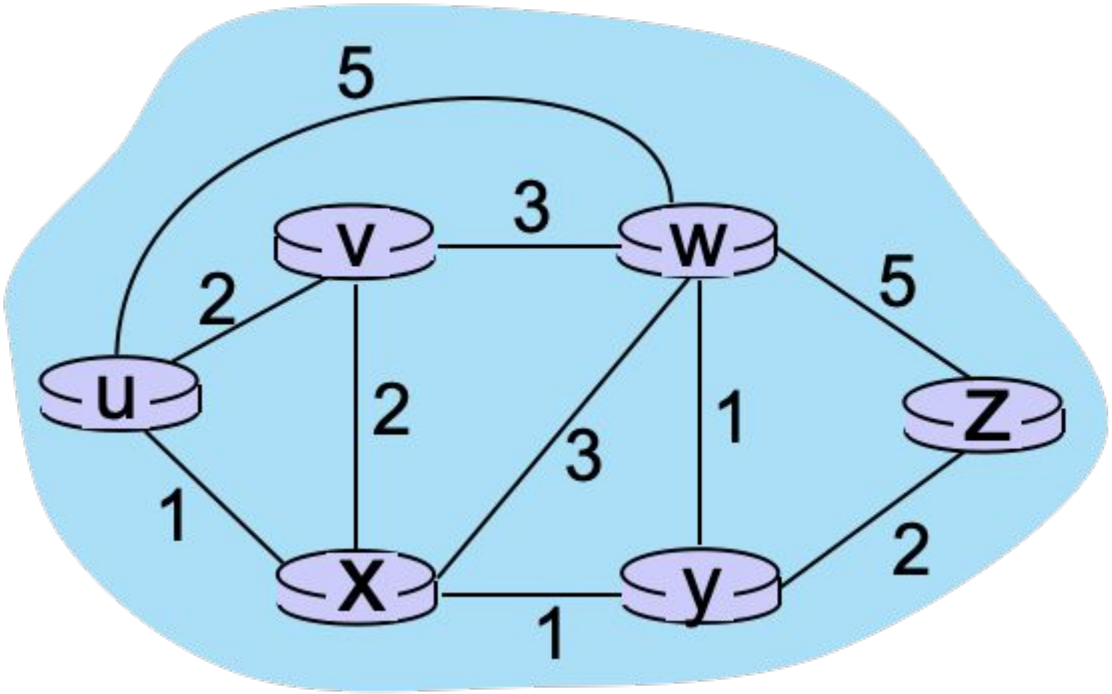


src	dest	output
A	*X*	East
B	*X*	South-East

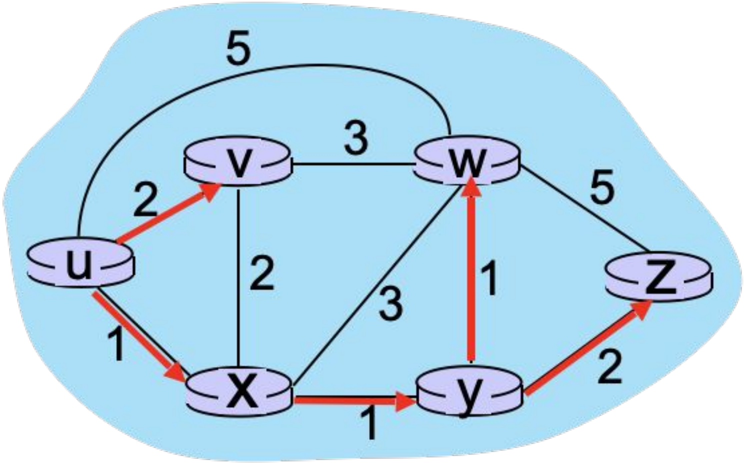
Destination-Based Routing, Once Paths from Sources Overlap They Remain the Same



Dijkstra’s Algorithm for Shortest Path Search



Forwarding Table Comes from Dijkstra's Algorithm Results



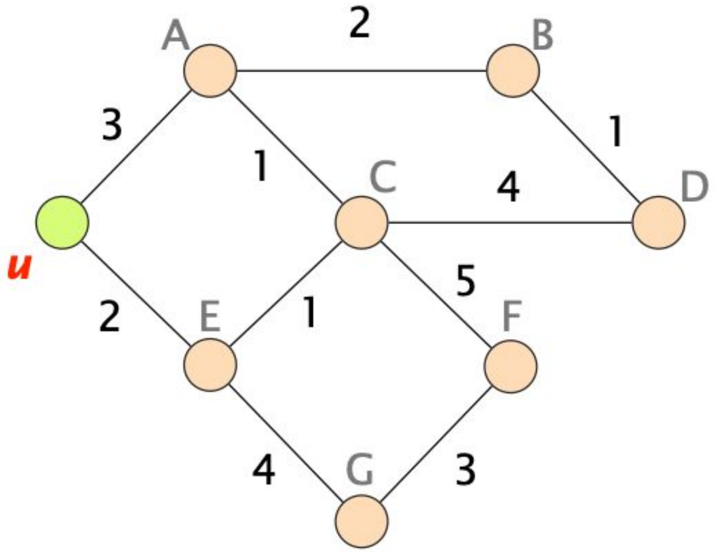
resulting forwarding table in u:

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

route from *u* to *v* directly

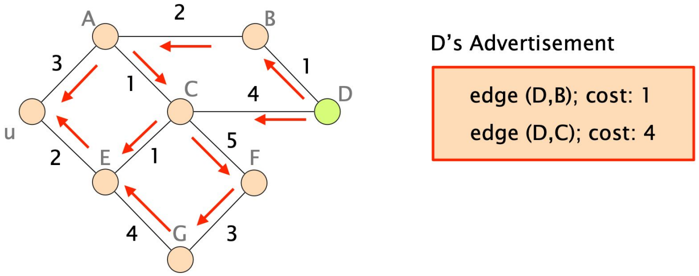
route from u to all other destinations via *x*

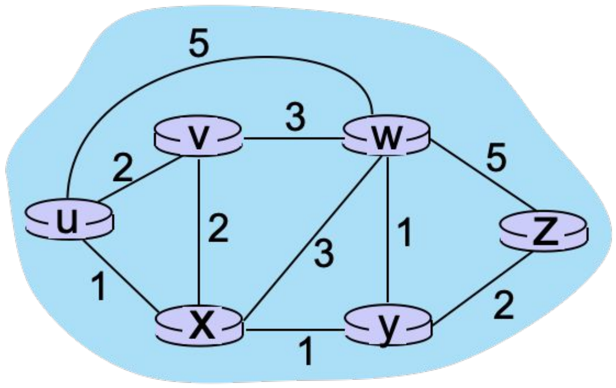
Dijkstra's Algorithm for Shortest Path Search



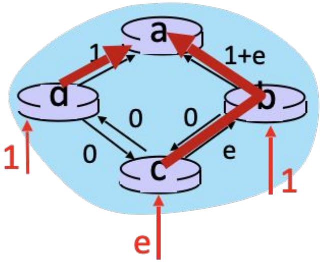
Dijkstra's Algorithm -> Link State Routing

- Each router floods its link state information to other n routers in order to generate a global view
- Updates are sent when things change, and only the difference is sent, not everything
- Any drawbacks you can think of?
- ```
  U: {v=2, x=1, w=5}
  V: {u=2, x=2, w=3}
  W: {v=3, u=5, x=3, y=1, z=5}
  X: {u=1, v=2, w=3, y=1}
  Y: {x=1, w=1, z=2}
  Z: {w=5, y=2}
  ```

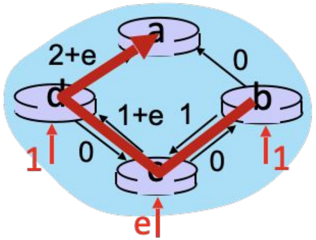



  ```

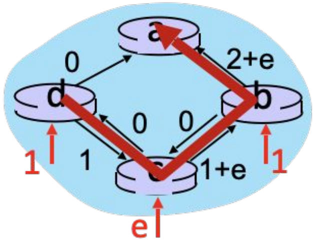
Dynamic Weights -> Route Oscillations



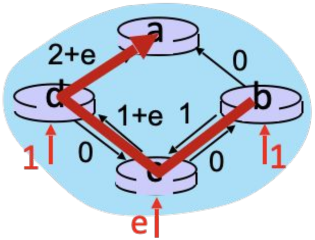
initially



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

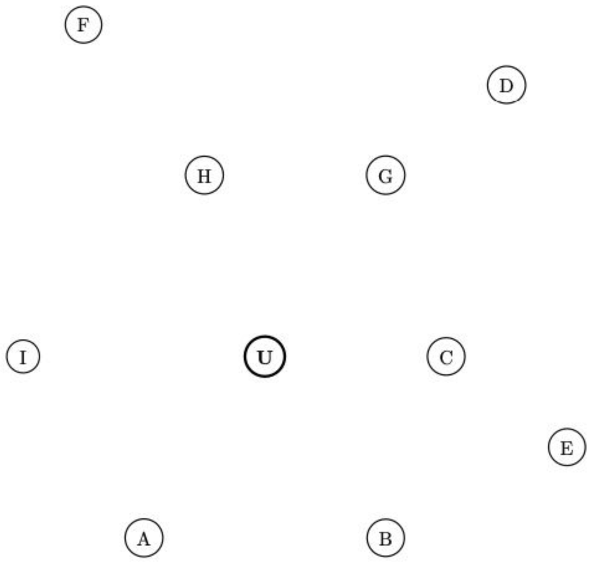


given these costs,
find new routing….
resulting in new costs



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

Reverse Dijkstra is Possible From Results

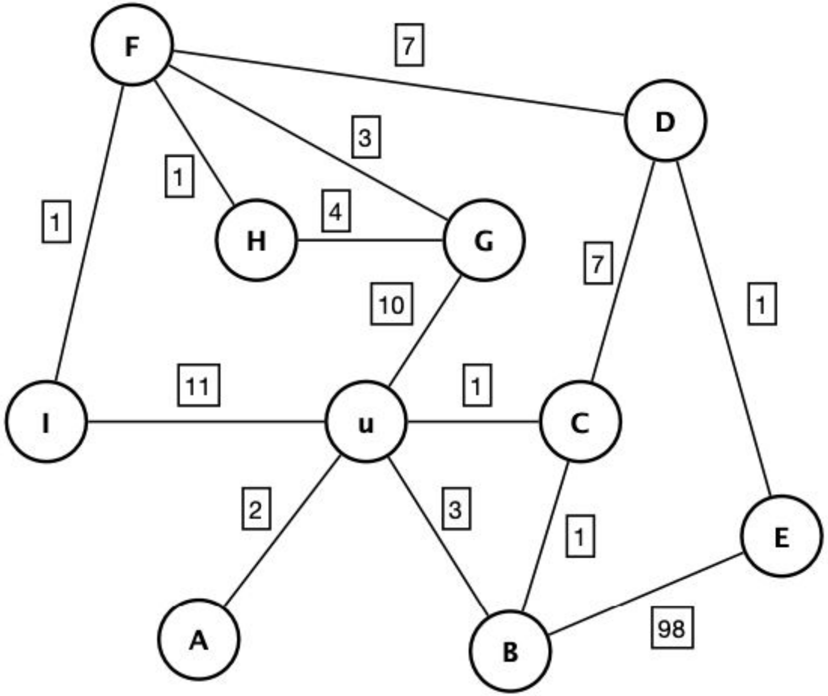


A network consisting of 10 nodes with unknown links and link weights.

#	U	A	B	C	D	E	F	G	H	I
1	0	2	3	1	-	-	-	10	-	11
2	0	2	2	1	8	-	-	10	-	11
3	0	2	2	1	8	-	-	10	-	11
4	0	2	2	1	8	100	-	10	-	11
5	0	2	2	1	8	9	15	10	-	11
6	0	2	2	1	8	9	15	10	-	11
7	0	2	2	1	8	9	13	10	14	11
8	0	2	2	1	8	9	12	10	14	11
9	0	2	2	1	8	9	12	10	13	11
10	0	2	2	1	8	9	12	10	13	11

For each iteration (1 to 10) the table shows the shortest path found by Dijkstra's algorithm performed on node **U** towards all other nodes.

Reverse Dijkstra is Possible From Results - Solution

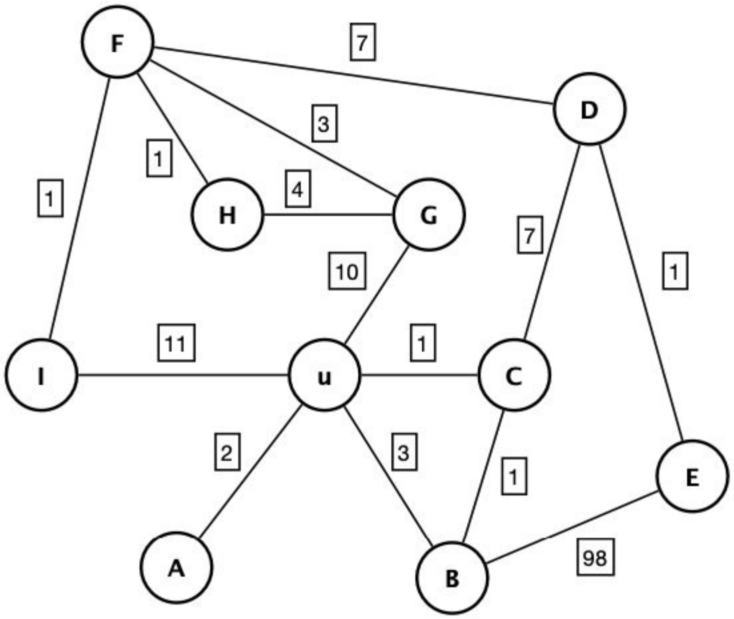


#	U	A	B	C	D	E	F	G	H	I
1	0	2	3	1	-	-	-	10	-	11
2	0	2	2	1	8	-	-	10	-	11
3	0	2	2	1	8	-	-	10	-	11
4	0	2	2	1	8	100	-	10	-	11
5	0	2	2	1	8	9	15	10	-	11
6	0	2	2	1	8	9	15	10	-	11
7	0	2	2	1	8	9	13	10	14	11
8	0	2	2	1	8	9	12	10	14	11
9	0	2	2	1	8	9	12	10	13	11
10	0	2	2	1	8	9	12	10	13	11

For each iteration (1 to 10) the table shows the shortest path found by Dijkstra's algorithm performed on node **U** towards all other nodes.

Reverse Dijkstra is Possible From Results

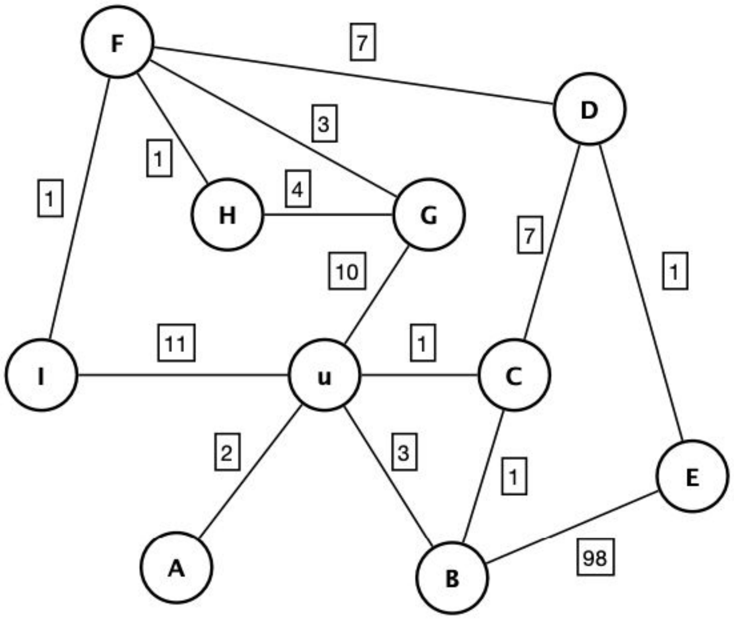
Could there be an additional link starting from node C which you could not identify based on the output from Dijkstra? If you think that is possible, give an example (link between node C and node ...) and indicate in which range the weight of this link could be. Otherwise, explain why this is not possible.



Reverse Dijkstra is Possible From Results

Could there be an additional link starting from node C which you could not identify based on the output from Dijkstra? If you think that is possible, give an example (link between node C and node ...) and indicate in which range the weight of this link could be. Otherwise, explain why this is not possible.

Solution: Possible. For example link between C and G with weight greater (or equal) than 9.



Link State Algorithms

Pros

- Fast convergence
- Event-driven updates
- Every router can determine the best path

Cons

- Computationally expensive
- Memory intensive
- If a network is constantly changing, bandwidth can suffer from overhead of messages

Link State Protocols

Open Shortest Path First (OSPF)

- Dominant LS protocol
- The routing protocol used within large autonomous systems - external is BGP (distance vector, next up)
- Open source
- If you have a network that is larger than small (>4 routers) you're probably best off using OSPF

Routing

Link State == global view

Distance vector == local view

Distance Vector Routing

Rather than building routes with a global view of the network, nodes (routers) only learn from their adjacent neighbors.

- Sometimes called “routing by rumor” or a “gossip” protocol

Distance Vector Routing

- Let $d_x(y)$ be the cost of the least-cost path known by x to reach y

until convergence

Distance Vector Routing

- Let $d_x(y)$ be the cost of the least-cost path known by x to reach y

- Each node bundles these distances into one message (called a vector) that it repeatedly sends to all its neighbors

until convergence

Distance Vector Routing

until convergence

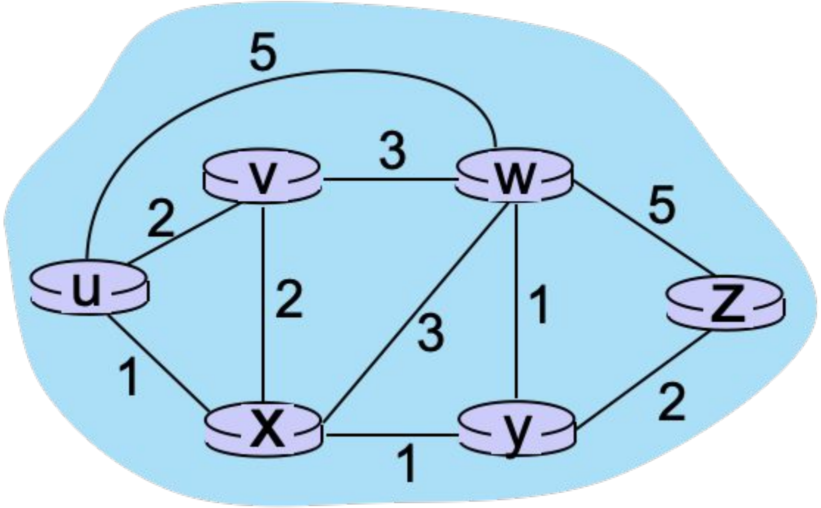
- Let $d_x(y)$ be the cost of the least-cost path known by x to reach y
- Each node bundles these distances into one message (called a vector) that it repeatedly sends to all its neighbors
- Each node updates its distances based on neighbors' vectors:
- $d_x(y) = \min\{ c(x,v) + d_v(y) \}$ over all neighbors v

Bellman-Ford

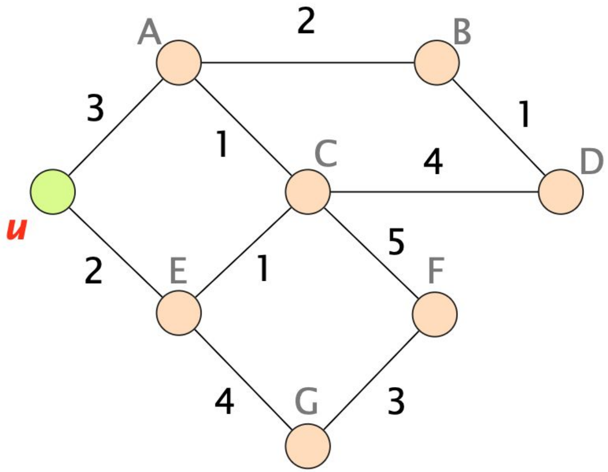
$$d_u(z) = \min\{c(u,v) + d_v(z),$$
$$c(u,x) + d_x(z),$$
$$c(u,w) + d_w(z)\}$$

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

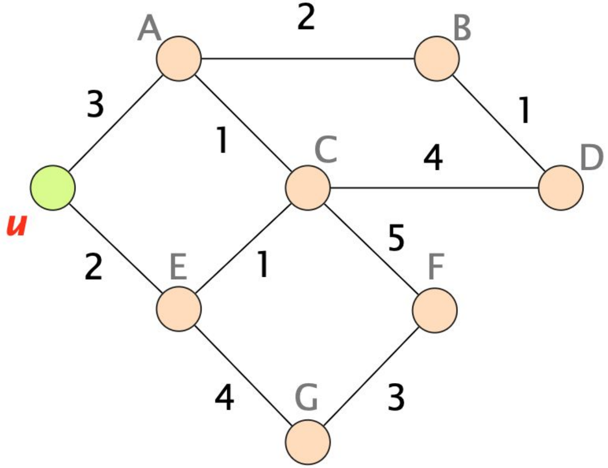
$$= 4$$



We'll Compute the Shortest Path from u to D



The Values Computed by a Node u Depend on What it Learns from its Neighbors (A and E)

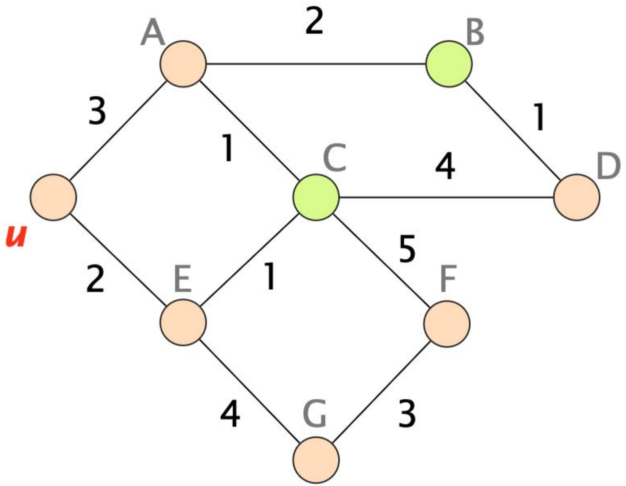


$$d_x(y) = \min\{ c(x,v) + d_v(y) \}$$

over all neighbors v

$$d_u(D) = \min\{ c(u,A) + d_A(D),\ c(u,E) + d_E(D) \}$$

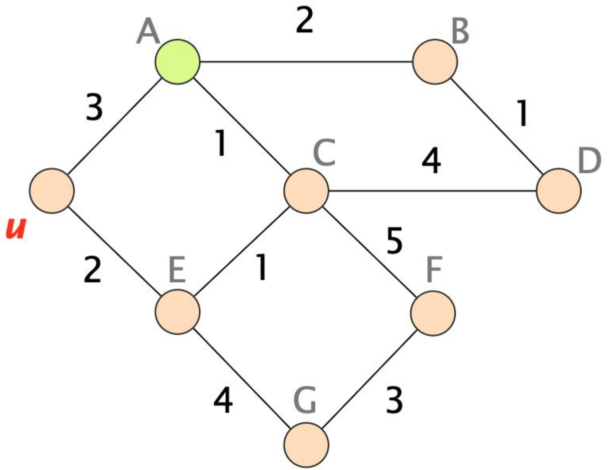
To Understand, Let's Start with Direct Neighbors of D



$d_B(D) = 1$

$d_C(D) = 4$

B and C Announce Their Vectors to Their Neighbors, Which Allows A to Compute a Path to D

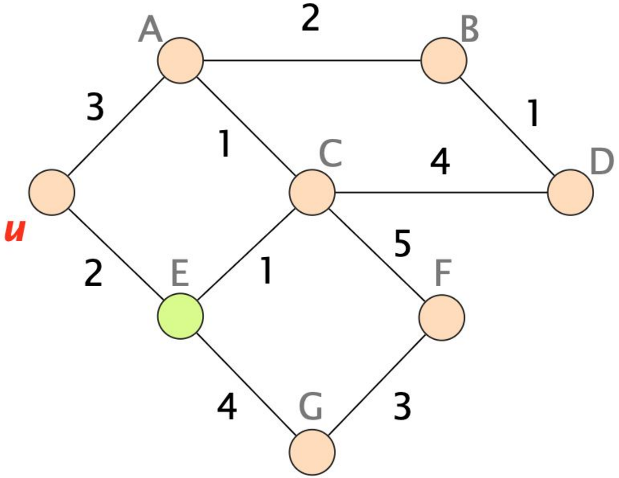


$$d_A(D) = \min\{\, 2 + d_B(D),\; 1 + d_C(D) \,\}$$

$d_B(D) = 1$, $d_C(D) = 4$

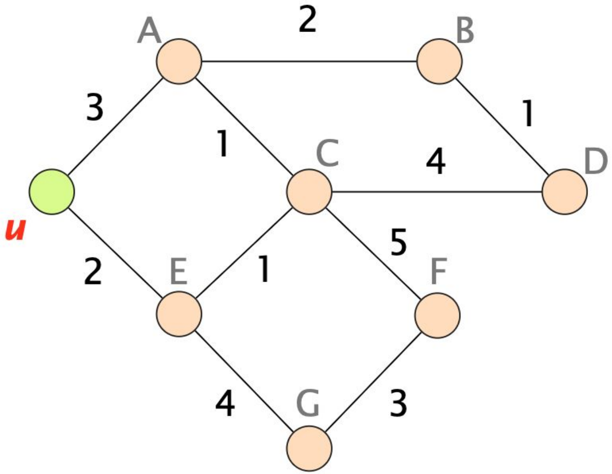
$= 3$

Any Time a Distance Vector Changes, Each Node Propagates it to its Neighbors



$$
\begin{aligned}
d_E(D) = \min \{ & 1 + d_C(D), \\
& 4 + d_G(D), \\
& 2 + d_u(D) \} \\
= 5 &
\end{aligned}
$$

The Process Eventually Converges to the Shortest Path Distance to Each Destination

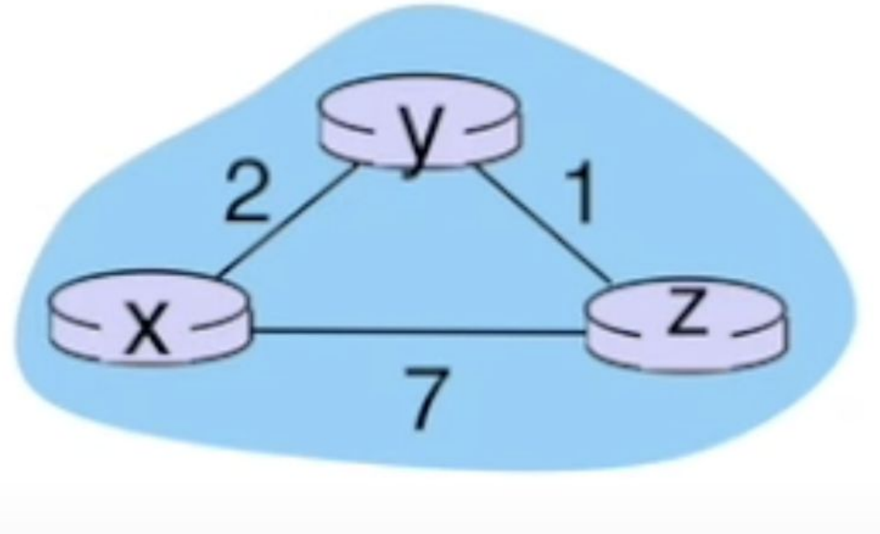


$$d_u(D) = \min \{ 3 + d_A(D), \; 2 + d_E(D) \}$$

$$= 6$$

Similar to LS Routing, u can Directly Create its Forwarding Table by Directing Traffic to the Best (whoever is advertising the lowest cost) Neighbor

DV Walkthrough



DV Walkthrough

Node X:

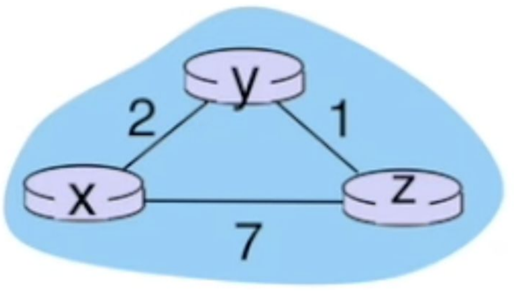
	X	Y	Z
X	0	2	7
Y	∞	∞	∞
Z	∞	∞	∞

Node Y:

	X	Y	Z
X	∞	∞	∞
Y	2	0	1
Z	∞	∞	∞

Node Z:

	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	7	1	0



DV Walkthrough

Node X:

	X	Y	Z
X	0	2	7
Y	∞	∞	∞
Z	∞	∞	∞

Node Y:

	X	Y	Z
X	∞	∞	∞
Y	2	0	1
Z	∞	∞	∞

Node Z:

	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	7	1	0

Node X:

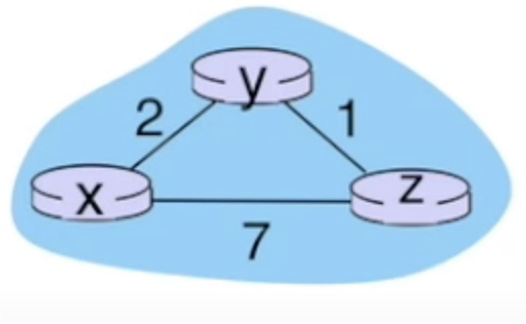
	X	Y	Z
X	0	2	**3**
Y	2	0	1
Z	7	1	0

Node Y:

	X	Y	Z
X	0	2	7
Y	2	0	1
Z	7	1	0

Node Z:

	X	Y	Z
X	0	2	7
Y	2	0	1
Z	**3**	1	0



DV Walkthrough

Node X:

	X	Y	Z
X	0	2	7
Y	∞	∞	∞
Z	∞	∞	∞

Node Y:

	X	Y	Z
X	∞	∞	∞
Y	2	0	1
Z	∞	∞	∞

Node Z:

	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	7	1	0

Node X:

	X	Y	Z
X	0	2	**3**
Y	2	0	1
Z	7	1	0

Node Y:

	X	Y	Z
X	0	2	7
Y	2	0	1
Z	7	1	0

Node Z:

	X	Y	Z
X	0	2	7
Y	2	0	1
Z	**3**	1	0

Node X:

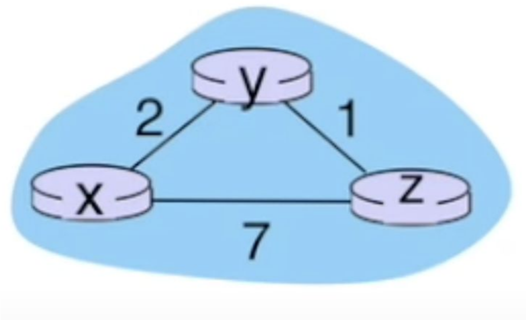
	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0

Node Y:

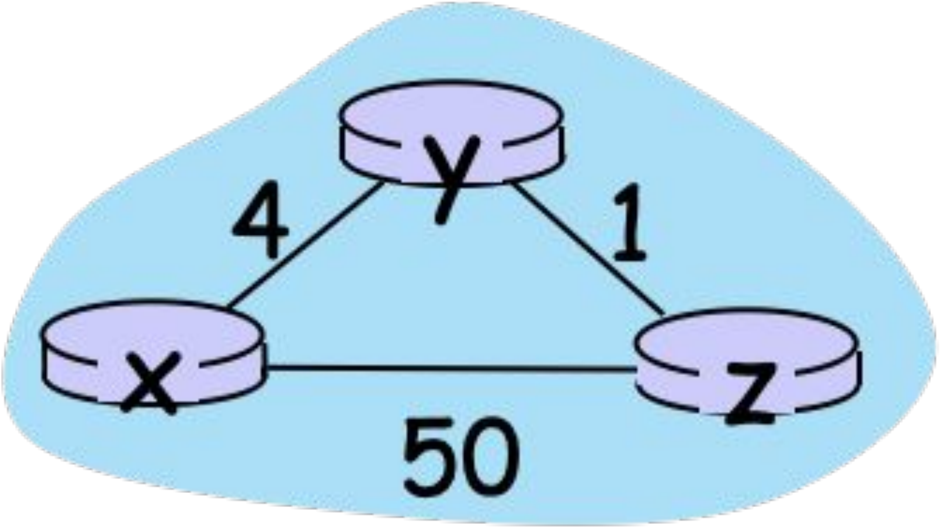
	X	Y	Z
X	0	2	7
Y	2	0	1
Z	7	1	0

Node Z:

	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0



DV



DV Solution

Node X:

	X	Y	Z
X	0	4	50
Y	∞	∞	∞
Z	∞	∞	∞

Node Y:

	X	Y	Z
X	∞	∞	∞
Y	4	0	1
Z	∞	∞	∞

Node Z:

	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	50	1	0

Node X:

	X	Y	Z
X	0	4	**5**
Y	4	0	1
Z	50	1	0

Node Y:

	X	Y	Z
X	0	4	50
Y	4	0	1
Z	50	1	0

Node Z:

	X	Y	Z
X	0	4	50
Y	4	0	1
Z	**5**	1	0

Node X:

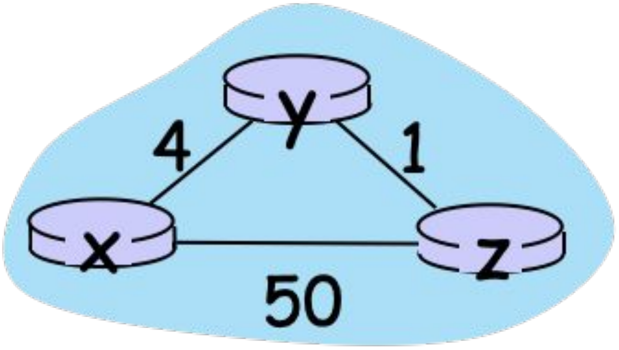
	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

Node Y:

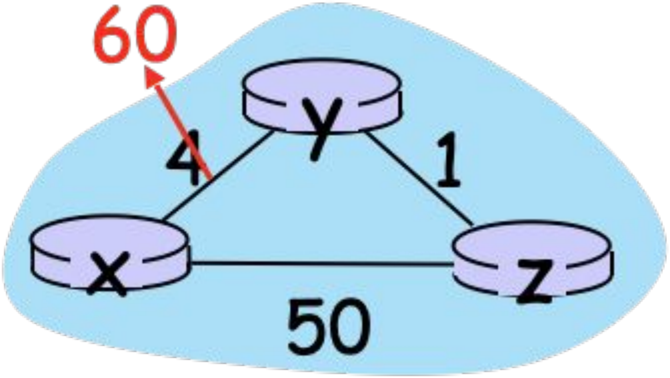
	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0



Distance Vector Suffers From the “Count to Infinity” Problem



Count to Infinity

Node X:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

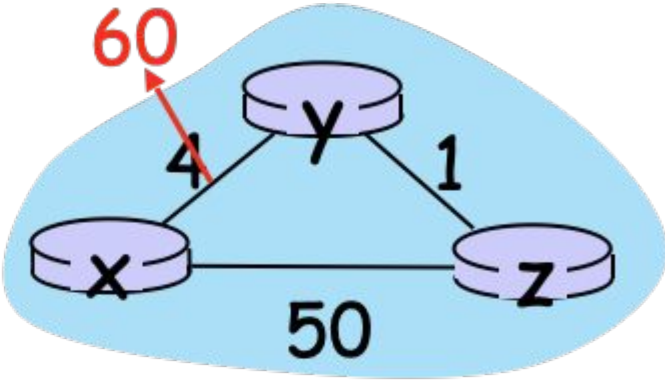
(Ignore X for simplicity)

Node Y:

	X	Y	Z
X	0	4	5
Y	**6**	0	1
Z	5	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0



Count to Infinity

Node X:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

(Ignore X for simplicity)

Node Y:

	X	Y	Z
X	0	4	5
Y	**6**	0	1
Z	5	1	0

Node Y:

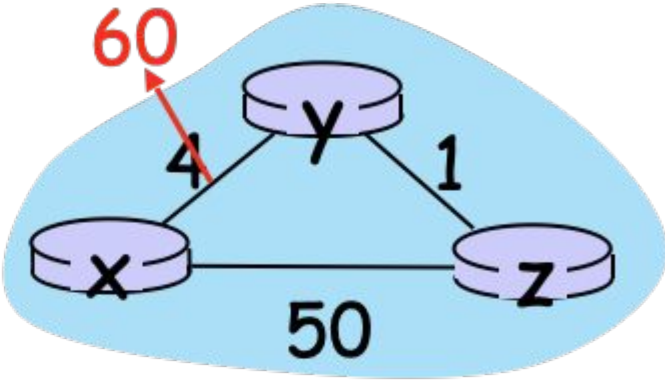
	X	Y	Z
X	0	4	5
Y	6	0	1
Z	5	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	6	0	1
Z	**7**	1	0



Count to Infinity

Node X:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

(Ignore X for simplicity)

Node Y:

	X	Y	Z
X	0	4	5
Y	**6**	0	1
Z	5	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

Node Y:

	X	Y	Z
X	0	4	5
Y	6	0	1
Z	5	1	0

Node Z:

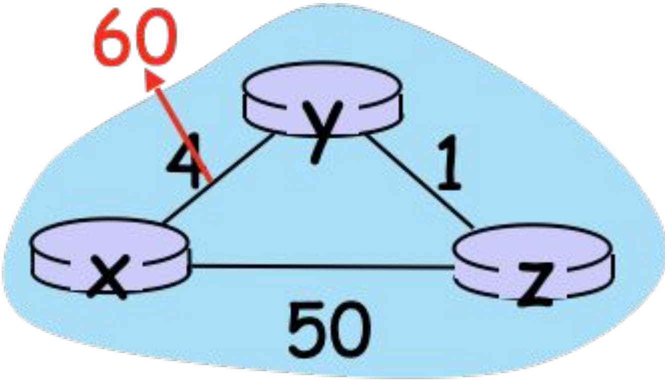
	X	Y	Z
X	0	4	5
Y	6	0	1
Z	**7**	1	0

Node Y:

	X	Y	Z
X	0	4	5
Y	**8**	0	1
Z	7	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	6	0	1
Z	7	1	0



Count to Infinity

Node X:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

(Ignore X for simplicity)

This will continue until they realize that X-Z is cheaper

Node Y:

	X	Y	Z
X	0	4	5
Y	**6**	0	1
Z	5	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

Node Y:

	X	Y	Z
X	0	4	5
Y	6	0	1
Z	5	1	0

Node Z:

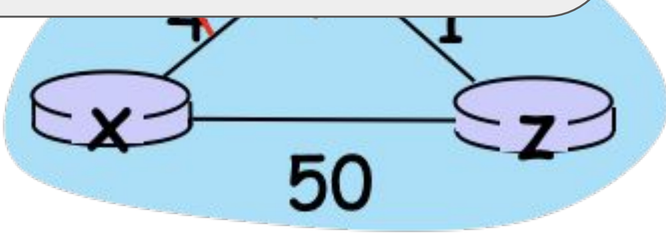
	X	Y	Z
X	0	4	5
Y	6	0	1
Z	**7**	1	0

Node Y:

	X	Y	Z
X	0		
Y	**8**	0	1
Z	7	1	0

Node Z:

	X	Y	Z
X	0	4	5
Y	6	0	1
Z	7	1	0



DV Routing

“Bad News Travels Slowly,
Good News Travels Fast”