Routing: Distance Vector Algorithm

Networking

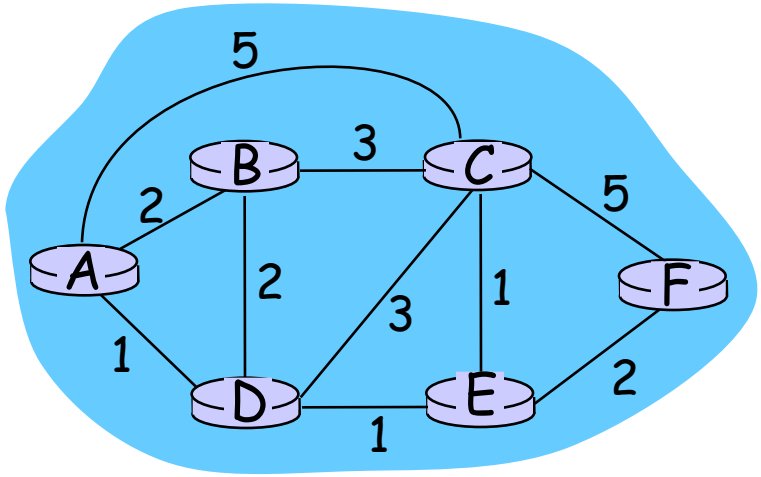
CS 3470, Section 1

Review

- So how is routing different from forwarding?

Routing

Network as a Graph



- The basic problem of routing is to find the lowest-cost path between any two nodes
  - Where the cost of a path equals the sum of the costs of all the edges that make up the path

Routing

- Routing protocols need to be both ***dynamic*** and ***distributed***
  - Deal with node and link failures, changing link costs, and scalability

Distributed Routing Algorithms

- Two standard distributed routing algorithms
  - ***Link state routing***
  - ***Distance vector routing***
- What link state routing protocol did we discuss last time?

Link State *vs* Distance Vector

- Both assume that
  - The address of each neighbor is known
  - The ***cost*** of reaching each neighbor is known
- Both find ***global*** information
  - By exchanging routing info among neighbors
- Differ in info exchanged and route computation
  - LS: tells ***every*** other node its ***distance*** to ***neighbors***
  - DV: tells ***neighbors*** its ***distance*** to ***every*** other node

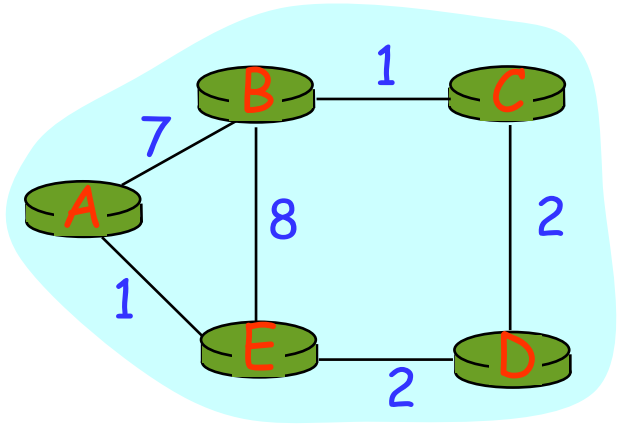
Distance Vector Routing

- A router tells neighbors its distance to every router
  - Communication between neighbors only
- Each router maintains a distance table
  - A row for each possible destination
  - A column for each neighbor
    - $D^X(Y,Z)$: distance from X to Y via Z

Distance Vector Routing

- Given a distance table we can find the shortest distance to a destination
  - i.e., the smallest value in the row corresponding to the destination.
- A list of <destination, shortest distance> tuples is called a ***distance vector***
  - Current least cost to each destination
  - Exchanged between neighbors
- Based on Bellman-Ford algorithm
  - Computes “shortest paths”

Distance Table: Example



Distance table at router E

$D^E()$	cost to destination via A	B	D
destination A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

- If E were to advertise its distance vector, it would send <A,1>,<B,5>,<C,4>,<D,2>

Distance Table to Routing Table

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

Outgoing link to use, cost

destination	
A	A,1
B	D,5
C	D,4
D	D,2

Distance table → Routing table

Distance Vector Routing Algorithm

- Now the real question…
- How do we compute this distance table?
- That's where we use Bellman-Ford algorithm.

Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info/iterate in lock step!

distributed:

- each node talks *only* with directly-attached neighbors

Distance Table data structure

- each node has its own
- row for each possible destination
- column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance } \textit{from } X \textit{ to } Y, \textit{ via } Z \text{ as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Distance Vector Routing: Overview

Iterative, asynchronous:

each iteration caused by:

- local link cost change
- message from neighbor: its least cost path change from neighbor

Distributed:

- each node notifies neighbors *only* when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

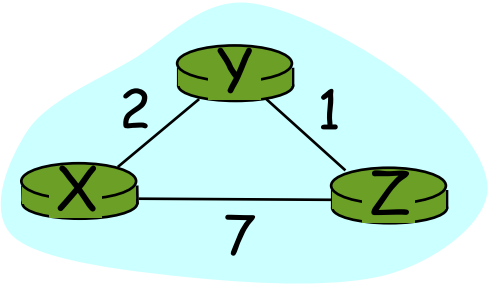
Each node:

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

recompute distance table

if least cost path to any dest has changed, *notify* neighbors

Distance Vector Algorithm: Example



D^X

dest	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y

dest	cost via X	cost via Z
X	2	∞
Z	∞	1

D^Z

dest	cost via X	cost via Y
X	7	∞
Y	∞	1

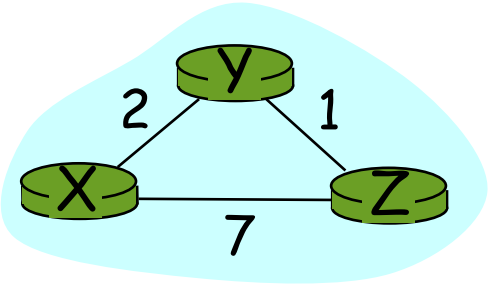
D^X

dest	cost via Y	cost via Z
Y	2	8
Z	3	7

$$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w\{D^Y(Z,w)\}$$
$$= 2+1 = 3$$

Distance Vector Algorithm: Example



X	cost via Y	cost via Z
dest Y	2	∞
dest Z	∞	7

X	cost via Y	cost via Z
dest Y	2	8
dest Z	3	7

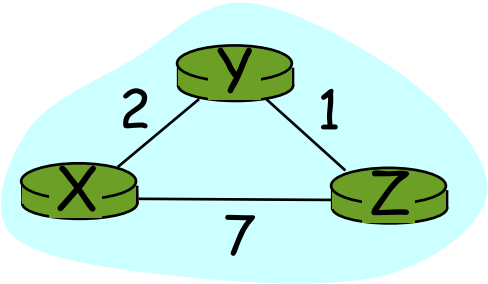
Y	cost via X	cost via Z
dest X	2	∞
dest Z	∞	1

Y	cost via X	cost via Z
dest X		
dest Z		

Z	cost via X	cost via Y
dest X	7	∞
dest Y	∞	1

Z	cost via X	cost via Y
dest X		
dest Y		

Distance Vector Algorithm: Example



X	cost via Y	cost via Z
dest Y	2	∞
dest Z	∞	7

X	cost via Y	cost via Z
dest Y	2	8
dest Z	3	7

X	cost via Y	cost via Z
dest Y		
dest Z		

Y	cost via X	cost via Z
dest X	2	∞
dest Z	∞	1

Y	cost via X	cost via Z
dest X	2	8
dest Z	9	1

Y	cost via X	cost via Z
dest X		
dest Z		

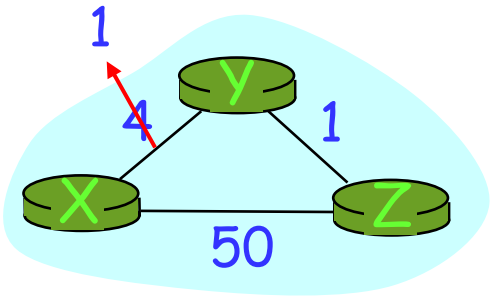
Z	cost via X	cost via Y
dest X	7	∞
dest Y	∞	1

Z	cost via X	cost via Y
dest X	7	3
dest Y	9	1

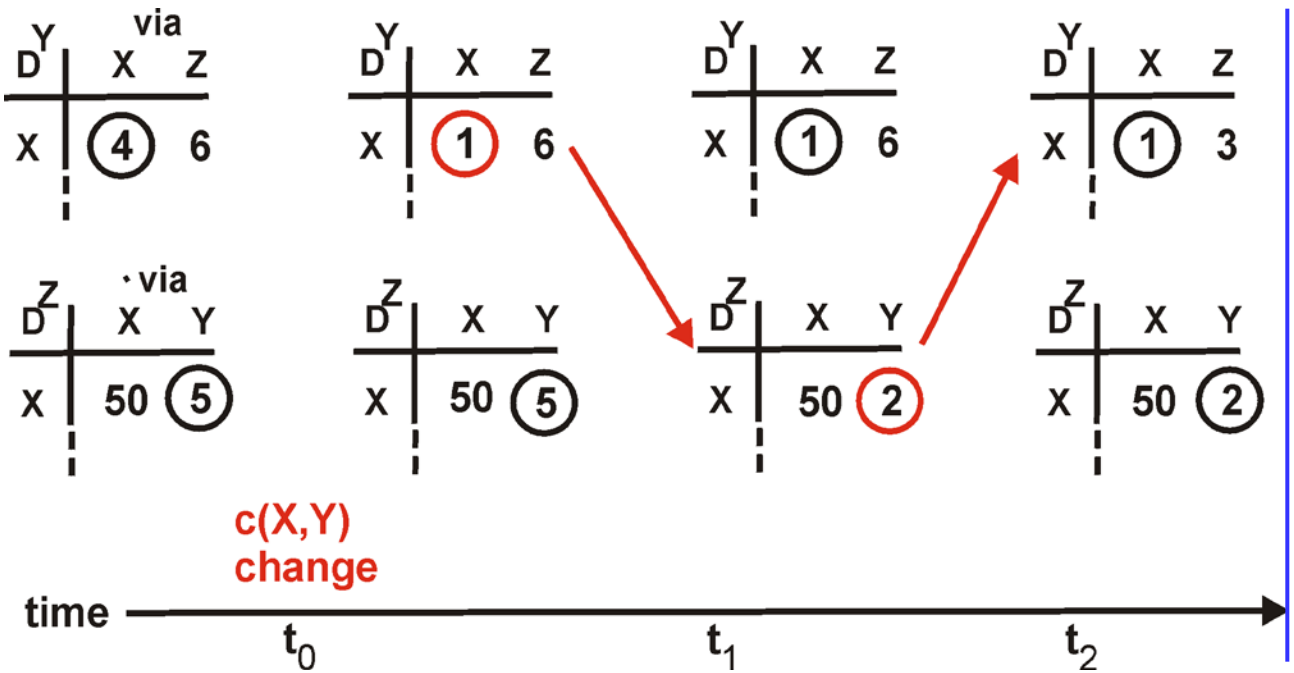
Z	cost via X	cost via Y
dest X		
dest Y		

Convergence of DV Routing

router detects local link cost change
updates distance table
if cost change in least cost path, notify neighbors



"good news travels fast"



algorithm terminates

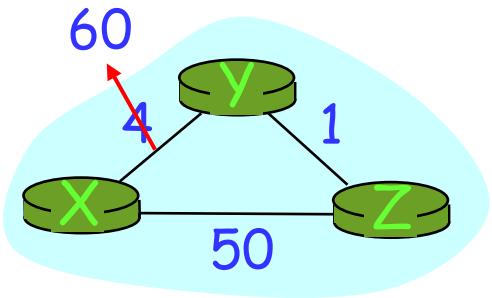
Problems with DV Routing

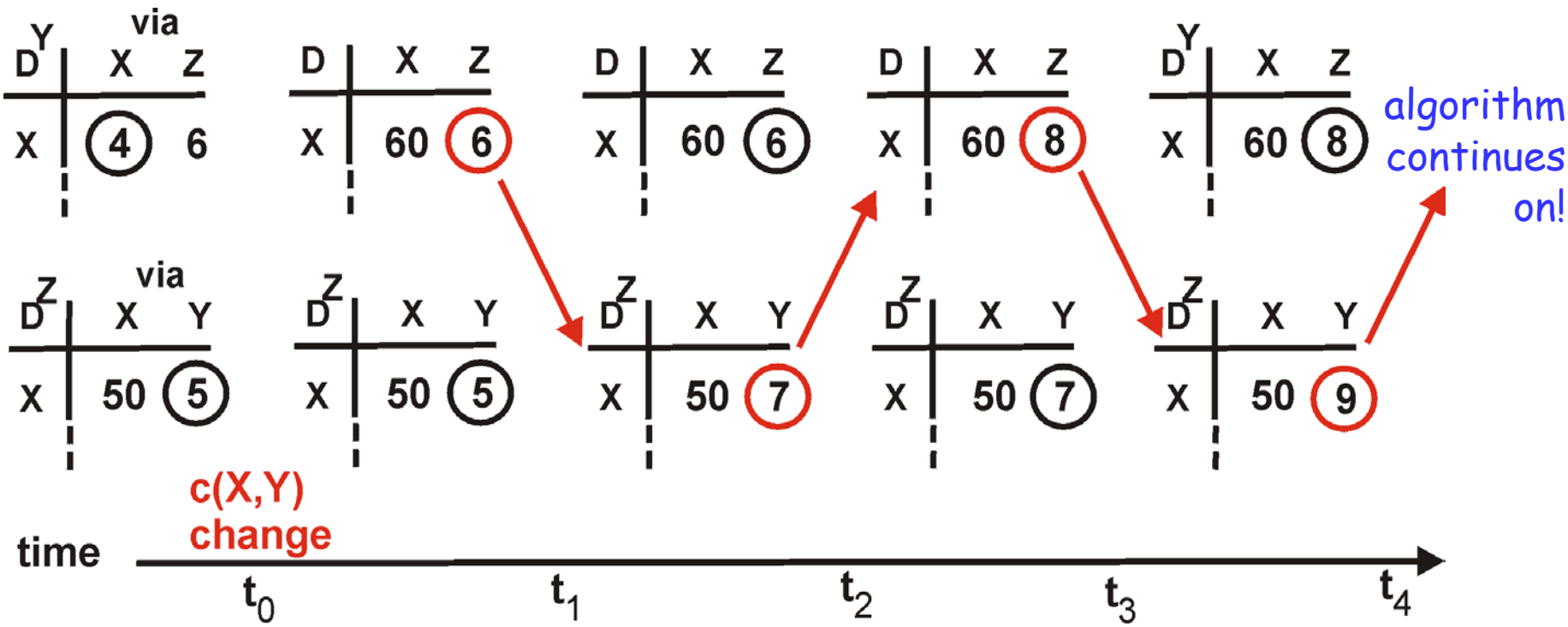
Link cost changes:

good news travels fast

bad news travels slow

"count to infinity" problem!



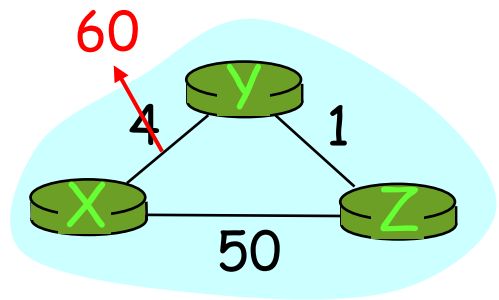


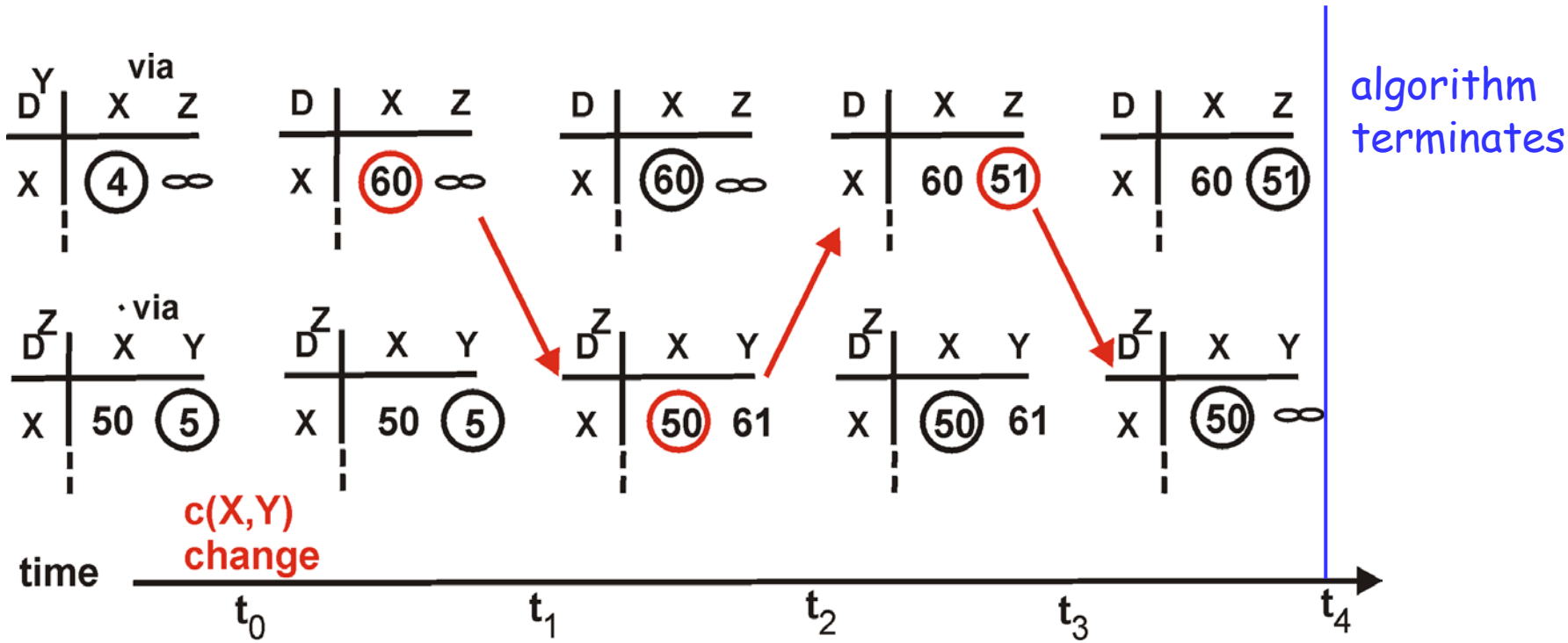
Fixes to Count-to-Infinity Problem

- Split horizon
  - A router never advertises the cost of a destination to a neighbor
    - If this neighbor is the next hop to that destination
- Split horizon with poisonous reverse
  - If X routes traffic to Z via Y, then
    - X tells Y that its distance to Z is infinity
      - Instead of not telling anything at all
  - Accelerates convergence

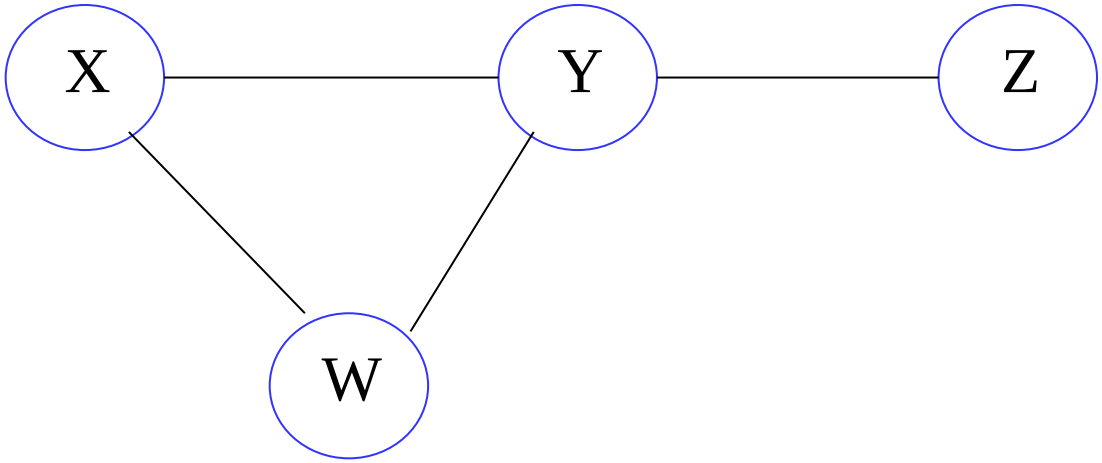
Split Horizon with Poisoned Reverse

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)



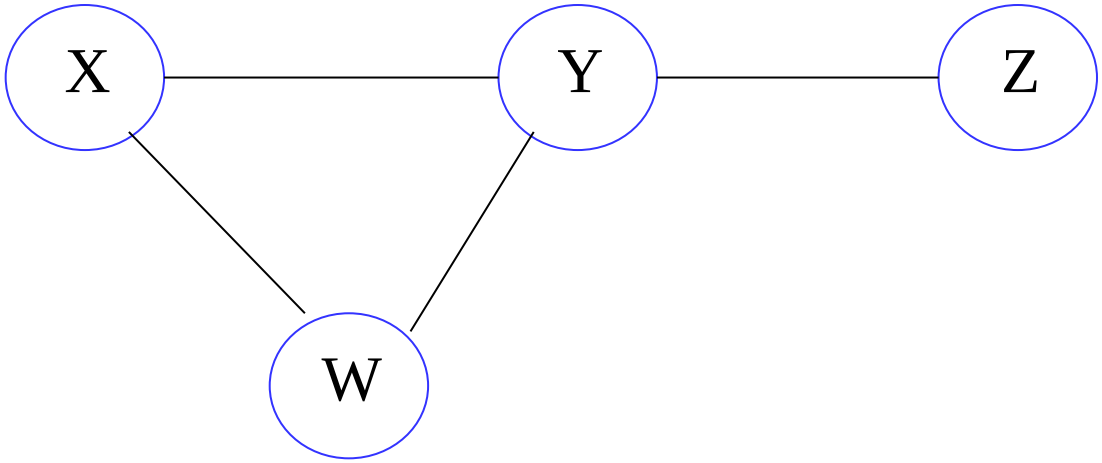


Count-to-Infinity Problem



- What happens when the link Y to Z goes down?
- All three routers X, Y, and W together count to infinity.
- Split horizon solution works only when two routers are involved in a loop.

Count-to-Infinity Problem



- To completely eliminate the problem, a router some how need to figure out the complete path to a destination.
- Obvious solution is to pass on the path information along with the distance vector.
  - This path vector approach is used in BGP.

Link State *vs* Distance Vector

- Tells everyone about neighbors
- Controlled flooding to exchange link state
- Dijkstra's algorithm
- Each router computes its own table
- May have oscillations

- Tells neighbors about everyone
- Exchanges distance vectors with neighbors
- Bellman-Ford algorithm
- Each router's table is used by others
- May have routing loops

RIP

- RIP == Routing Information Protocol
- RIP is a *distance vector* implementation

(network_address, distance) pairs

- Instead of advertising costs to the next router, RIP advertises the cost to the next *network*.