CS348: Computer Networks

Routing Algorithms

Dr. Manas Khatua
Assistant Professor
Dept. of CSE, IIT Guwahati
E-mail: manaskhatua@iitg.ac.in

Introduction

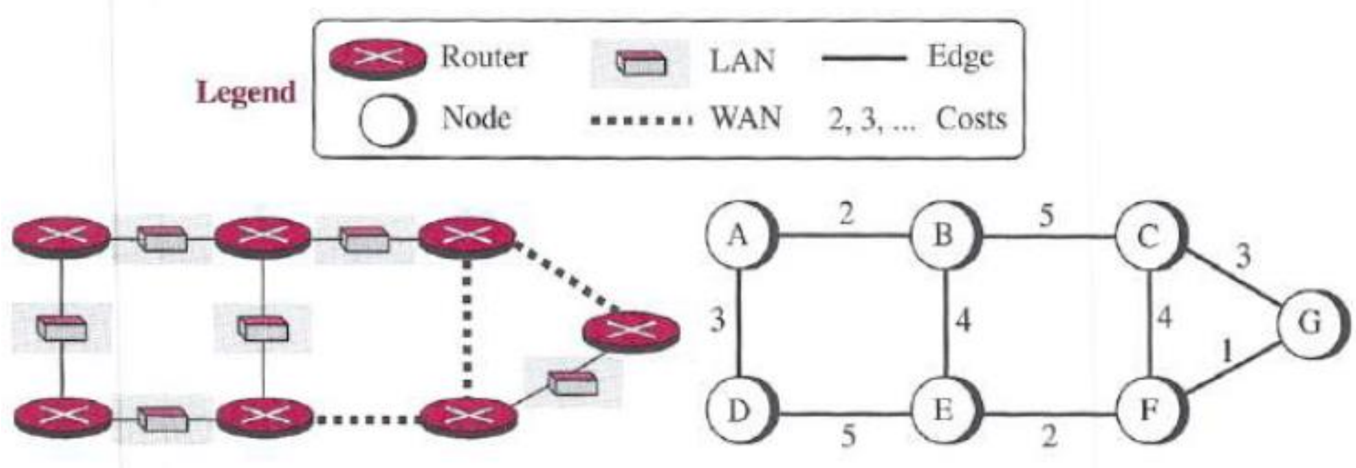
- Goal of the Network Layer is
  - deliver a datagram from its source to its destination.
- Network Layer determines the path
  - to deliver packet from source host to destination host,
  - irrespective of data forwarding service type (datagram / virtual-circuit).

Routing

boils down to

Finding path from **source router** to **destination router**

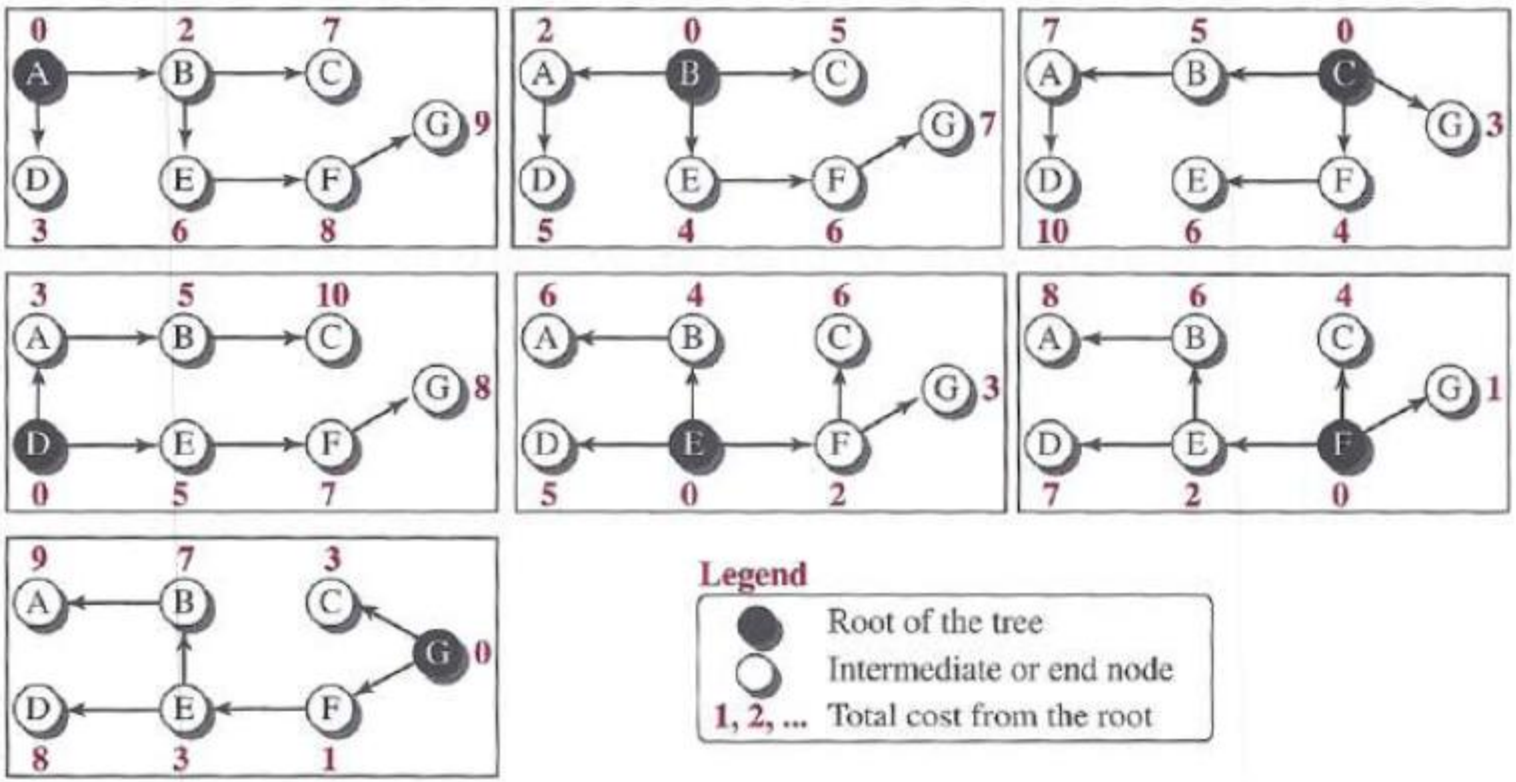
- Treat the Internet as a Graph



a. An internet

b. The weighted graph

Least cost routing



- one of the ways to interpret the *best route* from the source router to the destination router is to find the *least cost path* between the two.
  - Least cost path may not be shortest path

Types of Routing Algorithms

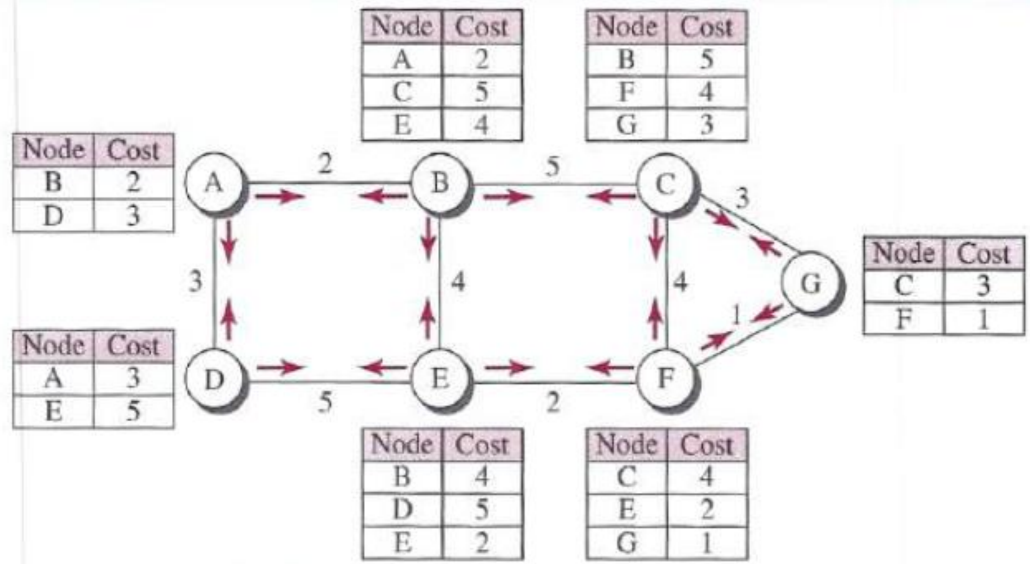
- **One way** to classify:
  - Global routing algo:
    - computes least-cost path using complete, global knowledge about the graph (i.e. network)
    - e.g., Link-State (LS) algorithm
  - Decentralized routing algo:
    - calculation of the least-cost path is carried out in an iterative, distributed manner. No node has complete information about the graph (i.e. network)
    - e.g., Distance-Vector (DV) algorithm

- **Second way** to classify:
  - Static routing algo:
    - routes change very slowly over time, often by human intervention
  - Dynamic routing algo:
    - change the routing paths with the network traffic loads or topology change.

- **Third way** to classify:
  - Load-sensitive routing algo:
    - choose path using link cost
    - e.g., ARPAnet routing algorithms
  - Load-insensitive routing algo:
    - Link cost does not play any role in path selection
    - e.g., Today's Internet routing algorithms (BGP, RIP, OSPF)

Link-State Routing

- Network topology and all link states are known to each node.
  - the cost associated with an edge defines the state of the link

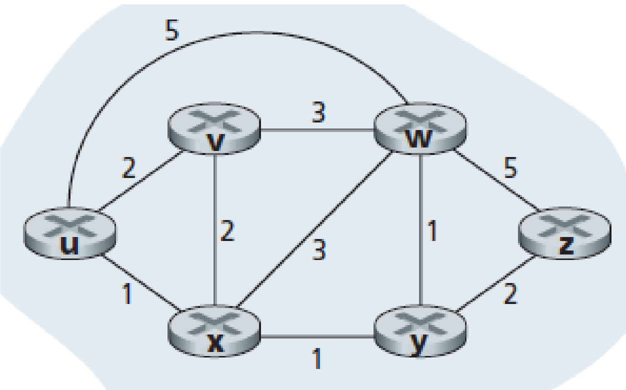


- How to achieve?
  - a router continuously tells all nodes what it knows about the neighbours
  - Each node broadcast link-state packets containing the identities and costs of its attached links
  - This is done by link-state broadcast scheme
  - Then, all nodes will have complete and identical view of the network
  - Finally, each node run link-state routing algorithm to compute the same set of least-cost paths
  - Mostly used link-state algorithm is Dijkstra algorithm

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Link-State Algorithm

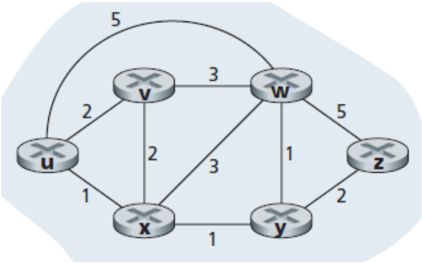


- ***D*(*v*)**: cost of the least-cost path from source node to destination *v*
- ***p(v)***: previous node (neighbor of *v*) along the current least-cost path from the source to *v*
- ***N'*** : subset of nodes

Link-State (LS) Algorithm for Source Node *u*

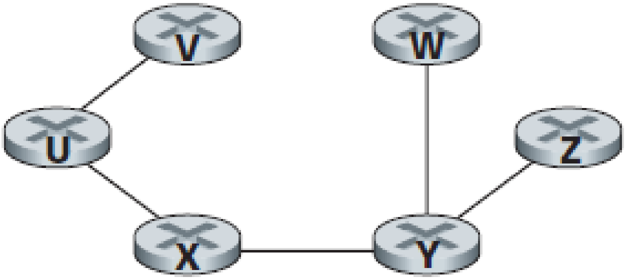
```
Initialization:
  N' = {u}
  for all nodes v
    if v is a neighbor of u
      then D(v) = c(u,v)
    else D(v) = ∞

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

Cont…



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					

Table 4.3 ♦ Running the link-state algorithm on the network in Figure 4.27



Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Figure 4.28 ♦ Least cost path and forwarding table for nodule u

Complexity Analysis

- Given ***n*** nodes (not counting the source), how much computation must be done in the worst case to find the least-cost paths from the source to all destinations?

- 1st iteration: we need to search through all n nodes to determine the node, *w,* not in N that has the minimum cost.
- 2nd iteration: we need to check $n - 1$ nodes to determine the minimum cost.
- 3rd iteration: need $n - 2$ nodes, and
- So on.

- The **total** number of nodes we need to **search** through over all the iterations is $n(n + 1)/2$,
- We say that the *preceding implementation of the LS algorithm* has worst-case complexity $O(n^2)$.

- Note: using heap data structure, the complexity could be reduced.

Routing Oscillation

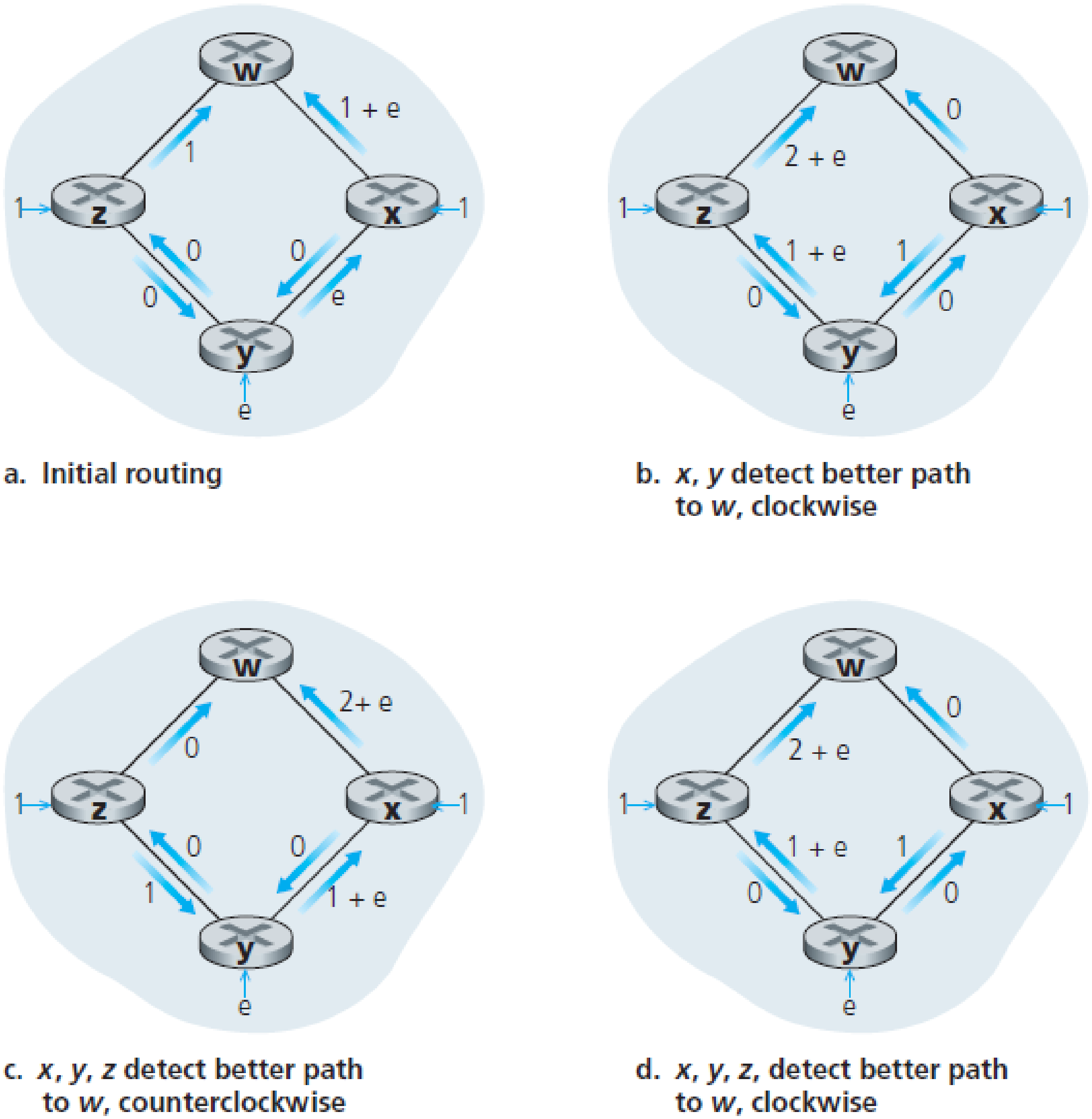


Figure 4.29 ♦ Oscillations with congestion-sensitive routing

Condition to occur:

- LS algorithm that uses load or congestion or delay-based link metric
- Link costs are not symmetric in both directions

This example:

- x, y, and z inject 1, e, and 1 unit of traffic respectively destined for w
- Link cost = traffic load on the link

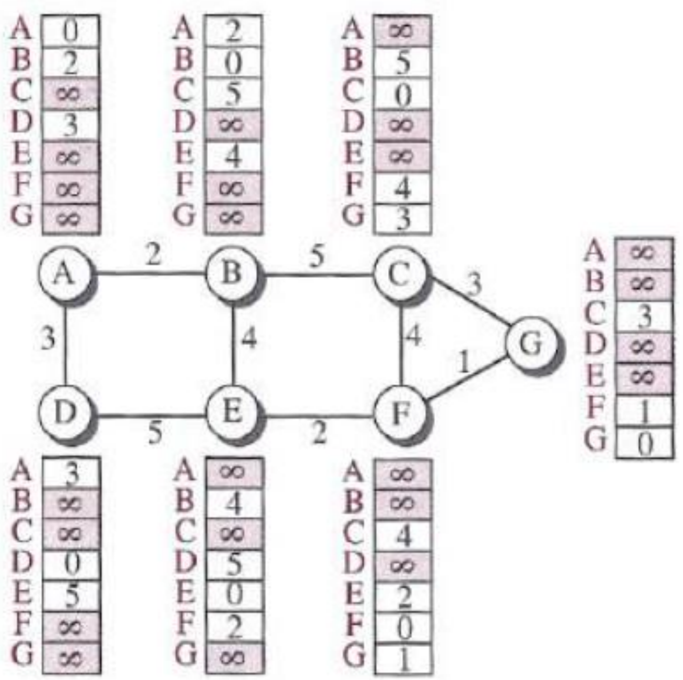
Solution:

- Ensure that not all routers run the LS algorithm at the same time

Distance Vector Routing

- Network topology and all link states are not known to any node.
  - the cost associated with an edge defines the state of the link
  - least-cost path is carried out in an iterative, asynchronous, and distributed manner

- How to achieve?
  - a router tells all of its neighbours what it knows about the whole internet
  - A node knows distance vector: one-dimensional array to represent the least-cost tree
  - Each node updates its distance-vector estimate when it
  - 1) either sees a cost change in one of its directly attached links
  - 2) or receives a distance vector update from some neighbour
  - Each node then distributes its new distance vector to its neighbors
  - Then, all nodes will have complete distance vector for the network starting from itself
  - Each node run Belman-Ford algorithm to update the vector
  - Bellman-Ford équation: $D_x(y) = \min_v\{ c(x,v) + D_v(y) \}$
    - $D_x(y)$ be the cost of the least-cost path from node *x* to node *y*.



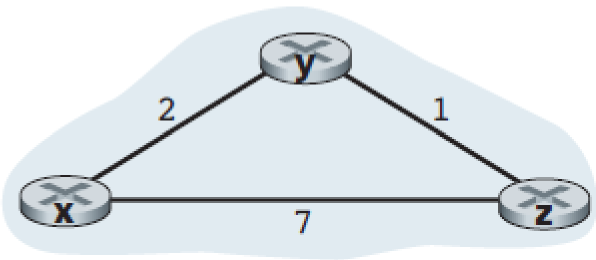
Distance-Vector Algorithm

Distance-Vector (DV) Algorithm

At each node, *x*:

```
Initialization:
   for all destinations y in N:
      D_x(y) = c(x,y)   /* if y is not a neighbor then c(x,y) = ∞ */
   for each neighbor w
      D_w(y) = ? for all destinations y in N
   for each neighbor w
      send distance vector 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 D_x = [D_x(y): y in N] to all neighbors

forever
```

Cont…



Node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node y table

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

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

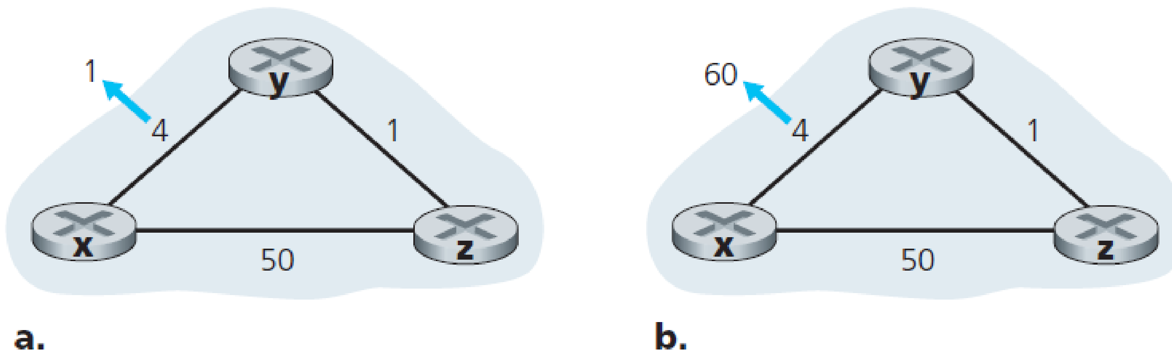
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

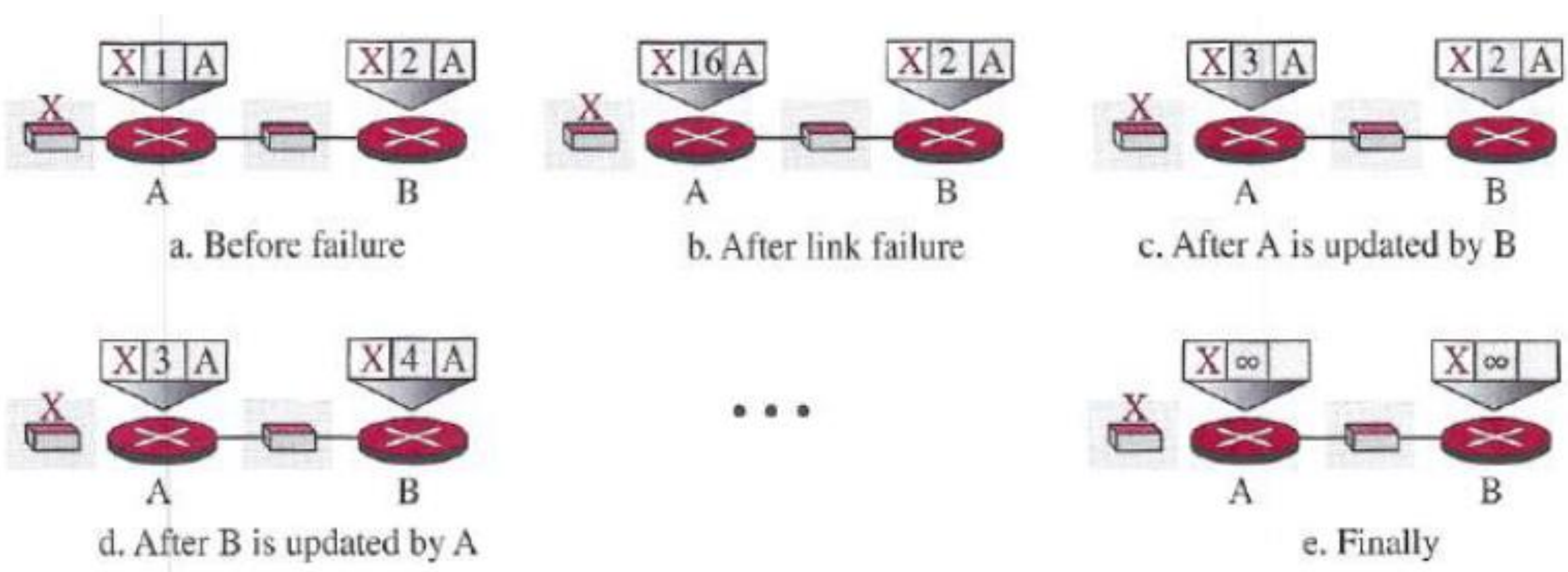
Time →

Routing Loop Problem

Loop for a while in case b:



Loop to Infinity:



Cont…

- Solutions:
  - Split Horizon: For routers to send information only to the neighbors that are not exclusive links to the destination.
    - Route deleted problem due to timer
  - Poison Reverse: “Do not use this value; what I know about this route comes from you”
    - The idea is simple — (in Fig b) if z routes through y to get to destination x, then z will advertise to y that its distance to x is infinity, that is, z will advertise to y that $D_z(x) = \infty$ (even though z knows $D_z(x) = 5$ in truth).

DV vs LS Routing

- *Message complexity:*
  - LS requires each node to know the cost of each link in the network.
  - This requires O(N*E) messages to be sent.
  - Also, whenever a link cost changes, the new link cost must be sent to all nodes.

  - The DV algorithm requires message exchanges between directly connected neighbors at each iteration.
  - Also, whenever a link cost changes, the new link cost vector must be sent to all neighbors if it changes least-cost path to a node.

- *Speed of convergence:*
  - Implementation of LS is an $O(N^2)$ algorithm requiring O(N*E) messages.
  - The DV algorithm can converge slowly and can have routing loops while the algorithm is converging.
  - The DV also suffers from the count-to-infinity problem.

- *Robustness:* What can happen if a router fails, misbehaves, or is sabotaged?
  - an LS node is computing only its own forwarding tables
  - This means route finding are somewhat separated under LS, providing a degree of robustness.
  - Under DV, a node can advertise incorrect least-cost paths to any or all destinations.
  - at each iteration, a node's calculation in DV is passed on to its neighbor and then indirectly to its neighbor's neighbor
  - In this sense, an incorrect node calculation can be diffused through the entire network under DV.

Thanks!