

# Lecture 16:

# Distance Vector Routing

---

CSE 123: Computer Networks

Aaron Schulman





# Lecture 16 Overview

---

- Wrap up discussion of link state routing
  - Convergence delay and how to mitigate it
- Distance vector routing

# Link State Convergence Delay

---



- Sources of convergence delay
  - ◆ Detection latency
  - ◆ Flooding of link-state information
  - ◆ Shortest-path computation
  - ◆ Creating the forwarding table
- Poor performance during convergence period
  - ◆ Lost packets due to black holes and TTL expiry
  - ◆ Looping packets consuming resources
  - ◆ Out-of-order packets reaching the destination
- Very bad for VoIP, online gaming, and video



# Reducing Delay

---

- Faster detection
  - ◆ Smaller LSP timers
  - ◆ Link-layer technologies that can detect failures
- Faster flooding
  - ◆ Flooding immediately
  - ◆ Sending link-state packets with high-priority
- Faster computation
  - ◆ Faster processors on the routers
  - ◆ Incremental Dijkstra's algorithm
- Faster forwarding-table update
  - ◆ Data structures supporting incremental updates



# Real Link-state Protocols

---

- OSPF (Open Shortest Path First) and IS-IS
  - ◆ Most widely used intra-domain routing protocols
  - ◆ Run by almost all ISPs and many large organizations
- Basic link state algorithm plus many features:
  - ◆ Authentication of routing messages
  - ◆ Extra hierarchy: Partition into routing areas
    - » “Border” router pretends to be directly connected to all routers in an area (answers for them)
  - ◆ Load balancing: Multiple equal cost routes



# Link State evaluation

---

- Strengths
  - ◆ Loop free as long as LS database's are consistent
    - » Can have transient routing loops – shouldn't last long
  - ◆ Messages are small
  - ◆ Converges quickly
  - ◆ Guaranteed to converge
- Weaknesses
  - ◆ Must flood data across entire network (scalability?)
  - ◆ Must maintain state for entire topology (database)



# Distance vector algorithm

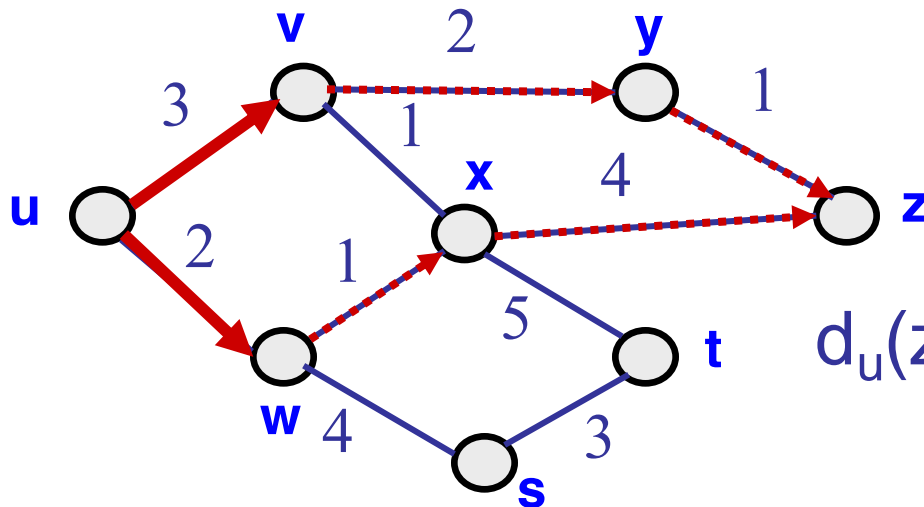
---

- Base assumption
  - ◆ Each router knows its **own address** and the cost to reach each of its **directly connected neighbors**
- Bellman-Ford algorithm
  - ◆ Distributed route computation using **only neighbor's info**
  - ◆ **No flooding like in link-state routing**
- Mitigating loops
  - ◆ Split horizon and poison reverse



# Bellman-Ford Algorithm

- Define distances at each node  $X$ 
  - ◆  $d_x(y)$  = cost of least-cost *path* from  $X$  to  $Y$
- Update distances based on neighbors
  - ◆  $d_x(y) = \min \{c(x,v) + d_v(y)\}$  over all neighbors  $V$



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



# Distance Vector Algorithm

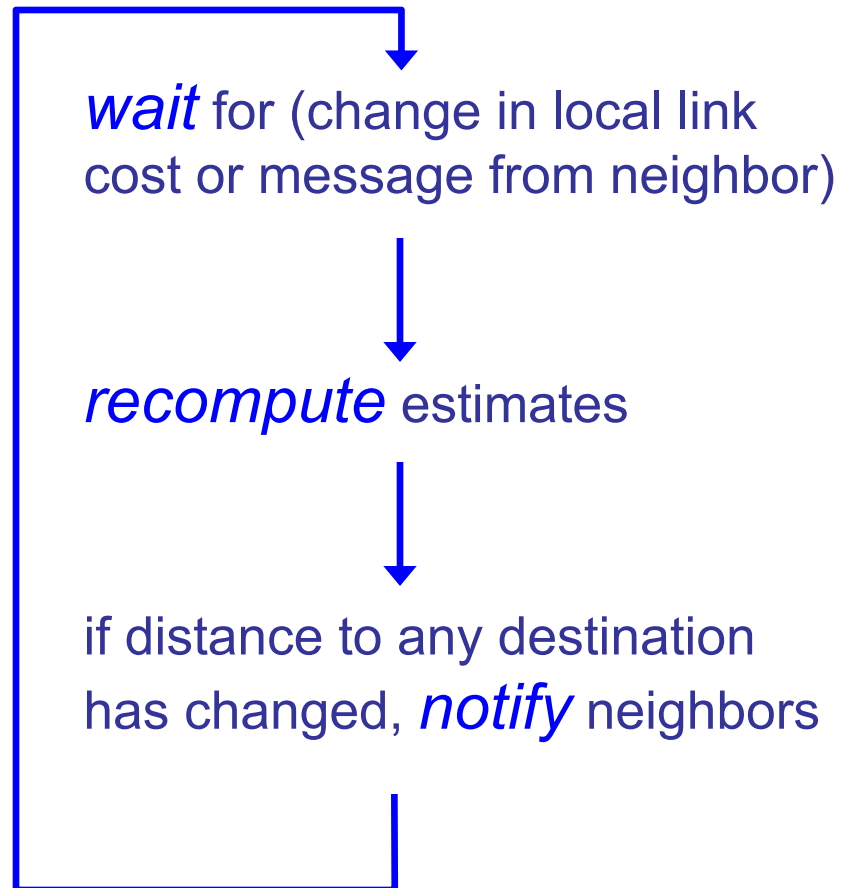
Iterative, asynchronous: each local iteration caused by:

- Local link cost change
- Distance vector update message from neighbor

Distributed:

- Each node notifies neighbors when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:



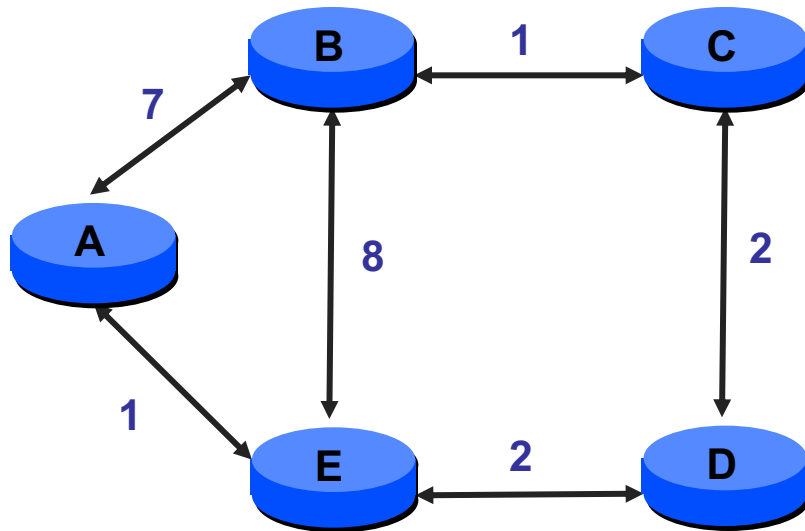


# Step-by-Step

- $c(x, v)$  = cost for direct link from  $x$  to  $v$ 
  - ◆ Node  $x$  maintains costs of direct links  $c(x, v)$
- $D_x(y)$  = estimate of least cost from  $x$  to  $y$ 
  - ◆ Node  $x$  maintains distance vector  $D_x = [D_x(y): y \in N]$
- Node  $x$  maintains its neighbors' distance vectors
  - ◆ For each neighbor  $v$ ,  $x$  maintains  $D_v = [D_v(y): y \in N]$
- Each node  $v$  periodically sends  $D_v$  to its neighbors
  - ◆ And neighbors update their own distance vectors
  - ◆  $D_x(y) \leftarrow \min_v \{c(x, v) + D_v(y)\}$  for each node  $y \in N$



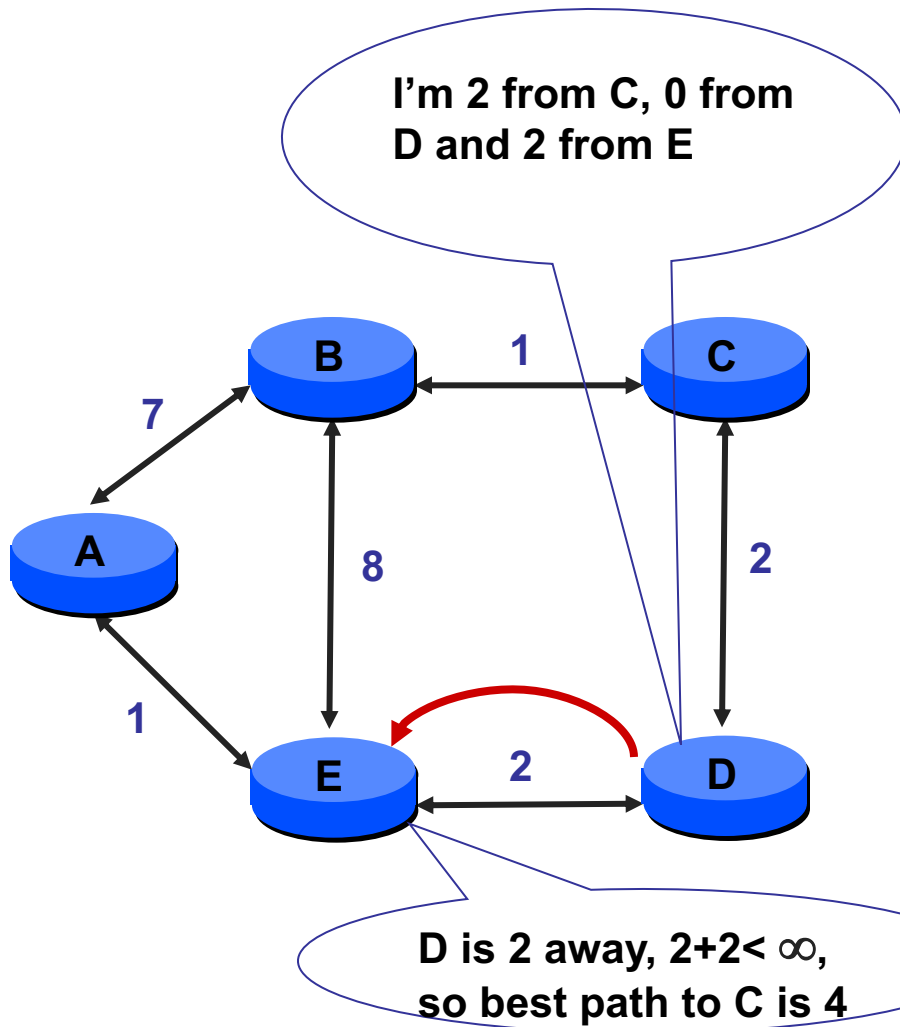
# Example: Initial State



| Info at node | Distance to Node |          |          |          |          |
|--------------|------------------|----------|----------|----------|----------|
|              | A                | B        | C        | D        | E        |
| A            | 0                | 7        | $\infty$ | $\infty$ | 1        |
| B            | 7                | 0        | 1        | $\infty$ | 8        |
| C            | $\infty$         | 1        | 0        | 2        | $\infty$ |
| D            | $\infty$         | $\infty$ | 2        | 0        | 2        |
| E            | 1                | 8        | $\infty$ | 2        | 0        |



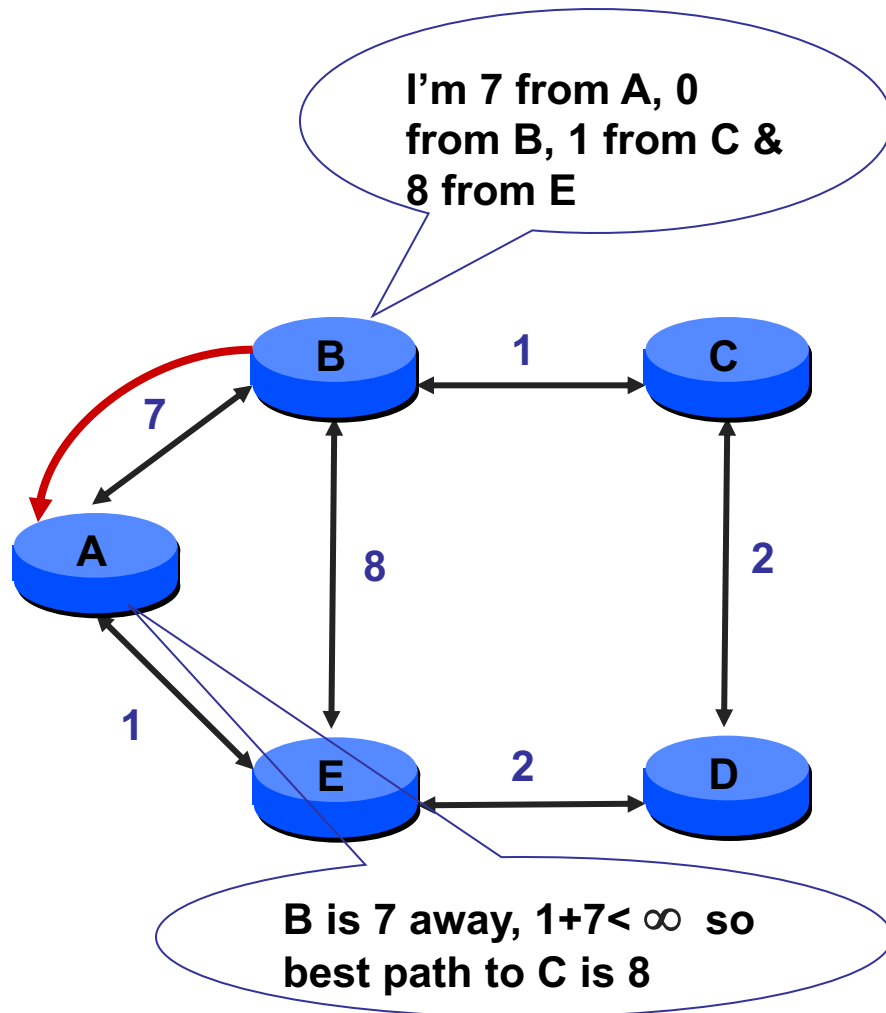
# *D* sends vector to *E*



| Info at node | Distance to Node |          |          |          |          |
|--------------|------------------|----------|----------|----------|----------|
|              | A                | B        | C        | D        | E        |
| A            | 0                | 7        | $\infty$ | $\infty$ | 1        |
| B            | 7                | 0        | 1        | $\infty$ | 8        |
| C            | $\infty$         | 1        | 0        | 2        | $\infty$ |
| D            | $\infty$         | $\infty$ | 2        | 0        | 2        |
| E            | 1                | 8        | 4        | 2        | 0        |



# B sends vector to A

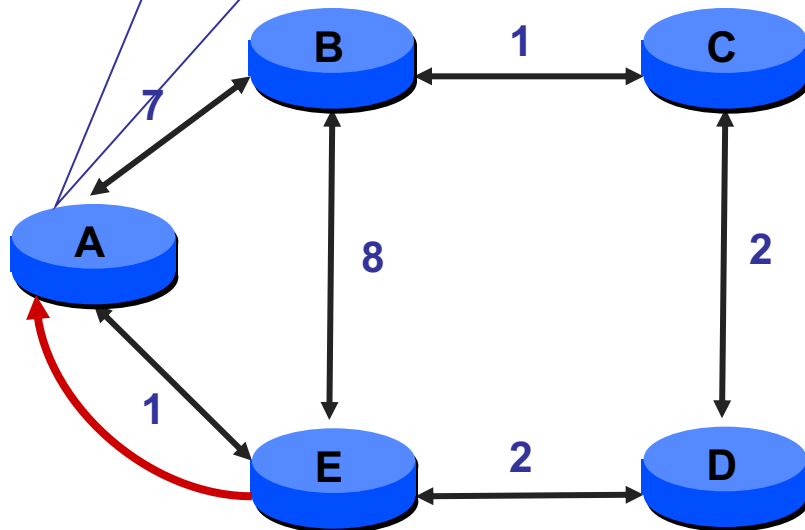


| Info at node | Distance to Node |          |   |          |          |
|--------------|------------------|----------|---|----------|----------|
|              | A                | B        | C | D        | E        |
| A            | 0                | 7        | 8 | $\infty$ | 1        |
| B            | 7                | 0        | 1 | $\infty$ | 8        |
| C            | $\infty$         | 1        | 0 | 2        | $\infty$ |
| D            | $\infty$         | $\infty$ | 2 | 0        | 2        |
| E            | 1                | 8        | 4 | 2        | 0        |



# *E* sends vector to *A*

E is 1 away,  $4+1 < 8$   
so C is 5 away,  $1+2 < \infty$   
so D is 3 away

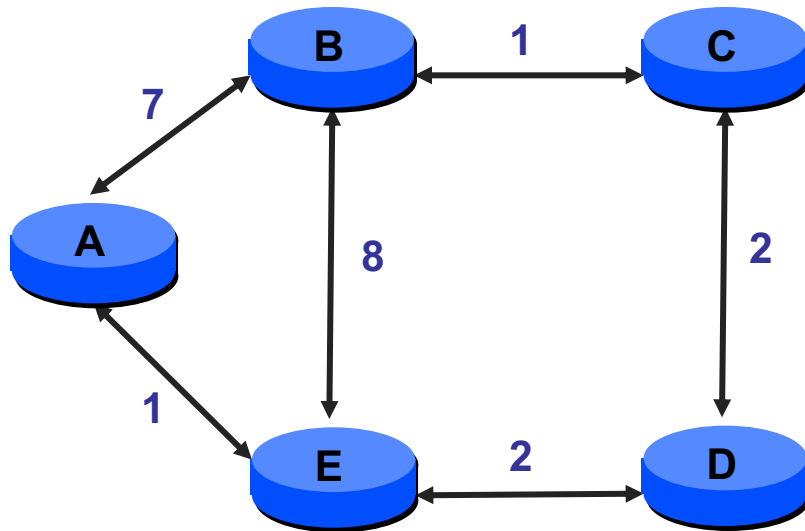


I'm 1 from A, 8 from B, 4 from C, 2 from D & 0 from E

| Info at node | Distance to Node |          |   |          |          |
|--------------|------------------|----------|---|----------|----------|
|              | A                | B        | C | D        | E        |
| A            | 0                | 7        | 5 | 3        | 1        |
| B            | 7                | 0        | 1 | $\infty$ | 8        |
| C            | $\infty$         | 1        | 0 | 2        | $\infty$ |
| D            | $\infty$         | $\infty$ | 2 | 0        | 2        |
| E            | 1                | 8        | 4 | 2        | 0        |



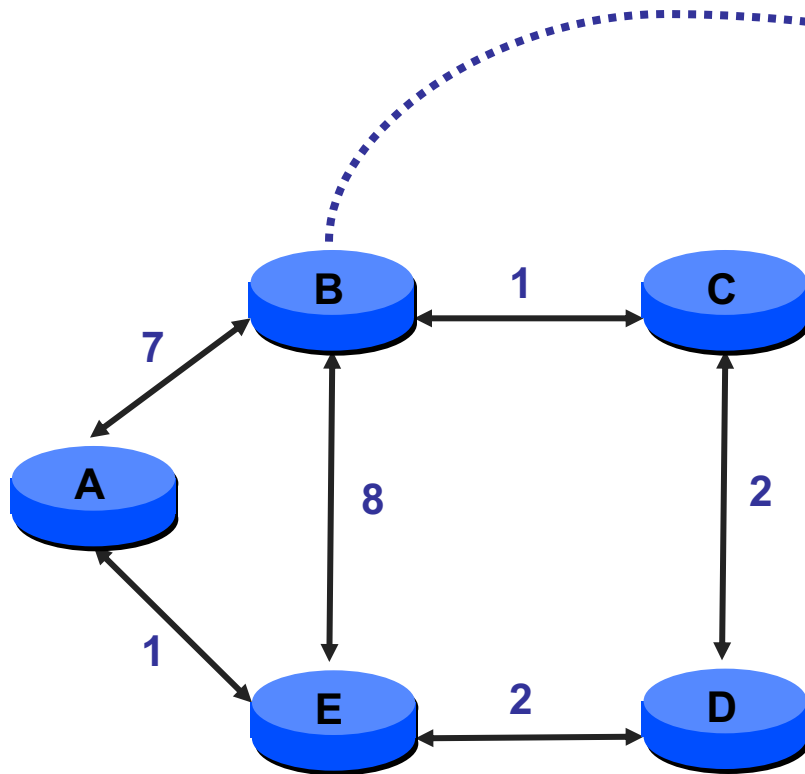
# ...until Convergence



| Info at node | Distance to Node |   |   |   |   |
|--------------|------------------|---|---|---|---|
|              | A                | B | C | D | E |
| A            | 0                | 6 | 5 | 3 | 1 |
| B            | 6                | 0 | 1 | 3 | 5 |
| C            | 5                | 1 | 0 | 2 | 4 |
| D            | 3                | 3 | 2 | 0 | 2 |
| E            | 1                | 5 | 4 | 2 | 0 |



# Node *B*'s distance vectors

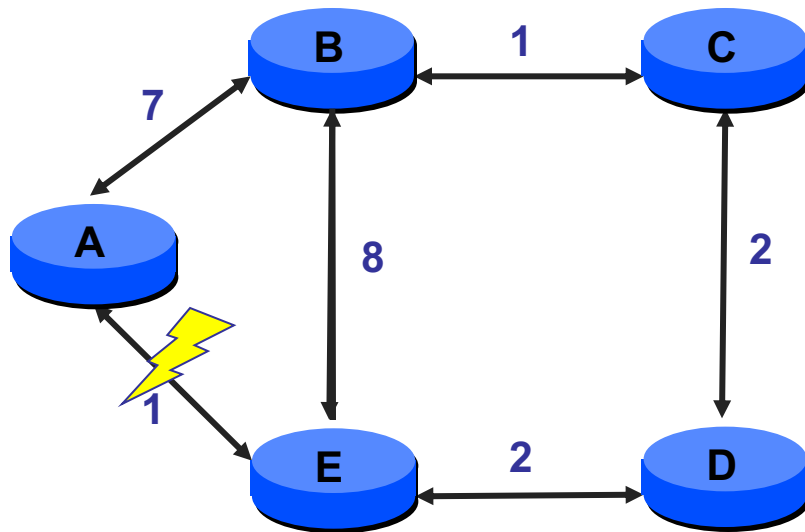


| Dest | Next hop |    |          |
|------|----------|----|----------|
|      | A        | E  | C        |
| A    | 7        | 9  | <b>6</b> |
| C    | 12       | 12 | <b>1</b> |
| D    | 10       | 10 | <b>3</b> |
| E    | 8        | 8  | <b>5</b> |



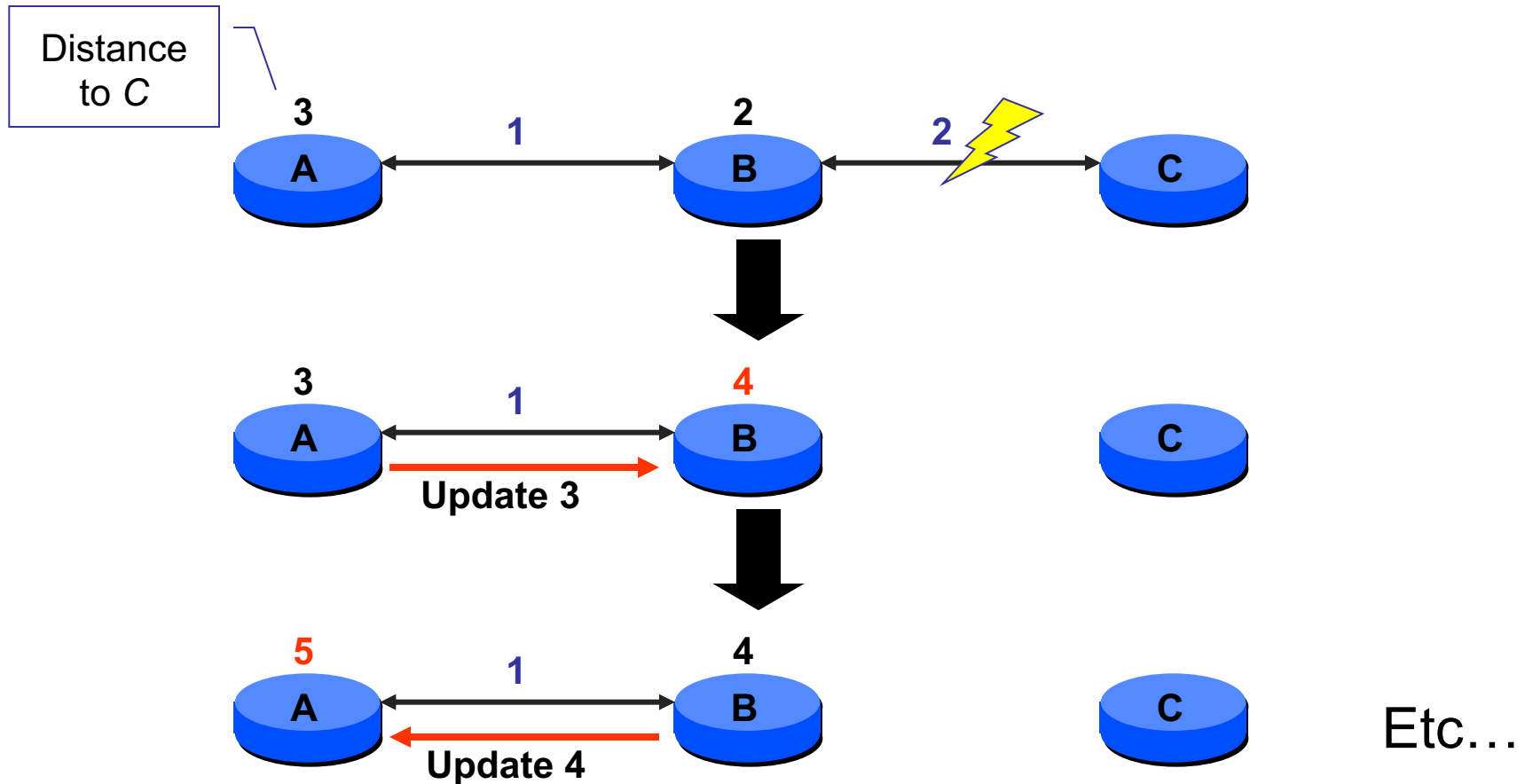
# Handling Link Failure

- A marks distance to E as  $\infty$ , and tells B
- E marks distance to A as  $\infty$ , and tells B and D
- B and D recompute routes and tell C, E and E
- etc... until converge



| Info at node | Distance to Node |   |   |    |    |
|--------------|------------------|---|---|----|----|
|              | A                | B | C | D  | E  |
| A            | 0                | 7 | 8 | 10 | 12 |
| B            | 7                | 0 | 1 | 3  | 5  |
| C            | 8                | 1 | 0 | 2  | 4  |
| D            | 10               | 3 | 2 | 0  | 2  |
| E            | 12               | 5 | 4 | 2  | 0  |

# Problem: Counting to Infinity





# Why so High?

---

- Updates don't contain enough information (no route)
- Can't totally order “bad news” (a link has gone down) above “good news” (a link is available)
- $B$  accepts  $A$ 's path to  $C$  that is *implicitly* through  $B$ !
- Aside: this also causes delays in convergence even when it doesn't count to infinity



# Mitigation Strategies

---

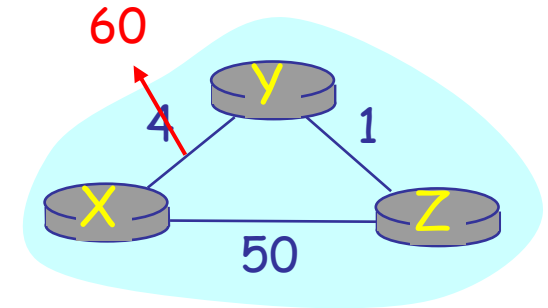
- Hold downs
  - ◆ As metric increases, delay propagating information
  - ◆ Limitation: Delays convergence
- Loop avoidance
  - ◆ Full path information in route advertisement
  - ◆ Explicit queries for loops
- Split horizon
  - ◆ Never advertise a destination **through its next hop**
    - » A doesn't advertise C to B
  - ◆ **Poison reverse**: Send **negative information** when advertising a destination through its next hop
    - » A advertises C to B with a metric of  $\infty$
    - » Limitation: Only works for “loop”s of size 2

# Poison Reverse Example



If Z routes through Y to get to X:

☠️<sub>Z:X</sub> Z tells Y its (Z's) distance to X is infinite  
(so Y won't route to X via Z)



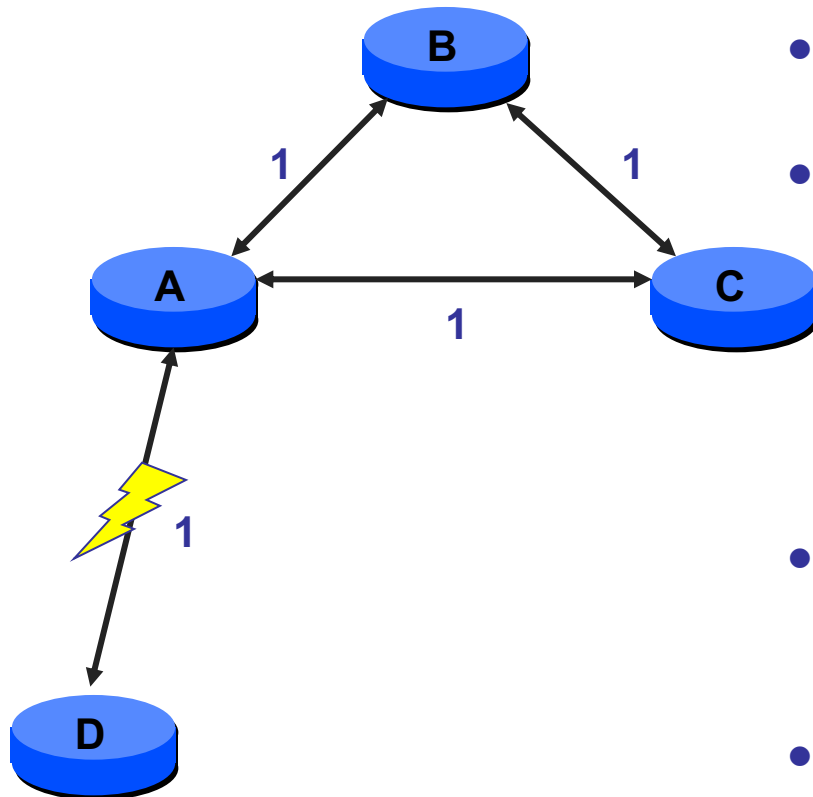
| Y              |   | via |  |                   |
|----------------|---|-----|--|-------------------|
| D <sup>Y</sup> | X | Z   |  |                   |
| x              | 4 | ∞   |  | ☠️ <sub>Z:X</sub> |

| Z              |    | via |  |  |
|----------------|----|-----|--|--|
| D <sup>Z</sup> | X  | Y   |  |  |
| x              | 50 | 5   |  |  |





# Split Horizon Limitations



- A tells B & C that D is unreachable
- B computes new route through C
  - ◆ Tells C that D is unreachable (poison reverse)
  - ◆ Tells A it has path of cost 3 (split horizon doesn't apply)
- A computes new route through B
  - ◆ A tells C that D is now reachable
- Etc...



# In practice

---

- RIP: Routing Information Protocol
  - ◆ DV protocol with hop count as metric
    - » Infinity value is 16 hops; limits network size
    - » Includes split horizon with poison reverse
  - ◆ Routers send vectors every 30 seconds
    - » With triggered updates for link failures
    - » Time-out in 180 seconds to detect failures
  - ◆ Rarely used today
- EIGRP: proprietary Cisco protocol
  - ◆ Ensures loop-freedom (DUAL algorithm)
  - ◆ Only communicates changes (no regular broadcast)
  - ◆ Combine multiple metrics into a single metric (BW, delay, reliability, load)



# DV Summary

---

- Distance Vector shortest-path routing
  - ◆ Each node sends list of its shortest distance to each destination to its neighbors
  - ◆ Neighbors update their lists; iterate
- Weak at adapting to changes out of the box
  - ◆ Problems include loops and count to infinity



# For next time...

---

- HW3 out tonight
- Project 3 out Friday
- Read Chapter 4.1 in P&D (internetwork routing)