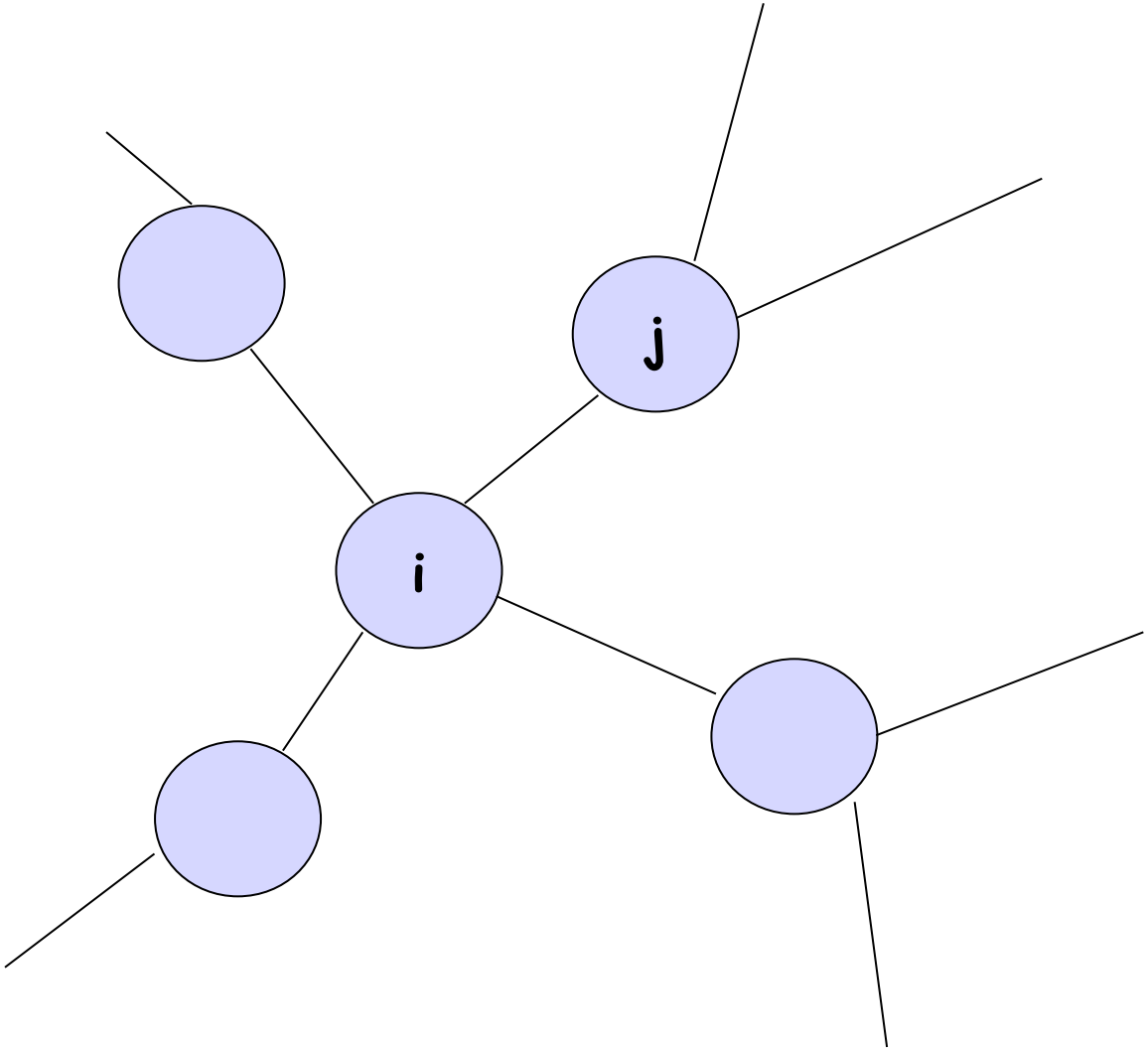
Network Routing:
Distance Vector

ABF Convergence

- There are too many different "runs" of ABF, so need to use monotonicity

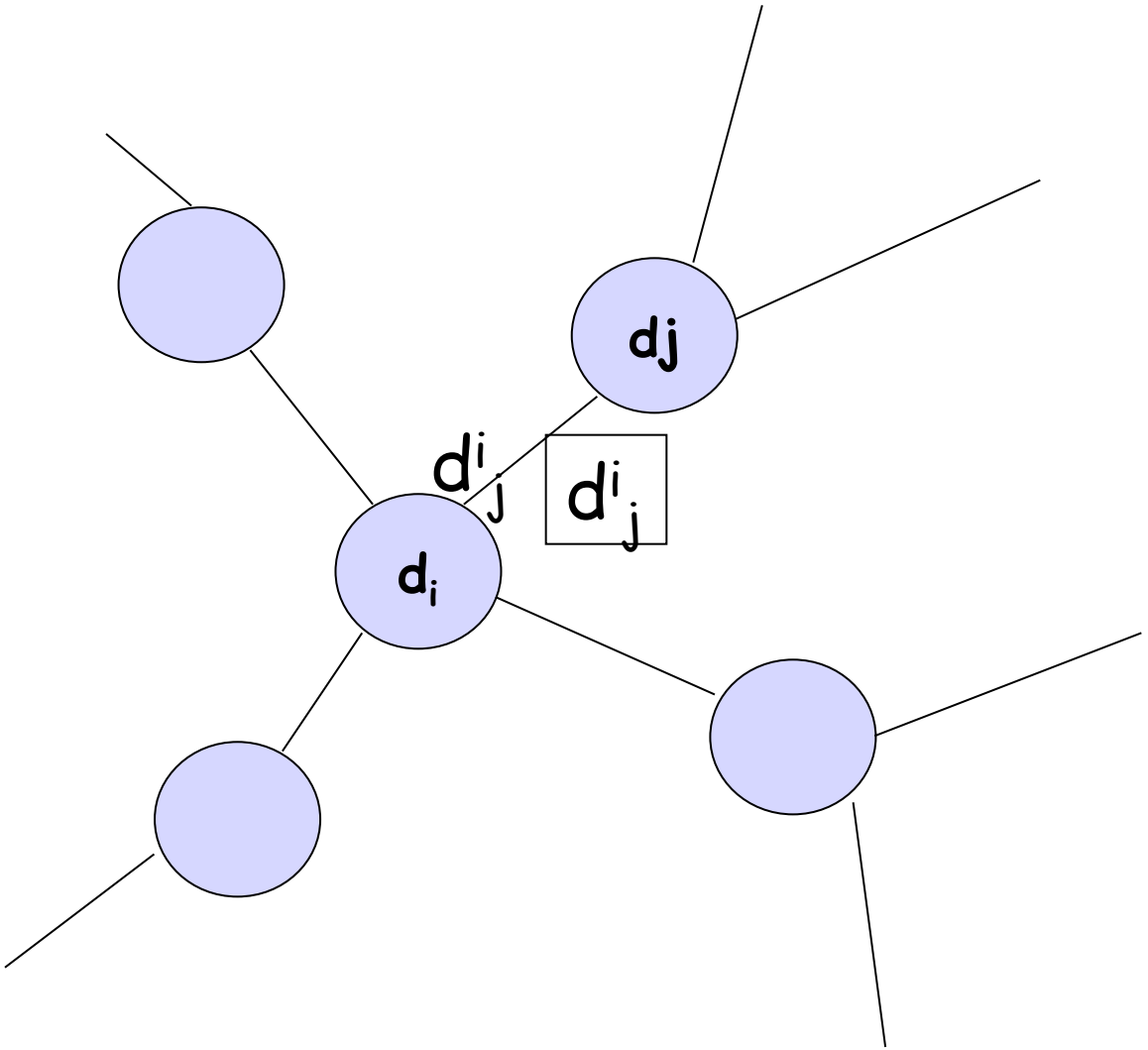
- Consider two sequences:
  - SBF/∞; call the sequence U()
  - SBF/-1; call the sequence L()

System State



where can distance estimate from node j appear?

System State



Three types of distance estimates from node j:

- d_j: current distance estimate at node j
- d^i_j: last d_j that neighbor i received
- d^i_j: those d_j that are still in transit to neighbor i

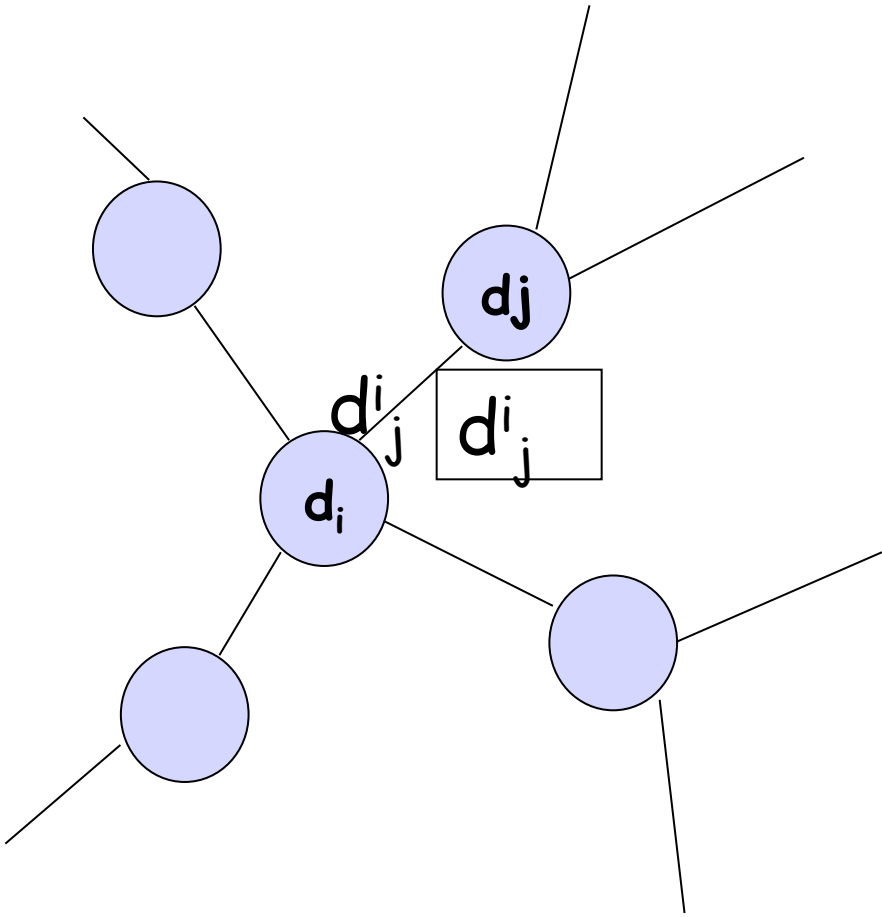
ABF Convergence

- Consider the time when the topology is stabilized as time 0

- U(0) and L(0) provide upper and lower bound at time 0 on all corresponding elements of states
  - $L_j(0) \le d_j \le U_j(0)$ for all d_j state at node j
  - $L_j(0) \le d^i_j \le U_j(0)$
  - $L_j(0) \le$ `update messages` $d^i_j \le U_j(0)$

ABF Convergence

- d_j
  - after at least one update at node j: d_j falls between $L_j(1) \leq d_j \leq U_j(1)$
- d^i_j :
  - eventually all d^i_j that are only bounded by $L_j(0)$ and $U_j(0)$ are replaced with in $L_j(1)$ and $U_j(1)$

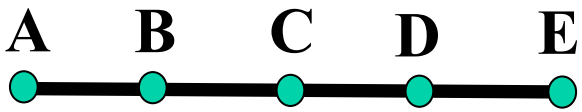


Asynchronous Bellman-Ford: Summary

- *Distributed*: each node communicates its routing table to its directly-attached neighbors
- *Iterative*: continues periodically or when link changes, e.g. detects a link failure
- *Asynchronous*: nodes need *not* exchange info/ iterate in lock step!
- *Convergence* in finite steps, independent of initial condition if network is connected

Properties of Distance-Vector Algorithms

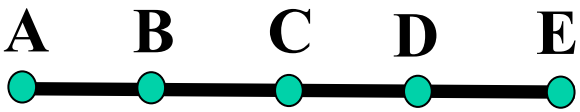
- Good news propagate fast



∞	∞	∞	∞	initially
1	∞	∞	∞	After 1 exchange
1	2	∞	∞	After 2 exchanges
1	2	3	∞	After 3 exchanges
1	2	3	4	After 4 exchanges

Properties of Distance-Vector Algorithms

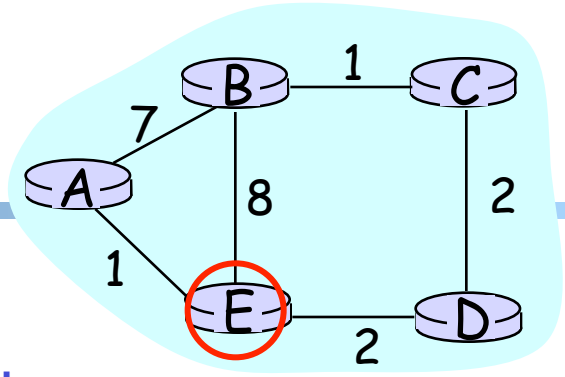
- Bad news propagate slowly



1	2	3	4	initially
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges
…				
∞	∞	∞	∞	

- This is called the *counting-to-infinity* problem
  - Question: why does counting-to-infinity happen?

The Reverse-Poison (Split-horizon) Hack



If the path to dest. is through neighbor h, report ∞ to neighbor h for dest.

distance tables from neighbors

computation

E's distance table

distance table E sends to its neighbors

$Dist^E()$	A	B	D	A	B	D	E's distance table
A	0	7	∞	1	15	∞	1, A
B	7	0	∞	8	8	∞	8, B
C	∞	1	2	∞	9	4	4, D
D	∞	∞	0	∞	∞	2	2, D
	1	8	2				
	c(E,A)	c(E,B)	c(E,D)				

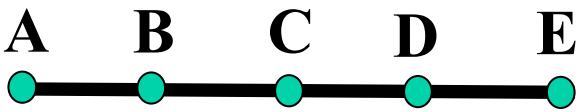
destinations

distance

through neighbor

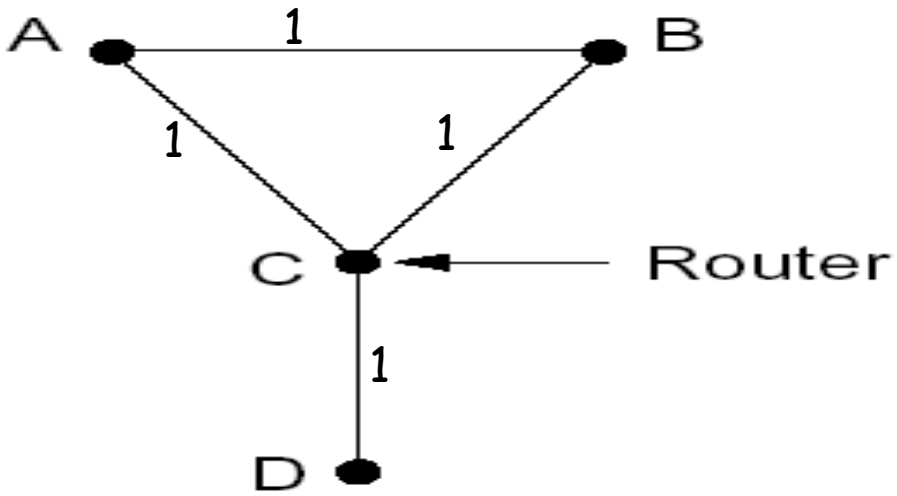
To A	To B	To D
A: ∞	A: 1	A: 1
B: 8	B: ∞	B: 8
C: 4	C: 4	C: ∞
D: 2	D: 2	D: ∞
E: 0	E: 0	E: 0

Reverse Poison Example



destinations	Distance table of node B: $Dist^B()$ B	Message from node C	B's new distance table
A	1, A	∞	∞
C	1, C	0	1, C
D	2, C	1	2, C
E	3, C	2	3, C

An Example Where Split-horizon Fails

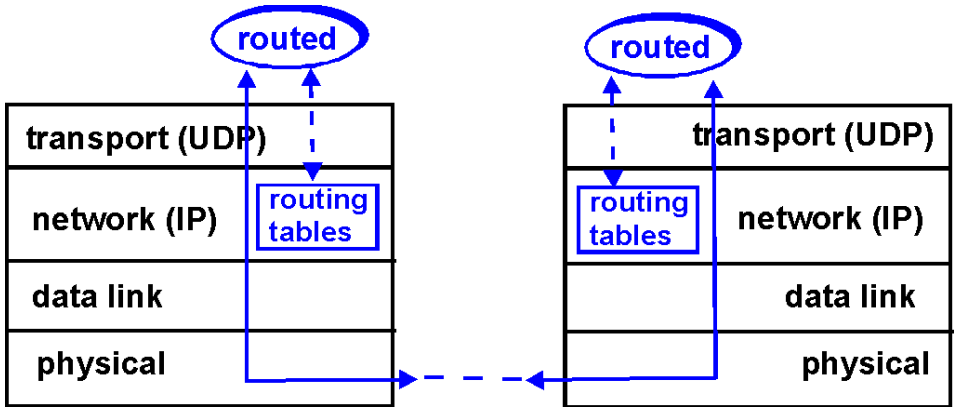


- When the link between C and D fails, C will set its distance to D as ∞
- However, unfortunate timing can cause problem
  - A receives the bad news (∞) from C, A will use B to go to D
  - A sends the news to C
  - C sends the news to B

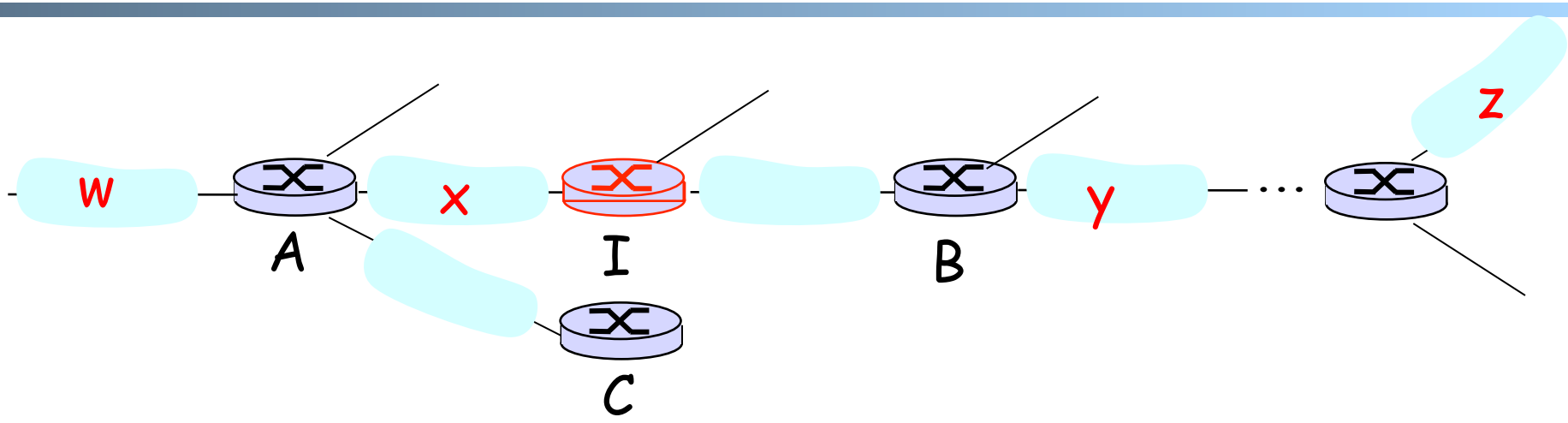
Question: what is the routing loop formed?

Example: RIP (Routing Information Protocol)

- Distance vector
- Included in BSD-UNIX Distribution in 1982
- Link cost: 1
- Distance metric: # of hops (max = 15 hops)
  - *why?*
- Distance vectors
  - Exchanged every 30 sec via Response Message (also called **advertisement**) using UDP
  - Each advertisement: route to up to 25 destination nets



RIP (Routing Information Protocol)



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing table in I

RIP: Link Failure and Recovery

- If no advertisement heard after 180 sec --> neighbor/link declared dead
  - Routes via neighbor invalidated
  - New advertisements sent to neighbors
  - Neighbors in turn send out new advertisements (if tables changed)
  - Link failure info quickly propagates to entire net
  - Reverse-poison used to prevent ping-pong loops (infinite distance = 16 hops)

IGRP and EIGRP: Cisco Proprietary Implementation

- Link metric:

$$\text{EIGRP}_{\text{metric}} = \{k1 \times \text{BW} + [(k2 \times \text{BW})/(256 - \text{load})] + k3 \times \text{delay}\} \times \{k5/(\text{reliability} + k4)\}$$

By default, k1=k3=1 and k2=k4=k5=0. The default composite metric for EIGRP, adjusted for scaling factors, is as follows:

$$\text{EIGRP}_{\text{metric}} = 256 \times \{ [10^7/\text{BW}_{\text{min}}] + [\text{sum\_of\_delays}] \}$$

IGRP and EIGRP: Cisco Proprietary Implementation

- BW_{min} is in kbps and the sum of delays are in 10s of microseconds.
- **Example**
  - The bandwidth and delay for an Ethernet interface are 10 Mbps and 1ms, respectively.
  - The calculated EIGRP BW metric is as follows:

$$256 \times 10^7/BW = 256 \times 10^7/10{,}000$$
$$= 256 \times 10000$$
$$= 256000$$

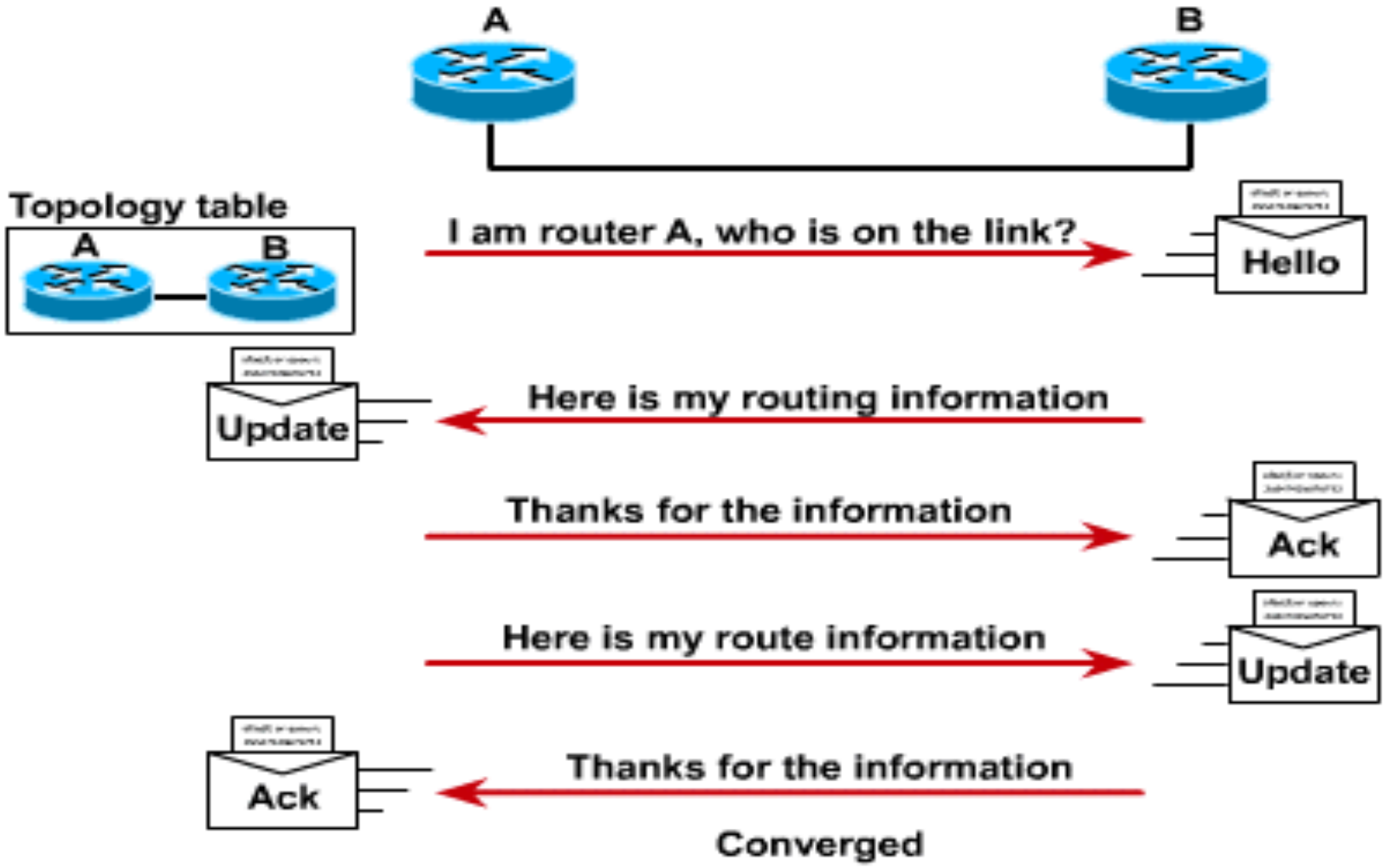
EIGRP Neighbor Discovery

- EIGRP routers actively establish relationships with their neighbors
- EIGRP routers establish adjacencies with neighbor routers by using small **hello** packets.
- The **Hello protocol** uses a multicast address of **224.0.0.10**, and all routers periodically send hellos.

EIGRP Neighbor Discovery

- On hearing hellos, the router creates a table of its neighbors.
- The continued receipt of these packets maintains the neighbor table
- By forming adjacencies, EIGRP routers do the following:
  - Dynamically learn of new routes that join their network
  - Identify routers that become either unreachable or inoperable
  - Rediscover routers that had previously been unreachable

Neighbor Discovery - 3



Default Hello Intervals and Hold Time for EIGRP

Bandwidth	Example Link	Default Hello Interval	Default Hold Time
1.544 Mbps or less	Multipoint Frame Relay	60 seconds	180 seconds
Greater than 1.544 Mbps	T1, Ethernet	5 seconds	15 seconds

Outline

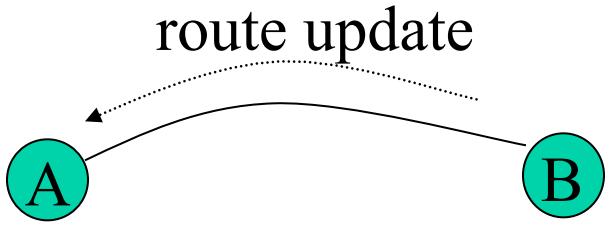
- Recap
- *Distance vector protocols*
  - synchronous Bellman-Ford (SBF)
  - asynchronous Bellman-Ford (ABF)
  - *destination-sequenced distance vector (DSDV)*

Destination-Sequenced Distance Vector protocol (DSDV)

- An extension of distance vector protocol to address the counting-to-infinity problem

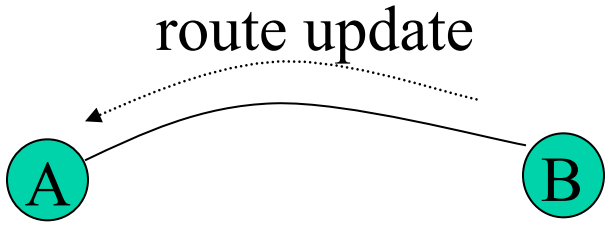
- Extension
  - DSDV tags each route with a sequence number
  - Each destination node D periodically advertises monotonically increasing even-numbered sequence numbers
  - When a node realizes that its link to a destination is broken, it advertises the route to D with an infinite metric and a sequence number which is one greater than the previous route (i.e. an odd seq. number)
    - The route is repaired by a later even-number advertisement from the destination

DSDV: More Detail



- Let's assume the destination node is D
- There are optimizations but we present a simple version:
  - Each node maintains only (S^B, d^B), where S^B is the sequence number at B for destination D and d^B is the best distance using a neighbor from B to D
- Both periodical and triggered updates
  - Periodically: D increases its seq. by 2 and broadcasts with (S^D, 0)
  - If B is using C as next hop to D and B discovers that C is no longer reachable
    - B increases its sequence number S^B by 1 and sends (S^B, ∞) to all neighbors ($d^B = \infty$)

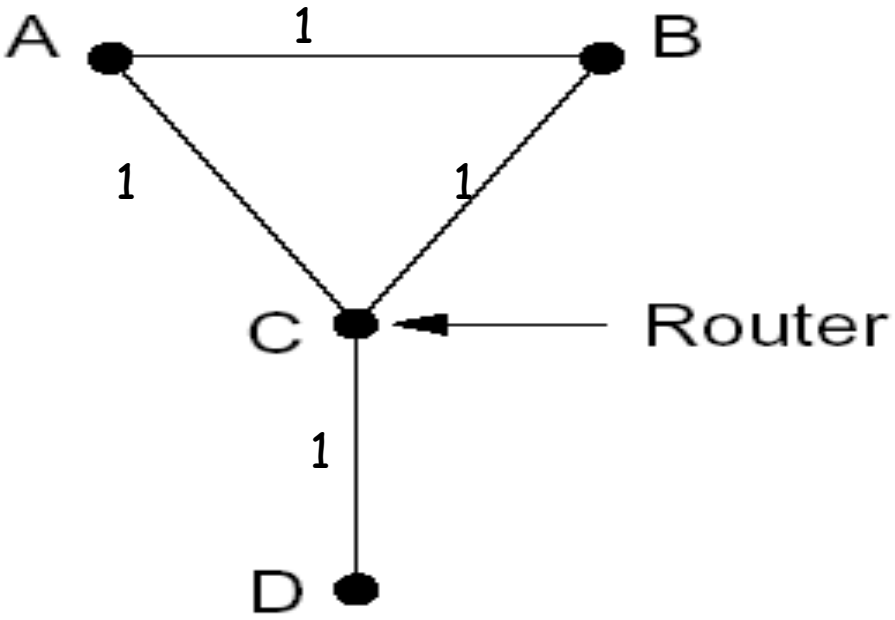
DSDV: Update



- Update after receiving a message
  - assume B sends to A its current state (S^B, d^B)
  - when A receives (S^B, d^B)
    - if $S^B > S^A$, then
      // always update if a higher seq#
      - » $S^A = S^B$
      - » if ($d^B == \infty$) $d^A = \infty$; else $d^A = d^B + d(A,B)$
    - else if $S^A == S^B$, then
      - » if $d^A > d^B + d(A,B)$
        // update for the same seq# only if better route
        $d^A = d^B + d(A,B)$ and uses B as next hop

Example

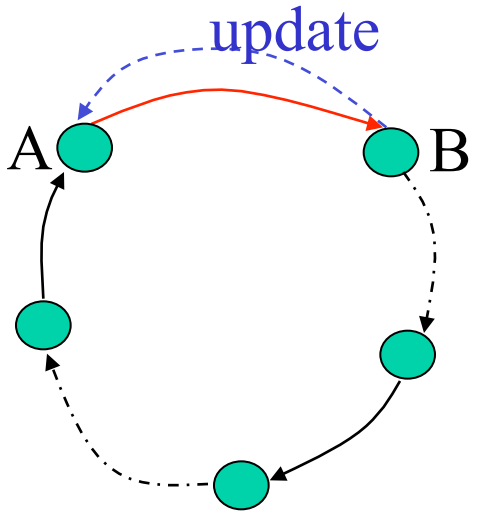
- When C discovers that C-D link is down, it increases its seq# and broadcasts its cost to be infinite



Claim: DSDV Does Not Form Loop

- Question: what is a loop?
  - A loop is a global state (consisting of the nodes' local states) at a global moment (observed by an oracle) such that there exist nodes A, B, C, … E such that A (locally) thinks B as down stream, B thinks C as down stream, … E thinks A as down stream

- Initially no loop (no one has next hop so no loop)
- Derive contradiction after a node processes an update,
  - E.g., when A receives the update from B, A decides to use B as next hop and forms a loop



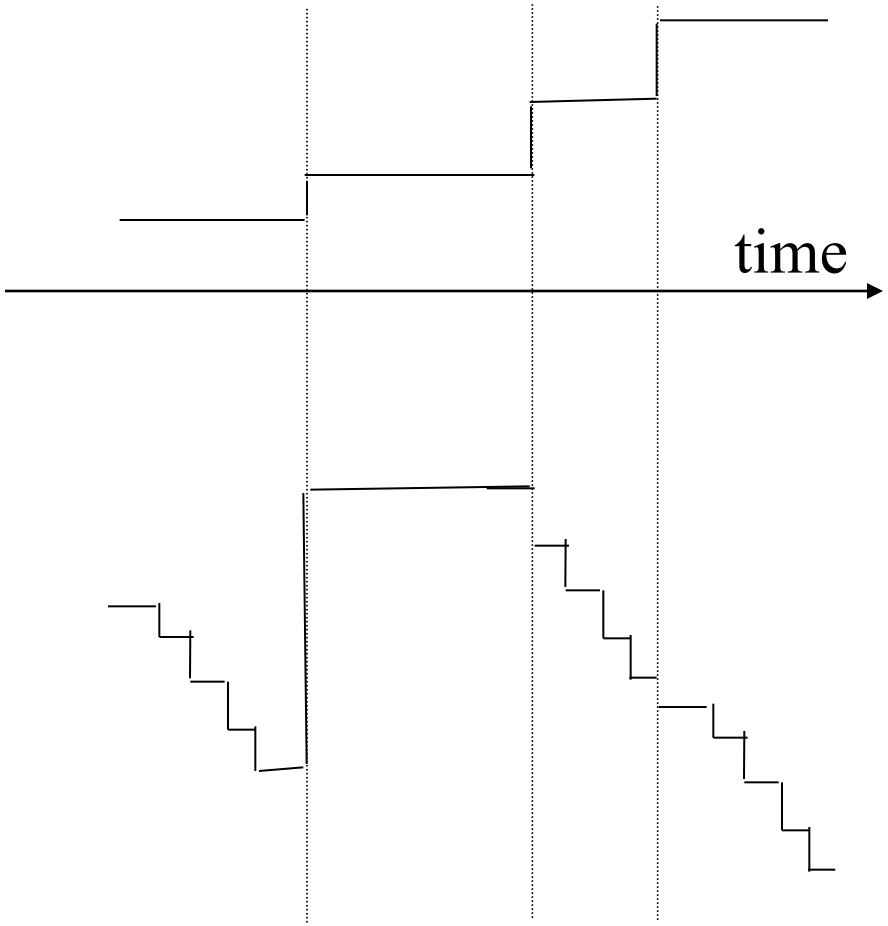
Background: Global Invariants

- This is a very effective method in understanding distributed asynchronous protocols
- Invariants are defined over the states of the distributed nodes

- Consider any node B.
- Discussion: what are some invariants over the state of node B, i.e., (S^B, d^B)?

Invariants of a Single Node B

- Some invariants about the state of a node
  - S^B is non-decreasing
  - d^B is non-increasing for the same sequence number



Invariants of if A Considers B as Next Hop

- Some invariants if A considers B as next hop

  - S^A cannot be an odd number, d^A is not ∞

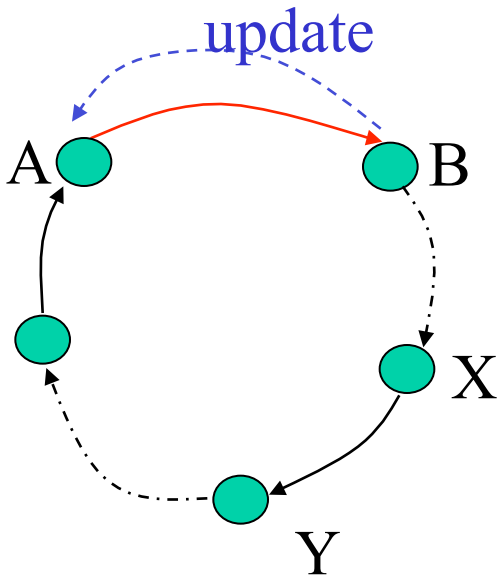
  - $S^B \geq S^A$

    because A is having the seq# which B last sent to A; B's seq# might be increased after B sent its state

    - if $S^B == S^A$ then $d^B < d^A$
      because d^A is based on d^B which B sent to A some time ago, $d^B < d^A$ since all link costs are positive; d^B might be decreased after B sent its state

Loop Freedom of DSDV

- Consider a critical moment
  - A starts to consider B as next hop, and we have a loop
- If any link in the loop (X considers Y as next hop) satisfies $S^Y > S^X$
  - by transition along the loop $S^B > S^B$



- If all nodes along the loop have the same sequence number
  - by transition along the loop $d^B > d^B$

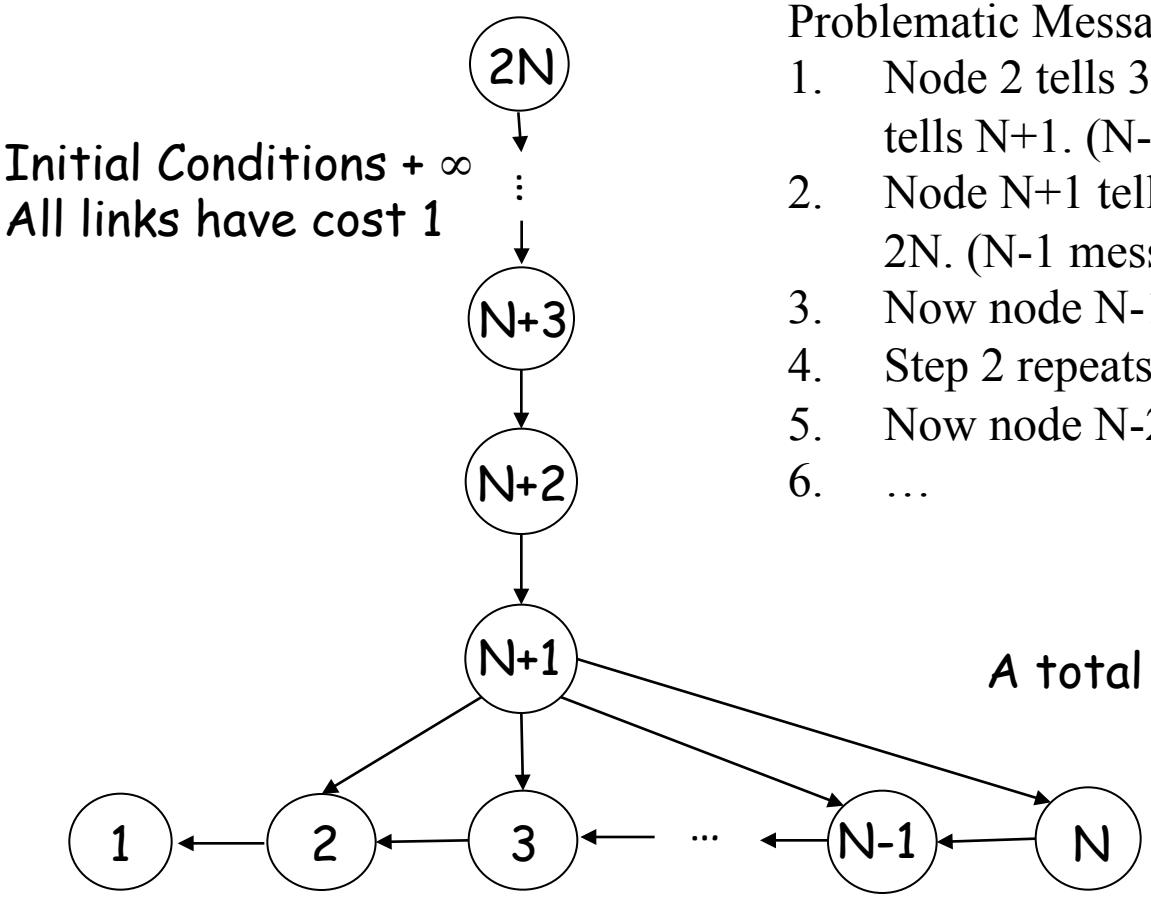
Summary: DSDV

- DSDV uses sequence number to avoid routing loops
  - Seq# partitions routing updates from different outside events
  - Within same event, no loop so long each node only decreases its distance
- EIGRP: a routing protocol in Cisco uses a Diffusive Update Algorithm (DUAL)
  - Locally adjust if known not causing routing loops
  - Diffusive computing to resolve non-local cases

Discussion: Distance Vector Routing

- What do you like about distance vector routing?
  - Simplicity
- What do you not like about distance vector routing?
  - Churns before convergence
    - Path exploration; count to infinite
  - Not enough information when making routing decision (only next-hops and their distance estimates)

Churns of DV: One Example



Problematic Message sequences

1. Node 2 tells 3. Node 3 tells 4…Node N tells N+1. (N-1 messages)
2. Node N+1 tells N+2, N+2 tells N+3,…, 2N. (N-1 messages)
3. Now node N-1 tells node N+1
4. Step 2 repeats
5. Now node N-2 tells node N+1
6. …

A total of $N^2 - 2$ messages

Question to think about: is this the worst case?