Distance Vector Routing Algorithm

RIP

Kameswari Chebrolu

Recap

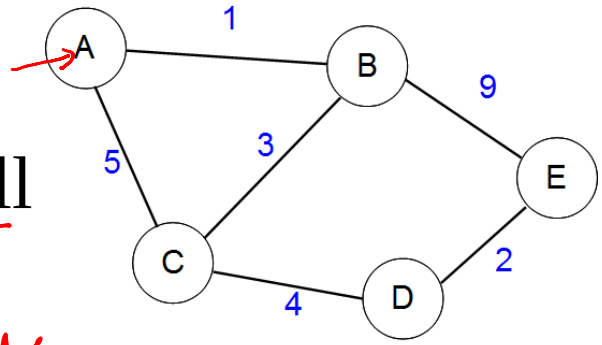
- Network Layer: Routing process
- Routing: Find the least cost path between two node
- Many approaches. Our focus: Dynamic, distributed algorithms
- Distance Vector Algorithm

Background

- Also goes by the name Bellman-Ford algorithm
- Used in ARPAnet
- Later in Internet under the routing protocol standard RIP (Routing Information Protocol)
- Now, it is not used much

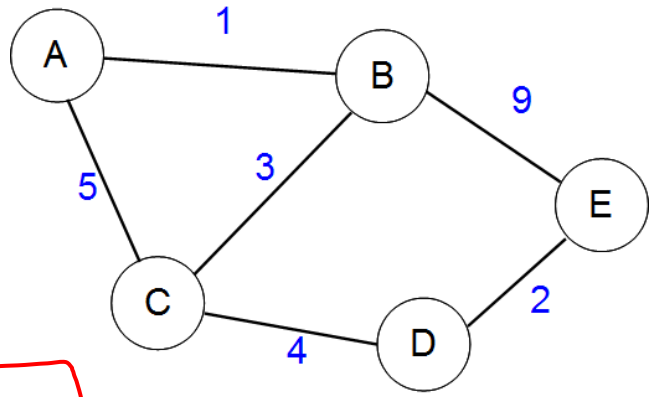
Protocol Framework

- Initial state at a node: distance (cost) to neighbors is known
- Final state at a node: distance (cost) to all nodes is known, and also the next-hop
- Need to handle
  - What information to exchange? (message format)
  - How to act on a message?
  - When to send a message?



State Maintained

- Each node maintains a routing table (distance vector)
  - Destination
  - Estimated cost to destination
  - Next hop via which to reach destination
- Initial state: Cost to neighbors



Dest	Cost	Next Hop
A	1	A
C	3	C
E	9	E

Initial Routing table at B

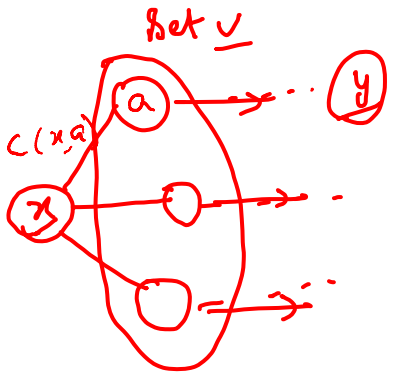
Dest	Cost	Next Hop
A	1	A
C	3	C
D	7	C
E	9	E

Final Routing table at B

Message Content

- Each node exchanges with all its neighbors "Routing Table" info
  - Destinations and 'Estimated' cost to destinations
  - Next hop information is not shared

Action at a router



- Bellman-Ford equation
  - $d_x(y) = \min_v\{c(x,v) + d_v(y)\}$
  - $d_x(y)$ – least cost path from node x to y
  - $\min_v$ – apply above eq. over all of x's neighbors

Action at a router

- On receiving a message from a neighbor v,
  - Update cost (estimate) to destinations based on above Bellman-ford equation; change next hop accordingly
  - For each y (destination in routing table of the received message)
    - $D_x(y) = \min\{\text{current estimate}, c(x,v) + D_v(y)\}$
  - Estimated costs finally converge to optimal cost after series of message exchanges

current estimate = $D_x(y)$

Reference Node C

Example

D	C	H
A	5	A
B	3	B
D	4	D

Routing Table of C

(1)

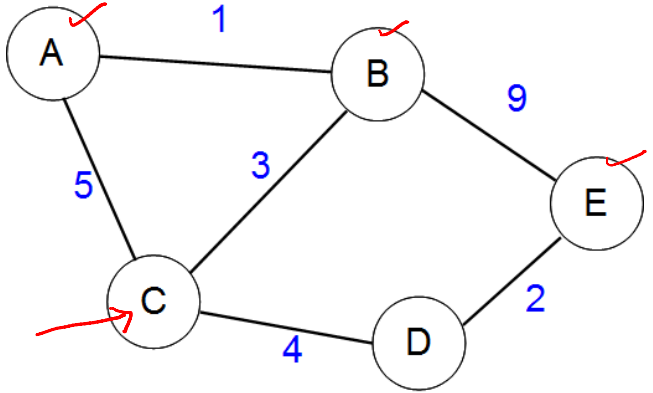
To	A
A	0
B	1
C	5

Message from A

C to A: C = 5

D	C	H
A	5	A
B	3	B
D	4	D

Routing Table of C



D	C	H
A	5	A
B	3	B
D	4	D

Routing Table of C

(2)

To	B
A	1
B	0
C	3
E	9

Message from B

C to B: C = 3

D	C	H
A	4	B
B	3	B
D	4	D
E	12	B

Routing Table of C

D	C	H
A	4	B
B	3	B
D	4	D
E	12	B

Routing Table of C

(3)

To	D
C	4
D	0
E	2

Message from D

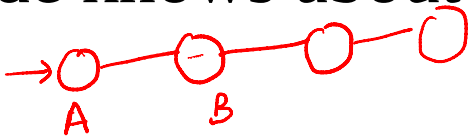
C to D: C = 4

D	C	H
A	4	B
B	3	B
D	4	D
E	6	D

Routing Table of C

Points to Note

- No topology change, convergence in a few rounds
  - After one message exchange, each node knows about nodes two hops away
  - After two message exchanges, each node knows about nodes three hops away
  - And so on...
- No node has global knowledge
- Fully distributed, yet maintains correct view



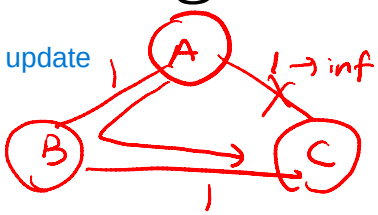
Updates

- When to send a routing message to neighbors?
- Triggered update: Sent whenever the DV changes
  - Link/Node failure or cost increase
- Periodic update: Sent even when no change in routing table
  - To tell others that “I am still alive”
  - To update others' DV in case some route becomes invalid
  - Order: few sec to few min

Periodic

C @ A ∞

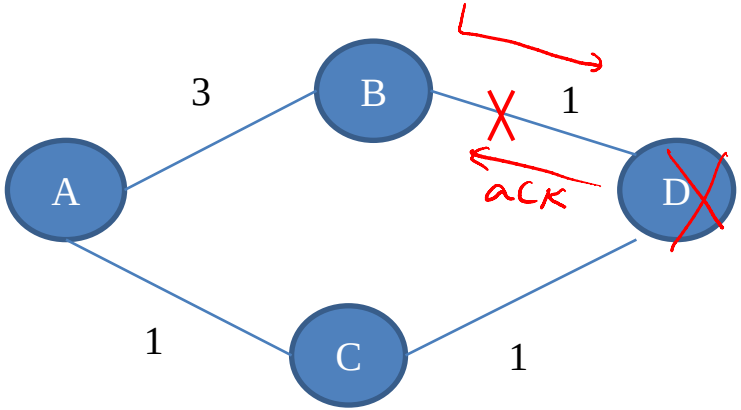
A deetcts link fail.. and makes a triggered update



B just ignores that trigered update, then how does A know that B is alive.. tahts why we need periodic updatres

Node/Link Failure

- How are node/link failures detected?
  - Didn't receive periodic update
  - Can also actively probe (probe-ack)



Summary

- Distance Vector: dynamic, distributed algorithm that works with local knowledge
- Based on Bellman-ford equation
- Handles node/link failures
- Ahead: Problems, solutions and standard related to distance vector algorithm