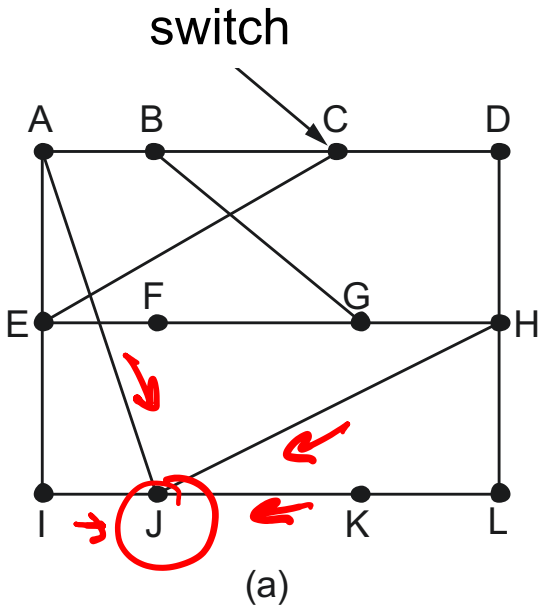
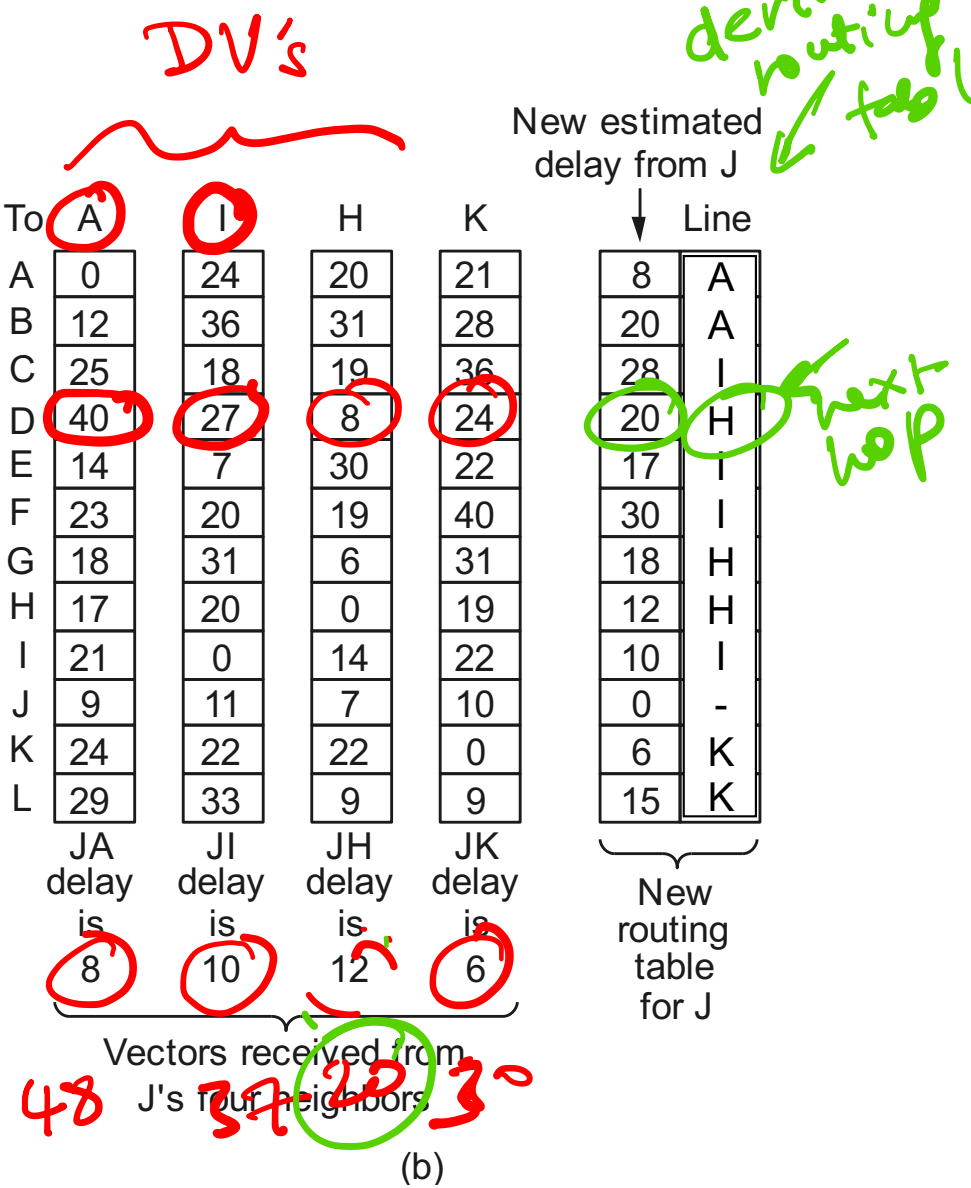
DV Routing



(a)

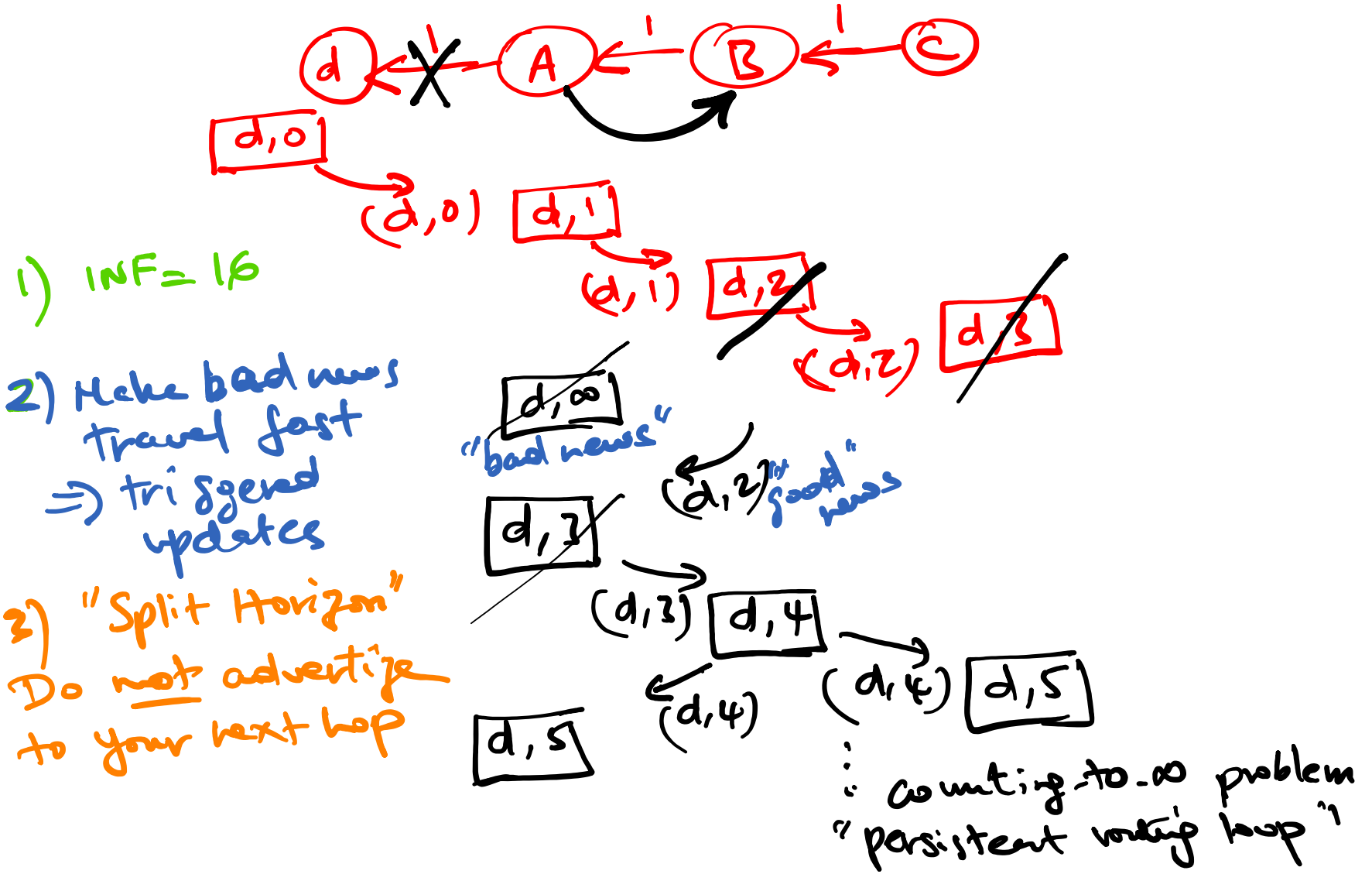


(b)

Distance Table
distance to each dest via each neighbor

dest D [48 37 29 30]

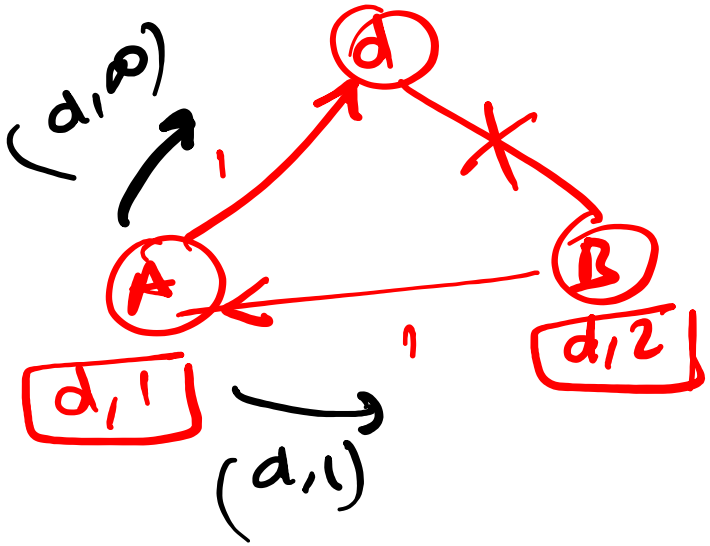
Routing Loops

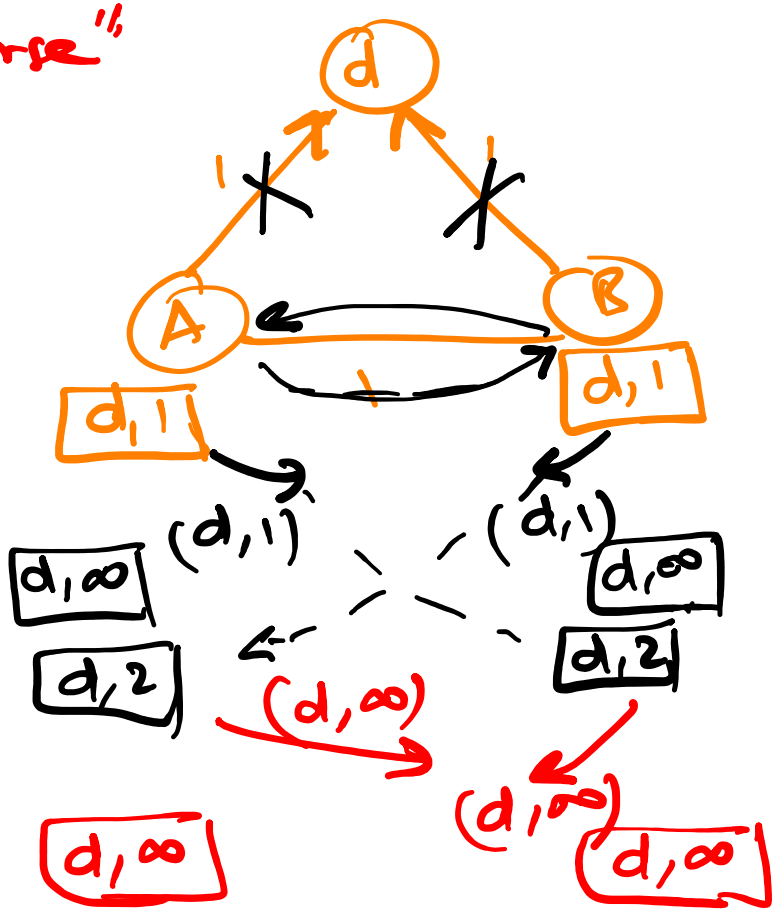


Routing Loops

"Split Horizon with Poison Reverse"

4) Advertize ∞ to your next hop "lie"

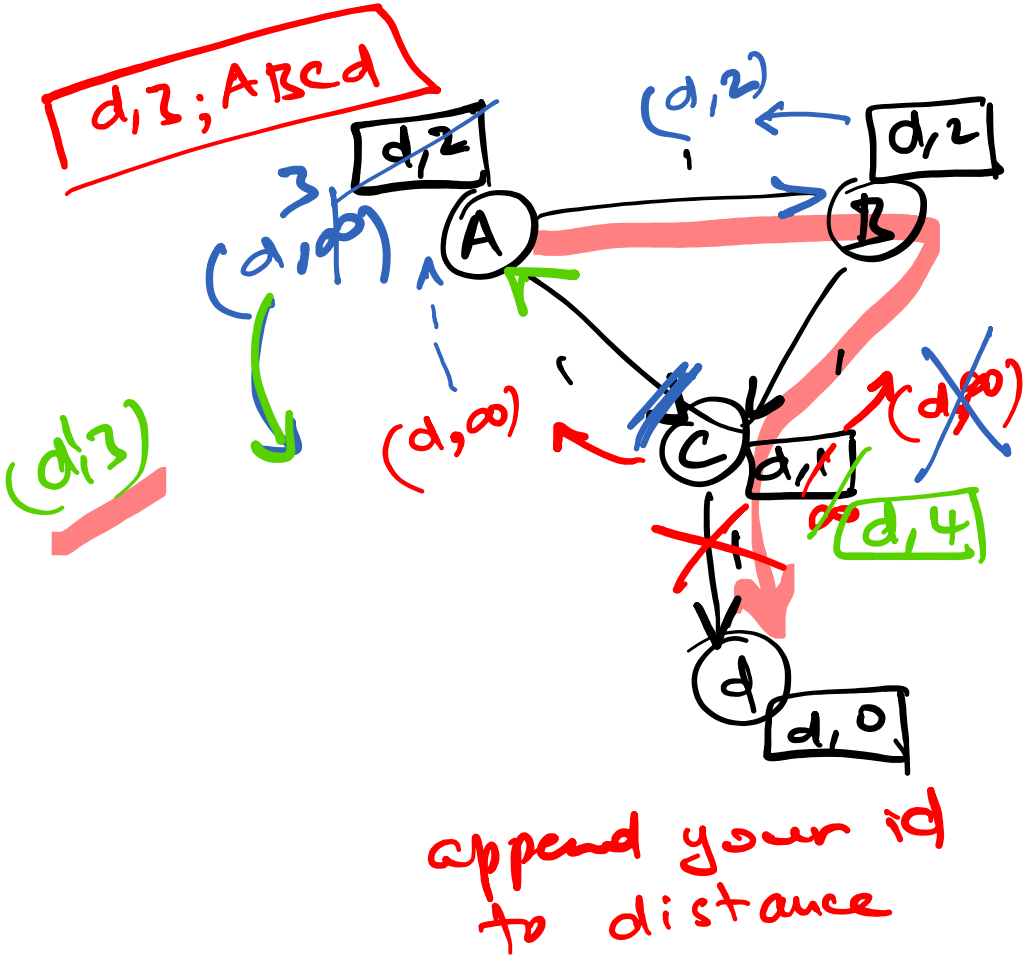




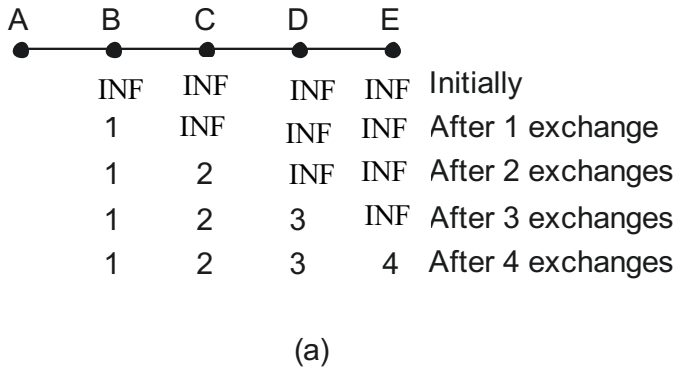
Routing Loops

PATH VECTOR

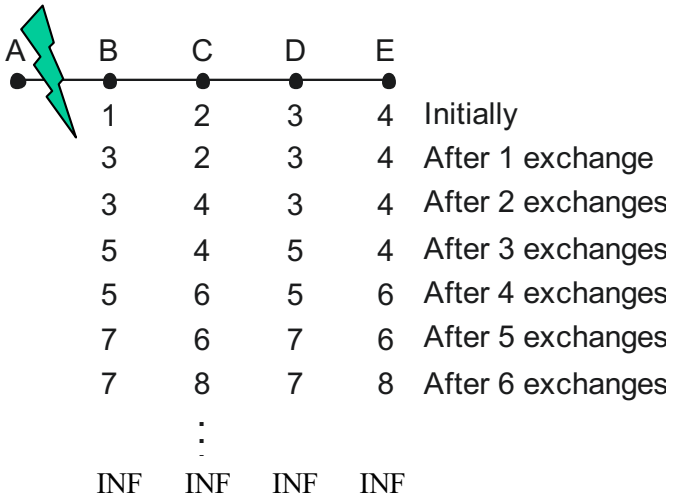
Do NOT advertise to a neighbor if they are on the path



Routing Loops



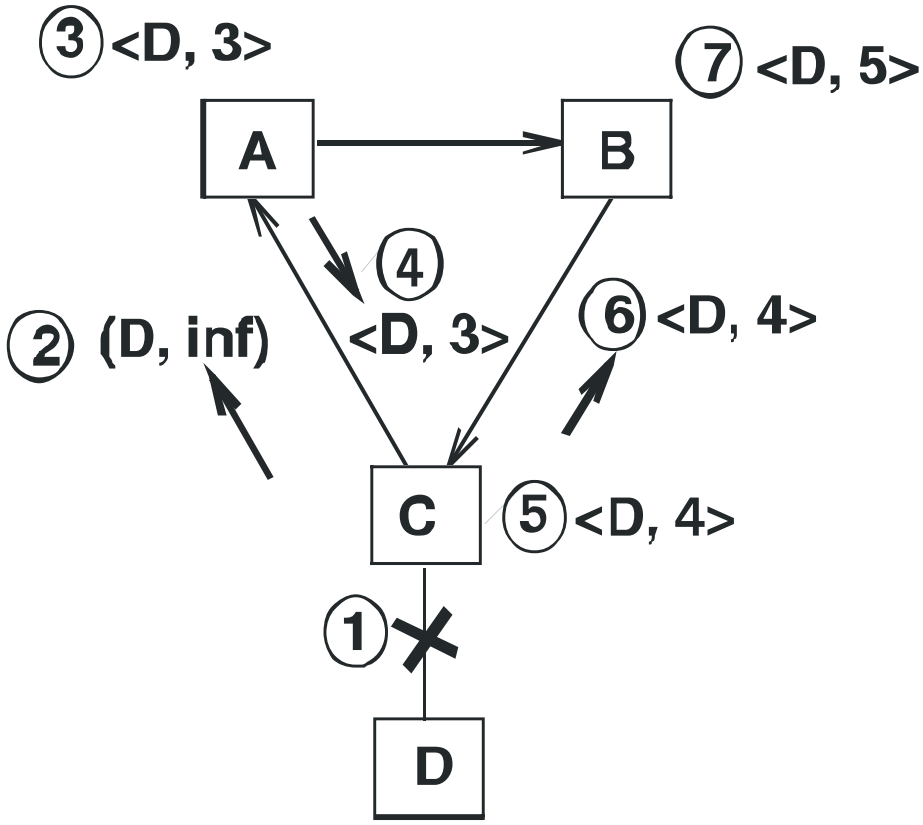
(a)



(b)

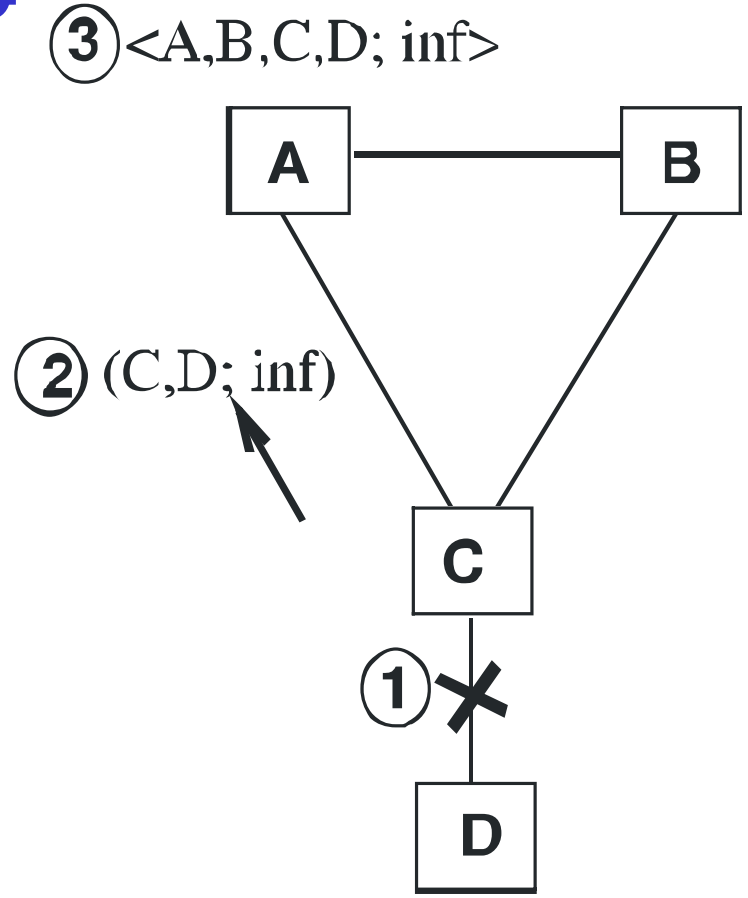
- Heuristics to break routing loops
  - set infinity to maximum distance/cost
  - split horizon: a node does not advertize to its next-hop
  - split horizon with poison reverse: a node advertizes to its next-hop a distance of infinity

Routing Loops can still happen!



- link CD goes down
- loop forms involving A, B and C

Path Vector Routing



- loops can be completely avoided

Distance-Vector versus Link-State

DV

memory

+ O(N * k) # neighbors

routing message size

- O(N)

path computation

distributed

- slow convergence
  routing loops/counting-to-∞

LS

- O(E) full view of network
  $O(N^2)$

+ O(k)

O(E) communication

centralized

+ converges fast

Distance-Vector versus Link-State

Distance-Vector:

- Easy to implement
- Larger routing update messages: message size is proportional to the number of nodes in the network
- Slow to converge: route computation is distributed
- Loops/count-to-infinity may happen
- If link changes don't affect shortest path, no message exchange

Distance-Vector versus Link-State

Link-State:

- Smaller routing update messages: message size depends on the number of neighbors a node has
- Converges quickly: route computation is centralized
- A node stores a complete view of the network
- Any link change requires a broadcast

Both have strengths and weaknesses.

One or the other is used in almost every network