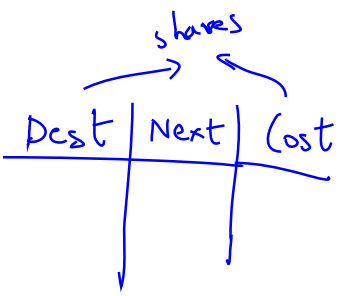
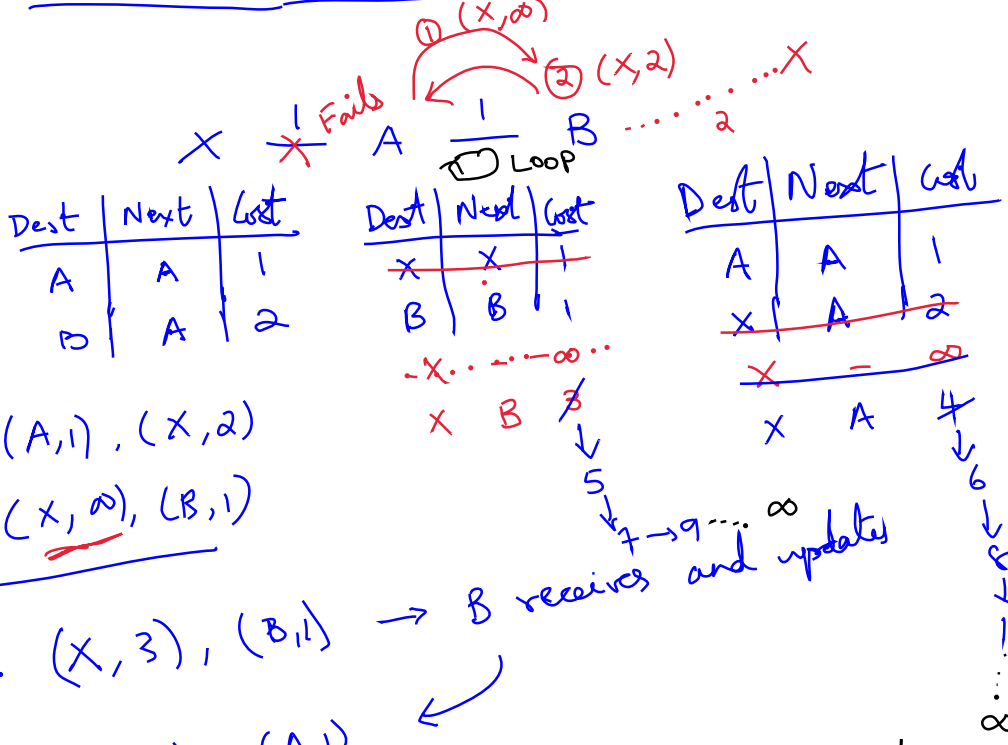
COUNT-TO-INFINITY PROBLEM





update

B tells A : (A,1) , (X,2)

A tells B : (X, ∞), (B,1)

update

A tells B : (X,3) , (B,1) → B receives and updates

B tells A : (X,4) ; (A,1)

A tells B : (X,5) , (B,1)

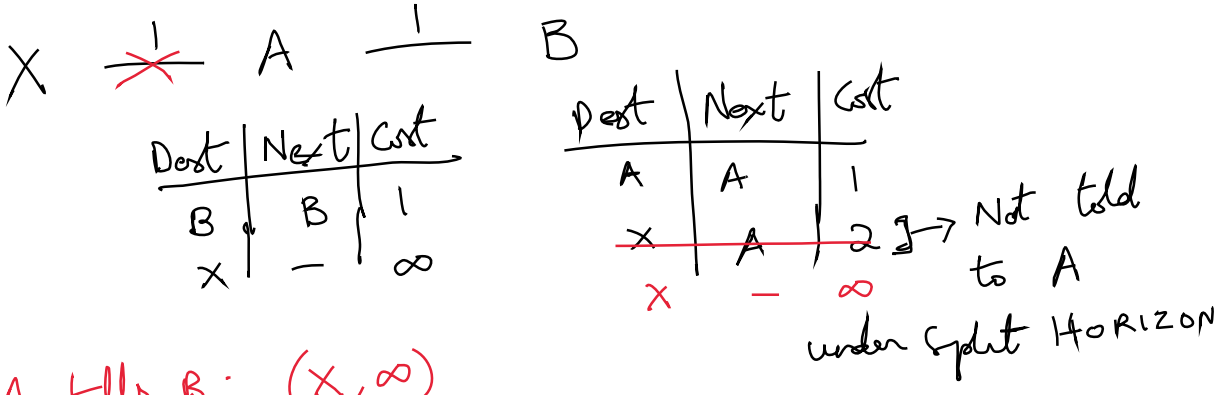
B tells A : (X,6) , (A,1) ∴

RIP: max. distance of 16

Cost = 16 ⇒ cannot reach destination

SPLIT-HORIZON

DO NOT ADVERTIZE INFORMATION ABOUT A DESTINATION TO A NEIGHBOUR IF THAT NEIGHBOUR IS THE NEXT HOP TO THE Destination



A tells B : (X, ∞)

B tells A : nothing about dist. to X.

Split Horizon with Poison Reverse:

A node tells its next hop to a destination that its distance to the destination is ∞

B sends advertisements (X, ∞) to A

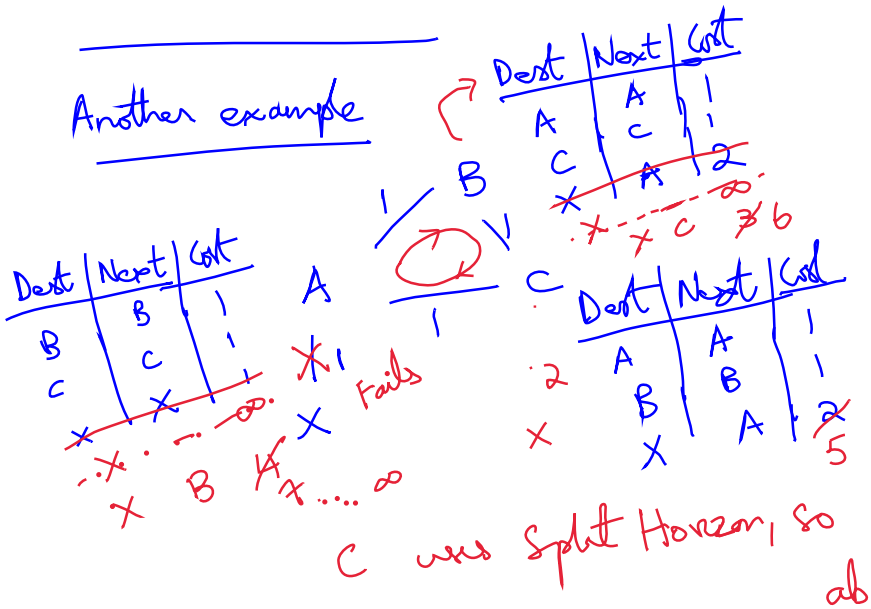
A sends (X, ∞) to B & C

Suppose the message to C gets lost

B sends (X, ∞) to C

C tells B, $(X, 2)$

Another example



C uses Split Horizon, so it does not give distance about X to A

B tells A: (X, 3)

A tells C: (X, 4)

RIP: Routing Information Protocol → D.V. based

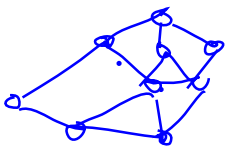
Cost of all links are 1, $(16 = \infty)$

max. cost allowed to a destination is 16

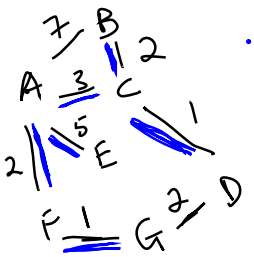
Advantages of Distance Vector: Simple & easy to implement

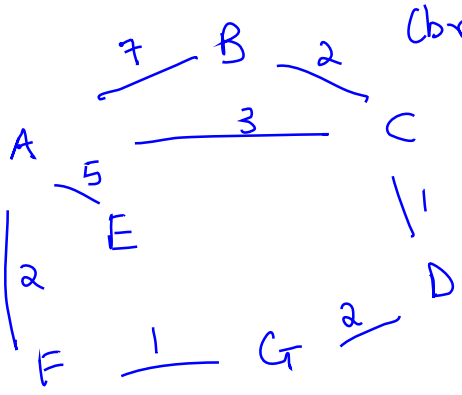
Disadvantages: - Count-to-infinity & routing loops

- convergence of routing tables



Link-State Routing





Each node sends to all others (broadcast) information about cost to immediate neighbours

D sends  to ALL

Dijkstra Algorithm Each node finds shortest path tree to all other nodes in network

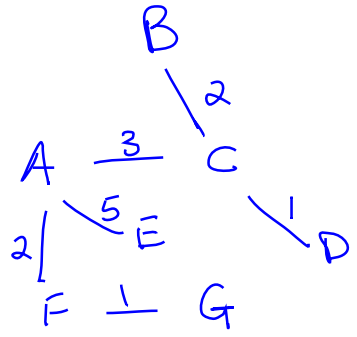
$T = \{A\}$ choose closest neighbour

$T = \{A, F\}$

$T = \{A, F, G\}$

$T = \{A, F, G, C\}$ $T = \{A, F, G, C, D, E, B\}$

Finally



A's Routing Table

Dest	Next	Cost
B	C	5
C	C	3
D	C	4
E	E	5
F	F	2
G	F	3

On Link Failure

- A and F broadcast to ALL, that their link has failed,
- ALL rerun Dijkstra's algs.

Advantages: No routing loops, count-to-infinity problems

Convergence of routing tables is fast

Disadv: Algo is more complex than D.V.

Q. What should be cost of a link