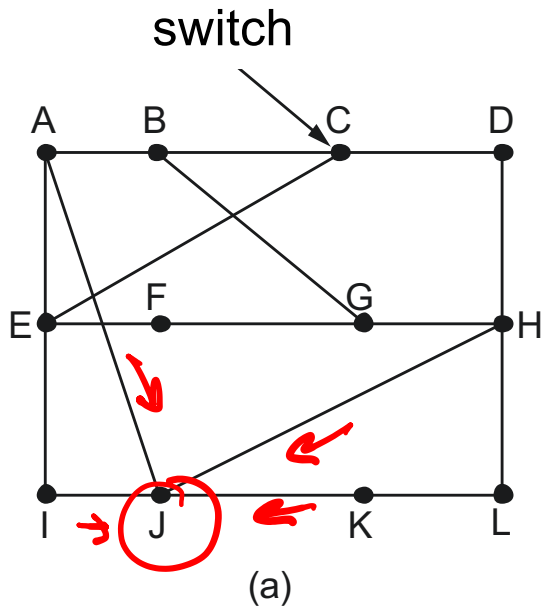


DV Routing



Distance Table
distance to each dest
via each neighbor

dest J $\begin{bmatrix} 48 & 37 & 20 & 10 \end{bmatrix}$

DV's

To	A	I	H	K
A	0	24	20	21
B	12	36	31	28
C	25	18	19	36
D	40	27	8	24
E	14	7	30	22
F	23	20	19	40
G	18	31	6	31
H	17	20	0	19
I	21	0	14	22
J	9	11	7	10
K	24	22	22	0
L	29	33	9	9

JA delay is 8 JI delay is 10 JH delay is 12 JK delay is 6

Vectors received from J's four neighbors: 48, 37, 20, 10

New estimated delay from J

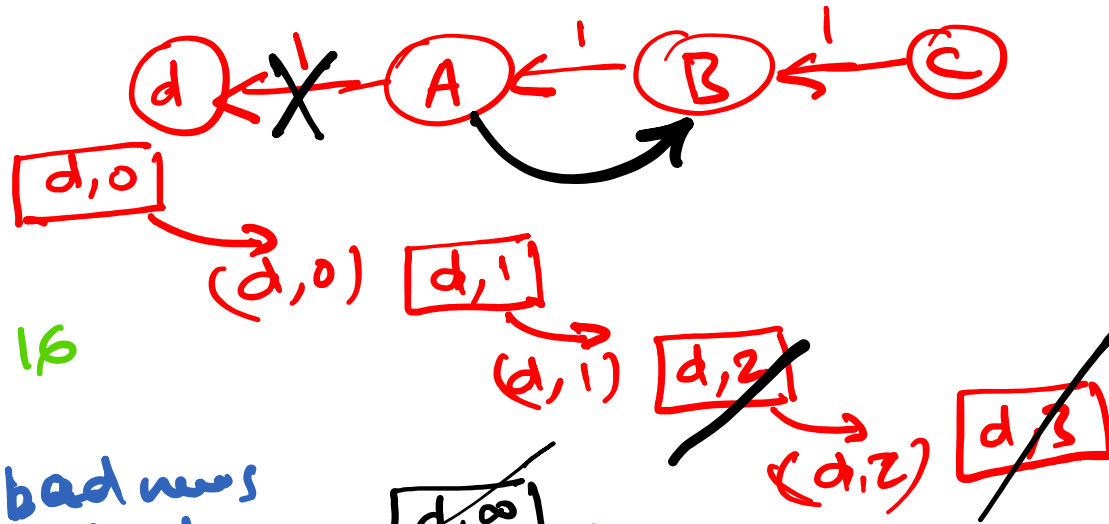
Line	
8	A
20	A
28	I
20	H
17	I
30	I
18	H
12	H
10	I
0	-
6	K
15	K

New routing table for J

derived routing table

next hop

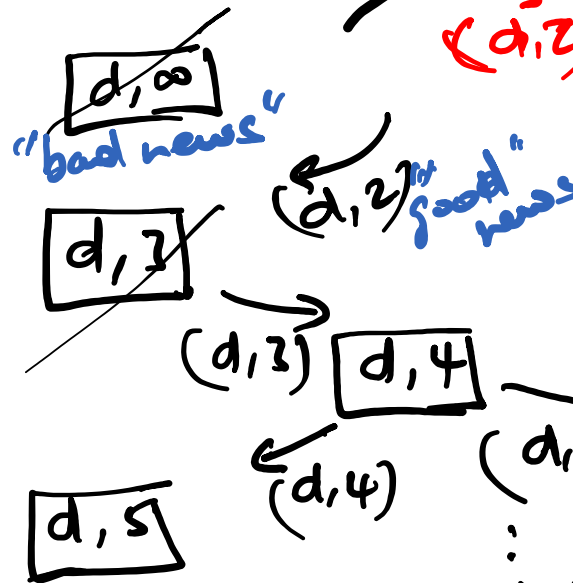
Routing Loops



1) $INF = 16$

2) Hehe bad news travel fast
 \Rightarrow triggered updates

3) "Split Horizon"
 Do not advertise to your next hop

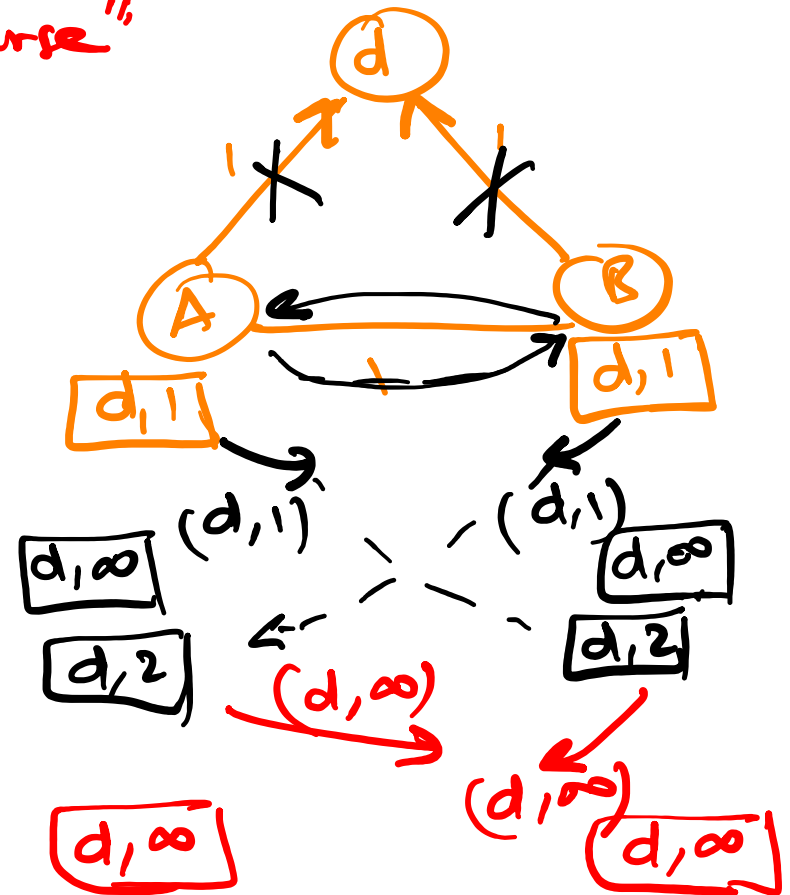
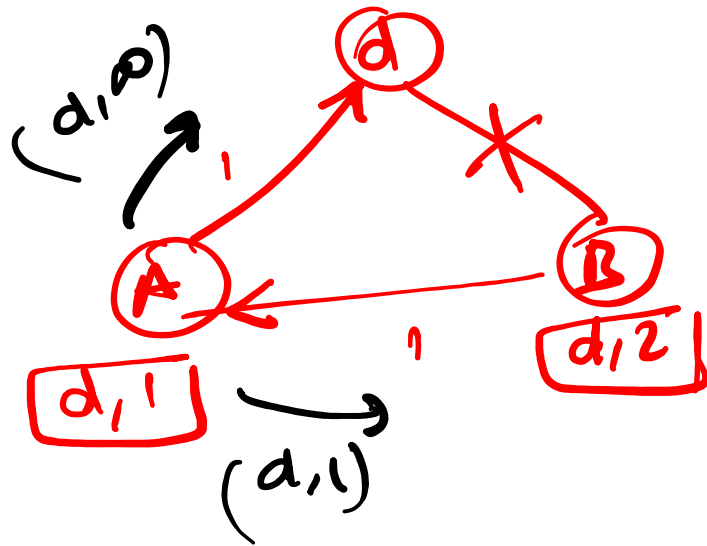


\therefore counting-to- ∞ problem
 "persistent routing loop"

Routing Loops

// Split Horizon with Poison Reverse //

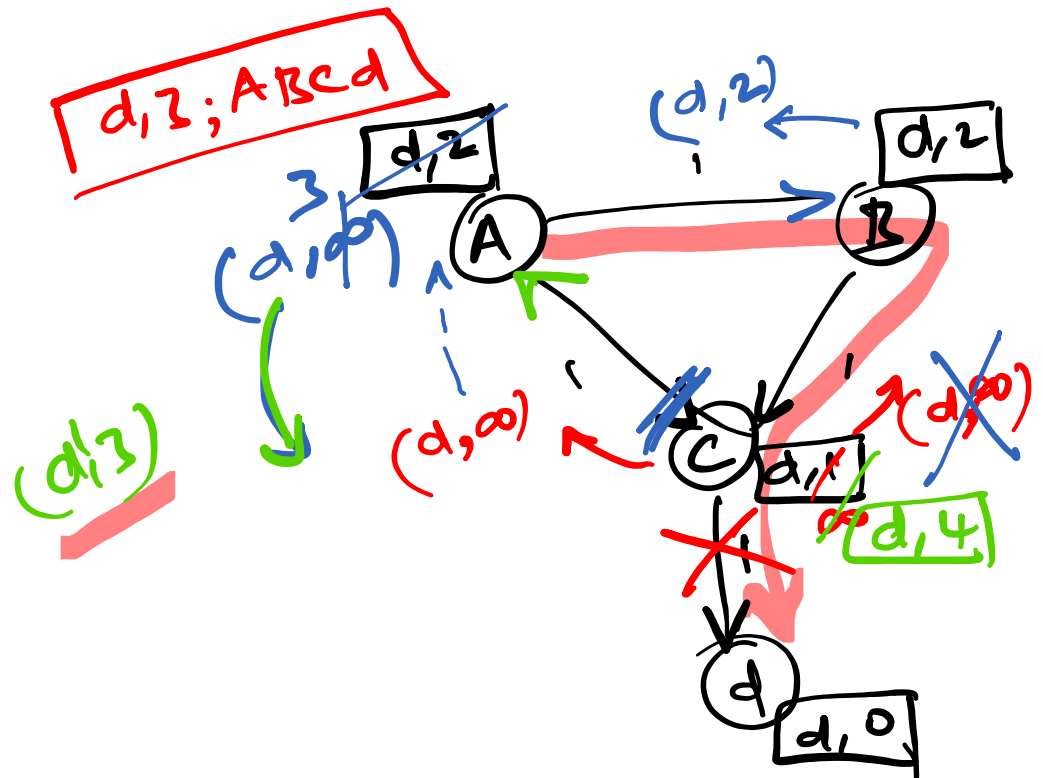
4) Advertise ∞ to your next hop
"lie"



Routing Loops

PATH VECTOR

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



append your id
to distance

Routing Loops

A	B	C	D	E	
•	•	•	•	•	
	INF	INF	INF	INF	Initially
	1	INF	INF	INF	After 1 exchange
	1	2	INF	INF	After 2 exchanges
	1	2	3	INF	After 3 exchanges
	1	2	3	4	After 4 exchanges

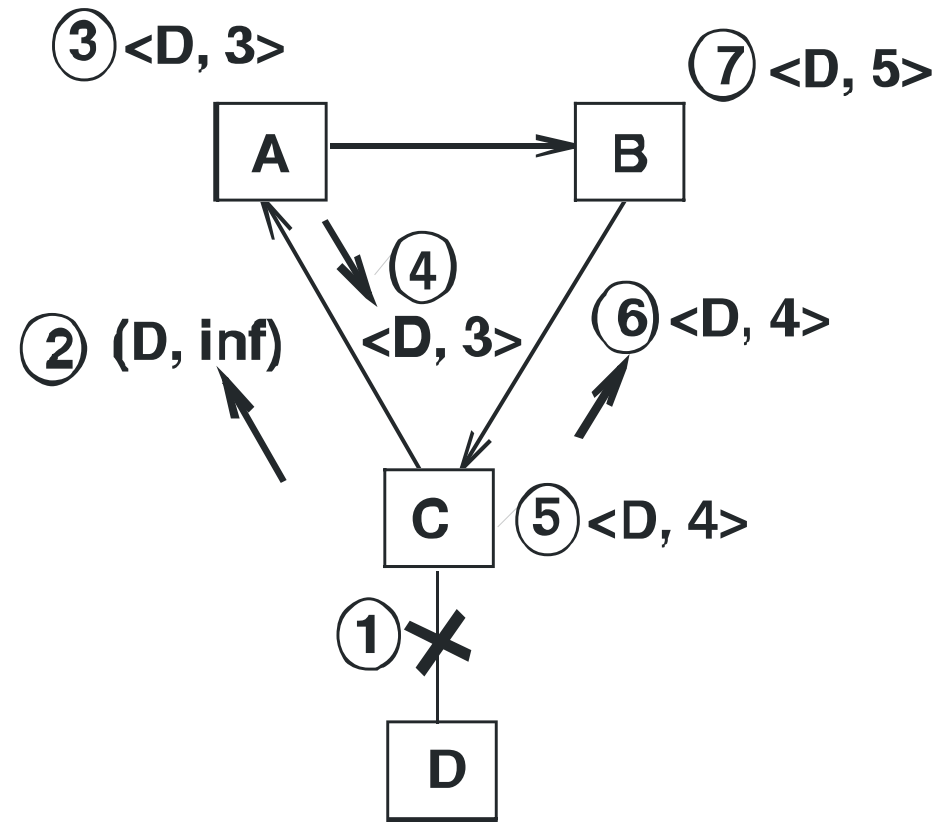
(a)

A	B	C	D	E	
•	•	•	•	•	
	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
	7	8	7	8	After 6 exchanges
		⋮			
	INF	INF	INF	INF	

(b)

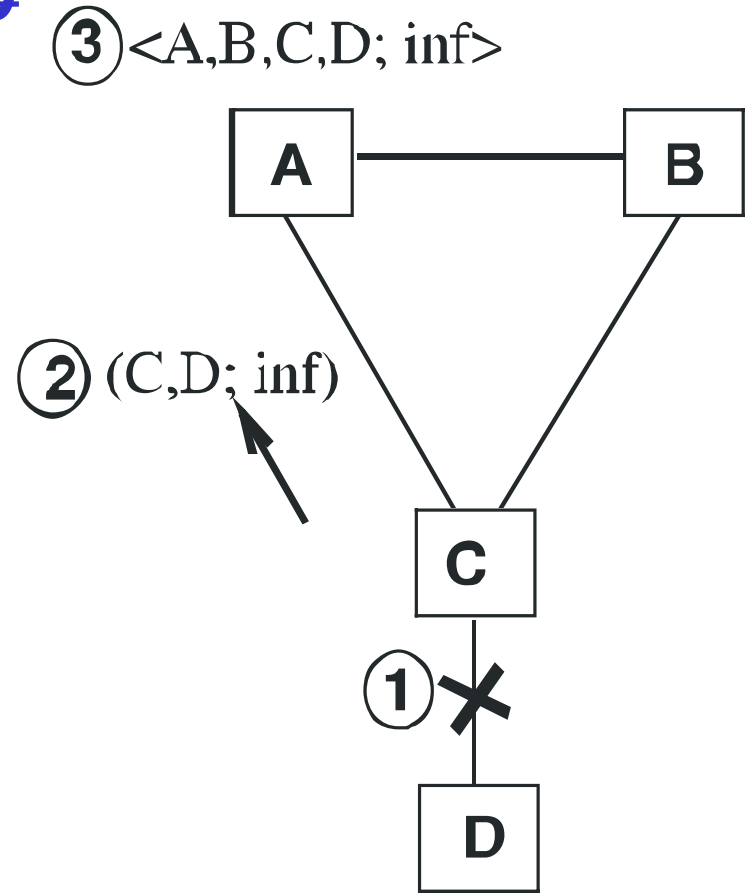
- ❑ Heuristics to break routing loops
 - m set infinity to maximum distance/cost
 - m split horizon: a node does not advertize to its next-hop
 - m split horizon with poison reverse: a node advertizes to its next-hop a distance of infinity

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

+ $O(N * k)$ \nwarrow # 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