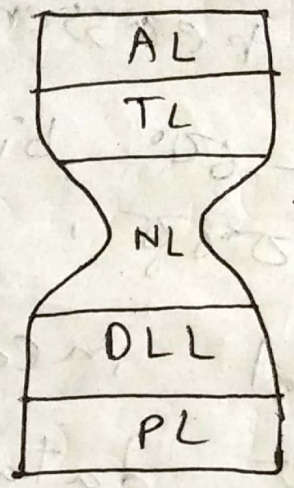
Network layer



Implementation simple হয়।

* connection-oriented → এর connection create হয় এবং এটা Virtual circuit

* connection-less → এর path follow হয় এবং এ Datagram service.

Router → receiving and sending করে, But 3 layers only physical, data-link, Network

→ প্রত্যেক process এর জন্য আলাদা ~~circuit~~ vc. (same or different machine does not matter). So, আলাদা id হবে,

vc number

12/6/23

CSE-321

Desired properties of a routing algorithm

- Correctness
- Simplicity
- Stability (i) should converge quickly
  ii) should work over a long period of time)
- Robustness (should be able to adopt all types of topology changes)
- Fairness
- Optimality

=> Conflicting metrics: Fairness vs Optimality (Slide)

Even in case of optimality, metrics might be conflicting in some cases.

Ex: Delay vs throughput

Throughput maximization through full network utilization

⇓

Network queues are close to their capacities

⇓

Queuing delay is increased

⇓

Total delay is increased

Conflicting!

=) However, in some cases, throughput and delay might not be conflicting

Ex: # of hops↓ (delay↓ BW consumption↓)

Throughput↑

Not conflicting

Routing Algos:

- Adaptive: Dynamic decision based on current traffic and topology
- Static/non-adaptive: Do not use estimates / measurement of current traffic and topology
  - → Shortest path routing (Ex: Dijkstra)
  - → Flooding

Flooding:

- Every incoming packet is sent out on every outgoing line except that one it arrived on.
- Disadv: Generates vast # of (∞, if there is no measure) duplicate packets.

=) **Avoidance :**

packet → প্রতি hop → hop count কমবে। 0 হলে আর যাবে না।

- Hop count (decremented at each hop ; discarded the packet for zero)
- Seq # (list of seq # for each src is stored for discarding duplicates)
- limit the growing list: Augmenting by a counter k summarizing up to k seq # are seen

Same seq # আবার আসলে পাঠাবে না, তাইলে same packet দুইবার send হবে না। Problem: seq # গুলো seen হয়েছে store করতে হবে। Solution, summarize করে store করা। counter → 0-15 আসছে, 18 আসছে। 16, 17 আসলে 18 পর্যন্ত store করবে।
0-15 store
15 store
15, 18 store

=) Selective flooding: Sent to only those lines that are approximately in the right direction.

Extra-info দেওয়া হবে, যেমন destination কোন দিকে, কোন path এ যাবে etc.

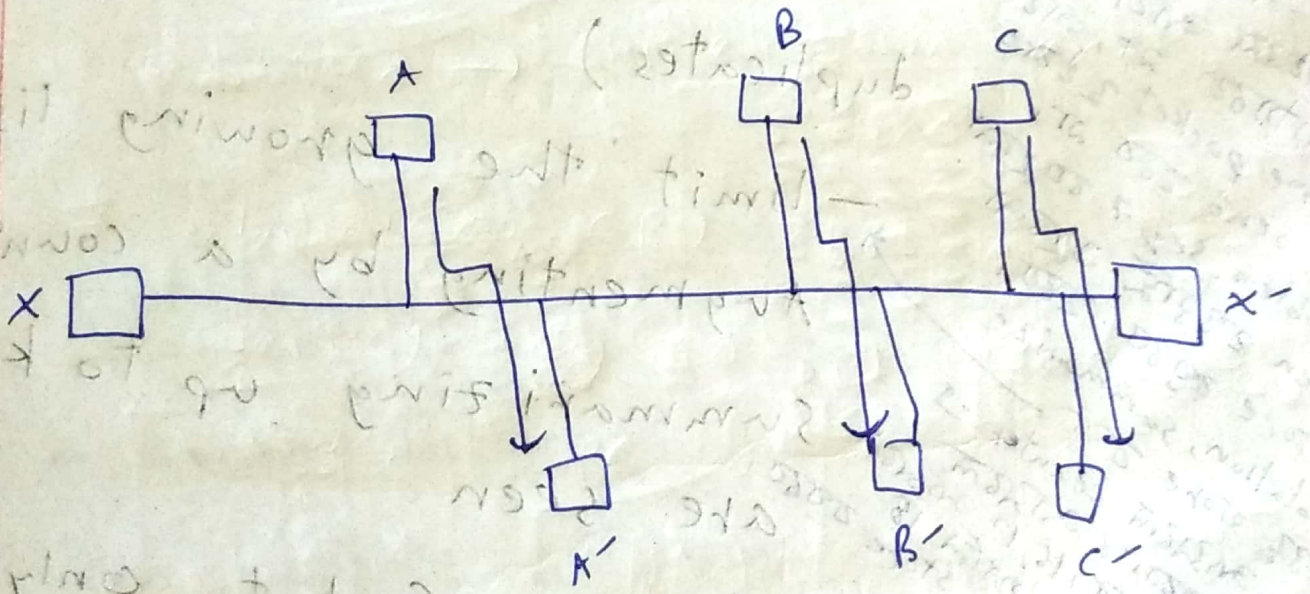
- Application: Military, distributed DB, Wireless network, benchmarking, ...

একসাথে update এ সব জায়গায় update করা যায়।

সবার flooding ensures যে destination এ পৌঁছাবে, যেহেতু সব জায়গায় পাঠানো হয়।

- হয় virtual circuit host থেকে host → host-to-host / external VC

- আর subnet-এ যে VC → Node-to-node / internal VC



X-X' হয়ত আর্থ না, ব্যান্ডউইথ saturate করে যেকোনো capacity.

Optimality principle → Any optimal path's subpaths are also optimal.

Sink tree → ... node ... sink ... optimal path ... tree ...

Distance vector routing

DVR → ... info store ... neighbour ... ...

LSR → Neighbour ... info store ... ...

Link State Routing

Distance Vector Routing (DVR)

- Each router contains a routing table
- Each routing table contains one entry for each router in the subnet
- Each entry contains two info:

① Preferred outgoing line/ next-hop node for corresponding destination router.

② Estimate of the distance to that destination (The best known distance)

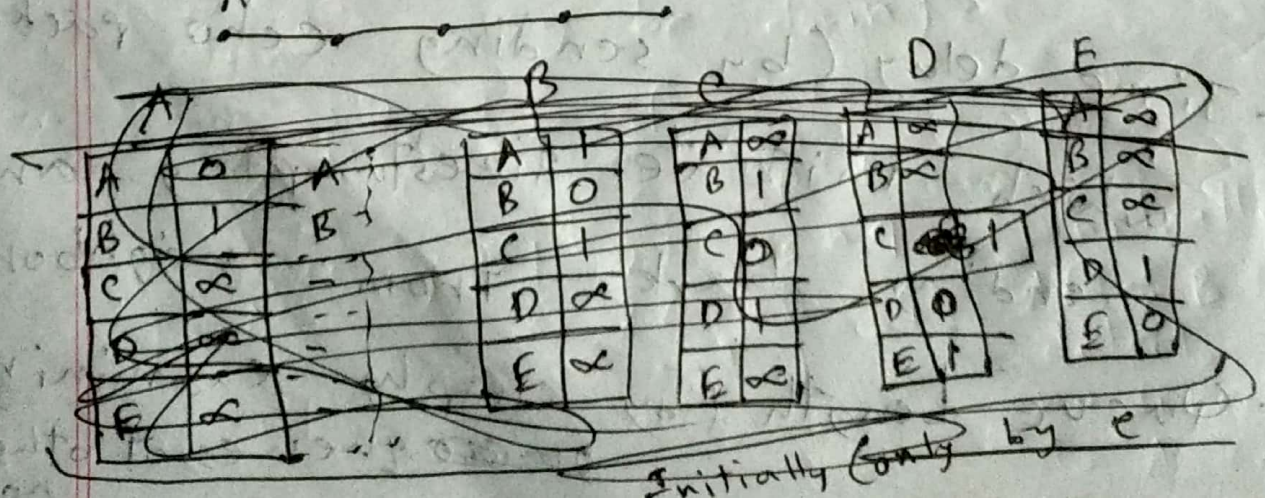
- Different metrics used for estimating distance:
  - # of hops (=1 for a neighbor)
  - Time delay (by sending echo packets, which initiate timestamping and sending back from neighbors)
  - Queue Length (by simply examining queues of other nodes)

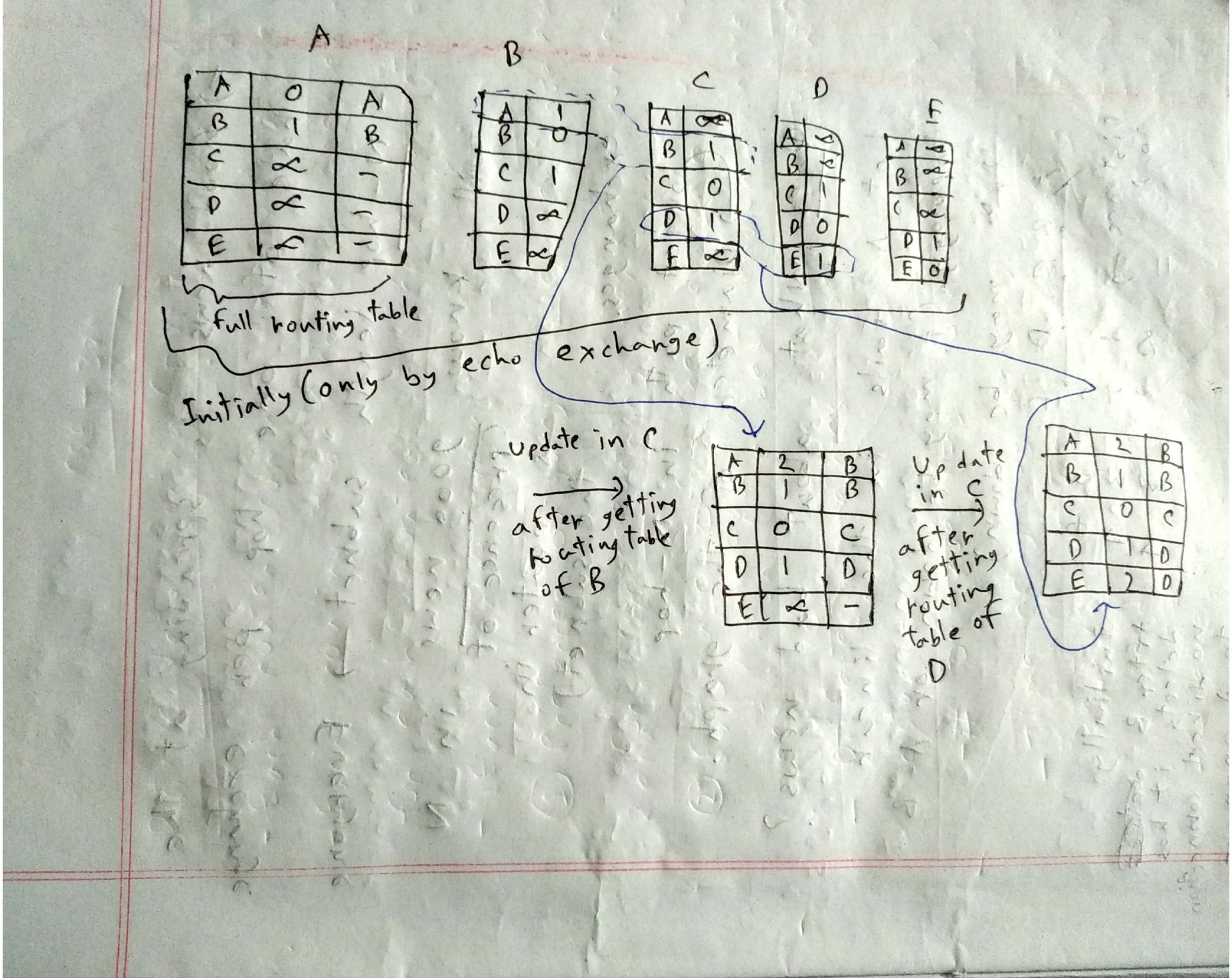
=) Algo:

① Each router exchanges own routing table info with its neighbors

↳ Only distance part (excluding the preferred next-hop) suffices

② After receiving an info from a neighbor, a router updates its own routing table accordingly.




A
B
C
D
F
full routing table
Initially (only by echo exchange)
Update in C
after getting routing table of B
Update in C
after getting routing table of D

⇒) Good news is spreading at the rate of one hop per exchange.

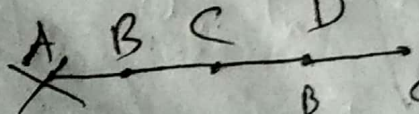
- N-hop length subnet → Everyone will know good news in N exchanges.

(presence of a router in the subnet)

↳ longest path in N-hop
↳ Diameter of the graph.

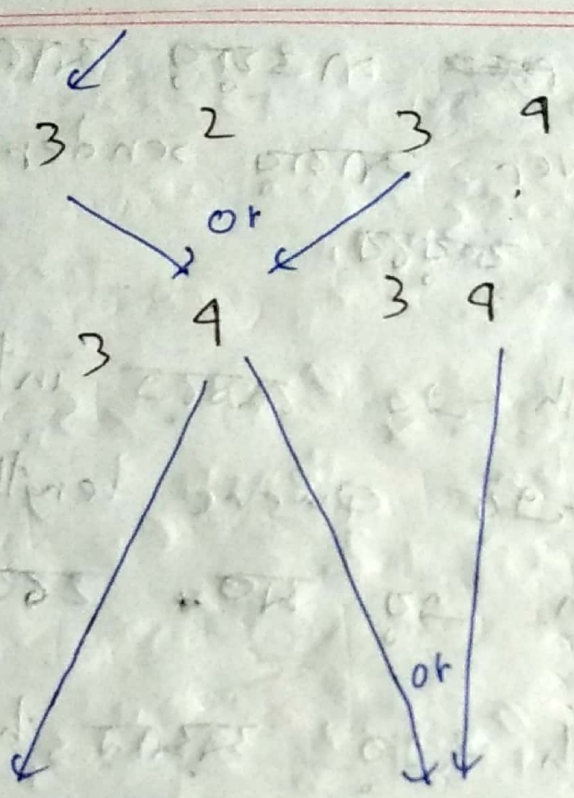
Ques: What about bad news?

Let A fails in the linear topology (A may be down or A-B link may be down)



Distance to A ⇒	B	C	D	E
	1	2	3	4
	∞	2	3	4

Initially
After B ~~detects~~ detects it has no direct connection to A anymore



B thinks it can go to A through C.

C has to update distance to A as both of its neighbors have 3-hop paths to A.

⇒ Bad news travels slowly!!!

Count-to-infinity Problem

↳ Can be set as (longest path + 1)

Reason: No idea about whether I am the next-hop node in the neighbors path (only know its own next-hop) ⇒ Elaboration and solution from algorithmic perspective (book from Patterson)

→ exchanged only between neighbours

- Echo packet ~~pack~~ পাওয়ায় সময় সময় receiver আবার sender এর দিকে send করবে।

- Queue length -এর ক্ষেত্রে intermediate সব node -এর Queue length এর summation এর মত হবে।

- Longest path -এর সমান hop -এ আগেই = আগেইতো ফিরে যায়।

CSE - 321

Algorithmic perspective of DVR (from the book of patterson)

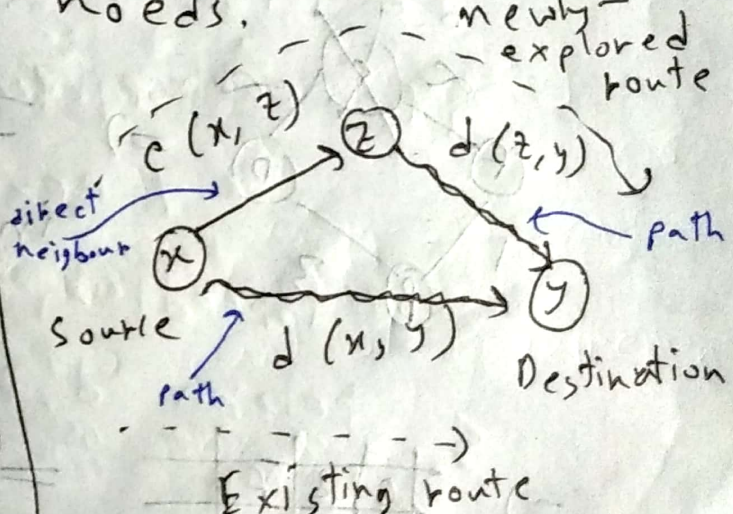
- Initially routing table in each node will have finite distance for only directly-connected noeds.

- Algorithm for update:

Update (x, y, z)

d ← c(x, z) + d(z, y)
  # cost of path from x to y with first hop z

if d < d(x, y)
  # found better route



=> Only direct neighbors are explored here.

return d, z # update cost, next-hop

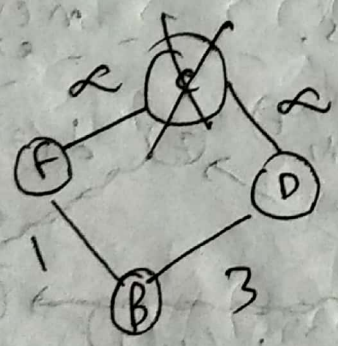
return d(x,y), nexthop(x,y)

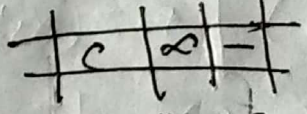
=) Repeat the update until convergence

(Bellman-Ford Algo)

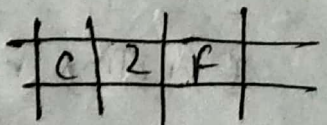
- Time of convergence: Diameter of the graph

=) Impact of node failure:

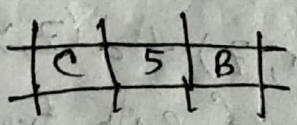


=) 

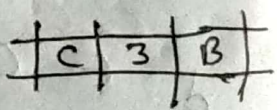
in both F and D's tables

=) 

B's table arrive at F and D

=) 

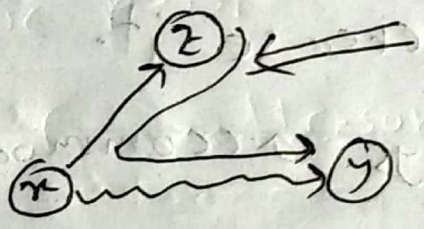
Updated entry in D's table



Updated entry in F's table

(Wrong!!)

=) Solution: while updating, a node has to make sure that it is NOT the next hop on the path from it's first-hop node.



Such circular routes will not be explored.

Update (x, y, z)

d ← c(x, z) + d(z, y)

if d < d(x, y) & x ≠ next-hop(z, y)

found better route

return d, z

else

return d(x, y), nexthop(x, y)

"Split Horizon Rule"

=) Now, fully ok? Ans: NO, as the entry in D's table still mistakenly

gets updated. Reason: B's state table entry to C updates table in D only once, as the algo only decreases the cost / distance, however, does <u>not</u> increase it.

⇒ Solution: Always accommodate the update from next hop node

update (x, y, z)

$d \leftarrow c(x,z) + d(z,y)$

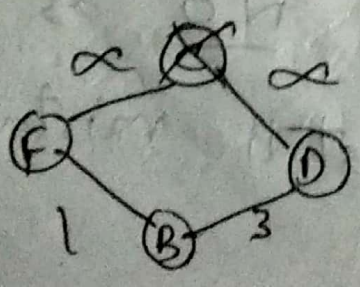
if nexthop $(x,y) = z \mid (d < d(x,y)$ &
$x \neq$ nexthop$(z,y))$

#forced update

return d, z

else

return $d(x,y)$, nexthop (x,y)



⇒

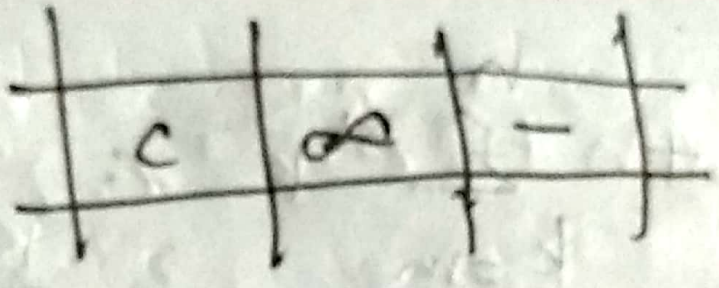
| C | 5 | B |

in D

⇒

| C | ∞ | - |

in B

=> 

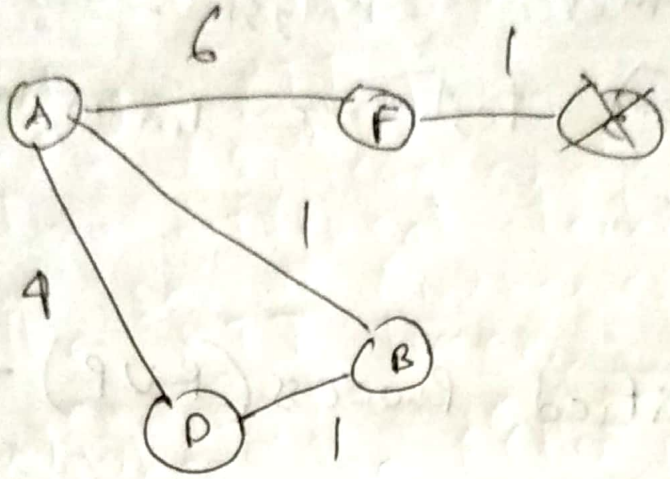
or

in D

=> Ques: All problems are solved??

19/06/23

CSE – 321



Fully solve possible না, unless full path না দেওয়া হয়।

→ Full path দেওয়া থাকে, তাই খুব ~~easy~~ সহজে information পাওয়া যায়, So shortest solution না,

→ Bad news একটা Trigger দিবে, তারপর wait করবে update পাওয়ার জন্য infinite সময় কিন্তু আমরা যতটা সম্ভব as much as possible.

~~Limit~~

Link state Routing (LSR)

- Basic difference with DVR:

DVR: Sends info of all nodes to neighbors

ECHO packet এর [illegible] সাথে সাথে reply দিতে হয়।

LSR - Sends info of neighbors to all nodes.

- Five operational steps in LSR:

Discover neighbors ⇒ Measure cost to each neighbor
⇓
Construct a packet incorporating the cost to neighbours
⇓
Send the info to all other routers
⇓
compute shortest Paths to every other routers.

① Discover neighbors: Identification of neighbors through exchanging special HELLO packets.

② Measuring line cost: Measurement of delay / cost to neighbors through exchanging special ECHO packets. The ECHO

Round Trip Time

Hello মধ্যে Echo আসার interval different হয়।

packet is sent back Immediately.

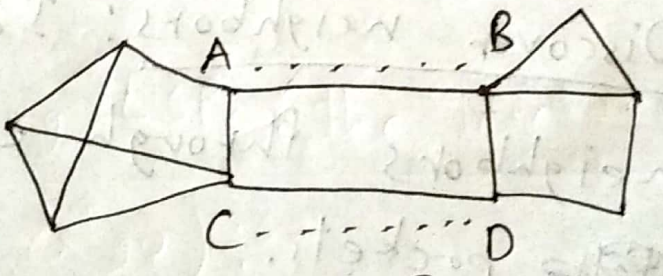
(delay = $\frac{RTT}{2}$)

⇒ Two different techniques to start the RTT timer.

a) When the ECHO packet is queued: Considers load balancing.

b) When the ECHO packet arrives in front of Q: Does NOT consider load balancing.

Load balancing



⇒ A-B and C-D are of same BW, however, experience different traffic loads (A-B > C-D for example). Then,

Flow control → sender और receiver का sending और receiving frequency का मेल करना synchronization.

. the traffic with less traffic (C-D) will be prefered for balancing traffic load.

⇒ Problem of considering load balancing :-

(A-B > C-D traffic) ⇒ choose C-D in new update ⇒ results in

(C-D > AB traffic) ⇐ Choose A-B in new update ⇐ continue ...

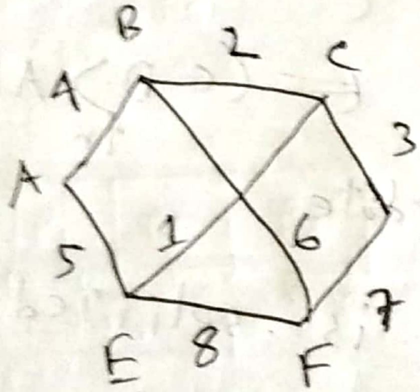
⇒ Solution: Traffic is splitted over multiple lines in fractions, rather choosing only a single line (such split inhibits drastic change in traffic load over a link just after one update).

ঐটা destination এ store থাকে অতক্ষণ হবে। এই age time পর্যন্ত store থাকবে, age 0 হয়ে গেলে [struck] তা হবে না।

③ Constructing Link State packets:

- An LSP ~~co~~ contains: own identity, a seq # (increases during each sending), age (decreases in each sec), List of neighbors

Just like Flooding. Already পাওয়া seq # হলে উটা বাদ দিয়ে দাও।



A	
seq	
Age	
B	4
E	5

B	
Seq	
Age	
A	4
C	2
F	6

④ Distributing LSPs (the trickiest part of the algo):

(Next class)

→ Age decrease হয়ে 0 হয়ে যায়।
- প্রতি hop -এ করণ
- Receiving node -এ holding buffer -এ থাকা অবস্থায়

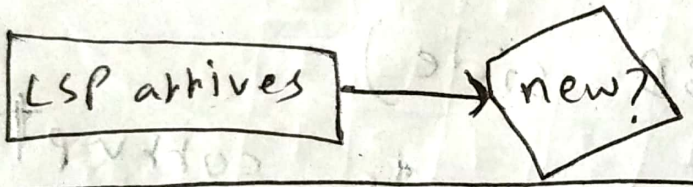
→ Receiving end -এও queue থাকতে পারে। But timer ও sender -এ।
↑
on/off

CSE - 313

True cost

CSE - 321

④ Distributing LSPs (trickiest part of the algo)



Similar to flooding
Extent is kept checked using seq #

Forword to all except the incoming one

⇒ Problems with the algo (mainly with seq #):

① wrapping up of the seq #: (0, 1, ...

max #, 0, 1, ...) seems to be obsolete

- 32-bit seq #; 1 LSP per sec

⇒) 137 years to wrap around

② Router Crash: (0, 1, ... n Crash 0, 1, ...)

Seems to be obsolete

- seq # starts from 0 after enabling a crashed router. So, all LSPs generated from this router (up to previously sent max #) will be discarded.

③ Corrupted seq #: (Huge effect if happen in MSB side)

- Suppose 16th bit gets corrupted (0 to 1) and 65,540 is received instead of 4. All valid LSPs with seq #5 to 65,540 will be rejected in this case.

Solution: Include "age" and decrement it once per second. When the "age" hits zero, info from that router is discarded.

[So, if any of the previous problem would happen, we will end up with only a few wrong discarding events (up to the time age > 0)]

- "Age" is also decremented during initial flooding process (to reduce the extent of flooding).

⇒ Increase robustness of the algo:

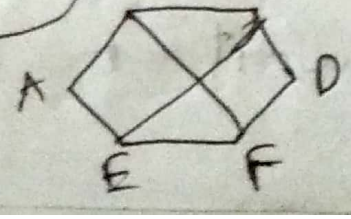
- An LSP, after being received, is not immediately queued for transmission. Instead, it is first put in a holding area to wait a short while.

Age এর hop -এ decrease করে, to limit flooding (just like no. of hops in flooding)

- The info in holding area helps in discarding duplicate LSPs, ~~deter~~ determining where to acknowledge (each LSP is ack-ed) and where to send (excluding the incoming line). The holding area is periodically ~~scc~~ scanned in a round-robin order to select a packet or acknowledge to send.

			send flag			Ack flag			
Src	Seq#	Age	A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	→ Arrive directly from A; So, Ack to A, send to rest
F	21	60	1	1	0	0	0	1	→ " " F; " " " F, " "
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	⇒ If a copy of C arrives through F before performing corresponding forwarding, this entry will become 10 01
D	21	59	1	0	0	0	1	1	

Holding area ~~of~~ in B



⇒ If we would store only the SEND Flag (assuming ACK flag as its complement), then we will have trouble at the ~~beginning~~ beginning (beginning of each period), when all SEND and ACK flags should be 0 (not complement of each other).

(5) Compute shortest paths to every other routers:

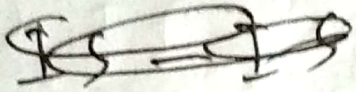
- Construct entire subnet graph after accumulating all LSPs. Each link is represented twice, which can be used separately or averaged.
- Shortest path determination in each node using Dijkstra's algo.

As directed graph

→ DVR localized decision করে, কম computation লাগে। LSR পুরো network নিয়ে কাজ করে, অনেক LSP থাকে, তাই time and space complexity LSR এ বেশি।

→ DVR-এ localized info থেকে decision হয়, LSR-এ সব info নিয়ে centralized decision করে। DVR distributivity support করে, LSR করে না।

→ LSR-এ count-to-infinity নাই, convergence quick. Sophisticated metrics use করতে পারে LSR-এ, কারণ full network-এর info থাকে, multiple path possible LSR-এ, DVR এ এসব support নাই।



LSR example:

IS-IS, OSPF

Internal System

Open Shortest Path First

21/6/23

CSE-301

General methods for sums:

25/6/23

CSE - 321

Hierarchical Routing:

Region,

→ Routing table ছোটো থাকবে, feasible হবে,

→ অসুবিধা: Individual কোনো router-এর জন্য hop বেশি লাগতে পারে।

Router

↳ Region → cluster → Zone → Group

Broadcast Routing:

→ flooding, প্রত্যেককে আশেপাশের packet পাঠাবে,

Problem ↓

Packet count অনেক হবে।
(same route এ same packet অনেক বার যাবে।)
Traffic বাড়বে, Bandwidth waste হবে।

→ multi-destination routing (packet-এ list of routers থাকবে) [এক router packet পাওয়ার পর list-এ যারা আছে তাদের কাছে পাঠিয়ে দেবে]

[Intermediate router, packet সংযুক্ত থাকে, তার কাছে থাকবে সবচেয়ে best way তে reach করা যায়, তাহলে সবচেয়ে Packet পাঠাবে,]

↑ সবচেয়ে best way তে reach করতে পারবে, যেই destination list এ থাকবে,

→ Spanning tree form করবে multi-destination এর জন্য. (LSR এ করলে full topology থাকবে so ST form করা easy. DVR এ করলে full topology, থাকবে না, তাহলে reverse path forwarding লাগবে, তাহলে Intermediate router এ যদি Packet আসে, তার কাছে source এ যেতে যেই path এ তার ঠিকঠাক সবচেয়ে, সেই path এ আসলে, তার receive করবে এবং অন্য Path গুলোয় forward করবে, But, অন্য path

এ packet আসলে, এটা receive, forward করতে বা, থামায় দিবে। এই ডিসিশনে just extra কিছু packet forwarding হচ্ছে (overhead).

Multicast Routing:

→ MST create করতে, যেসব router applicable বা, তাদের prune করে বলা হয়। LSR-এ easy, যেহেতু full topology জানা থাকে। DVR-এ কিছুটা কঠিন কারণ অন্য group → এর router packet আসে, check করে দেখবে, তার কাছে এই group-এর কোনো destination-এ নাই, তার কোনো কাছে packet যেতে হবে না, তখন সিদ্ধান্ত prune করতে হবে, previous router-এ prune signal পাঠায়। Problem হল * ডিফিক্ট router-এ tree maintain করতে

হয়, তখন overhead বাড়ায়। Solution
আবার দিবার জন্য router →
tree তৈরি থাকে, প্রথমে এই router
→ packet পাঠানো হয়, তারপর multicast
করা হয়।