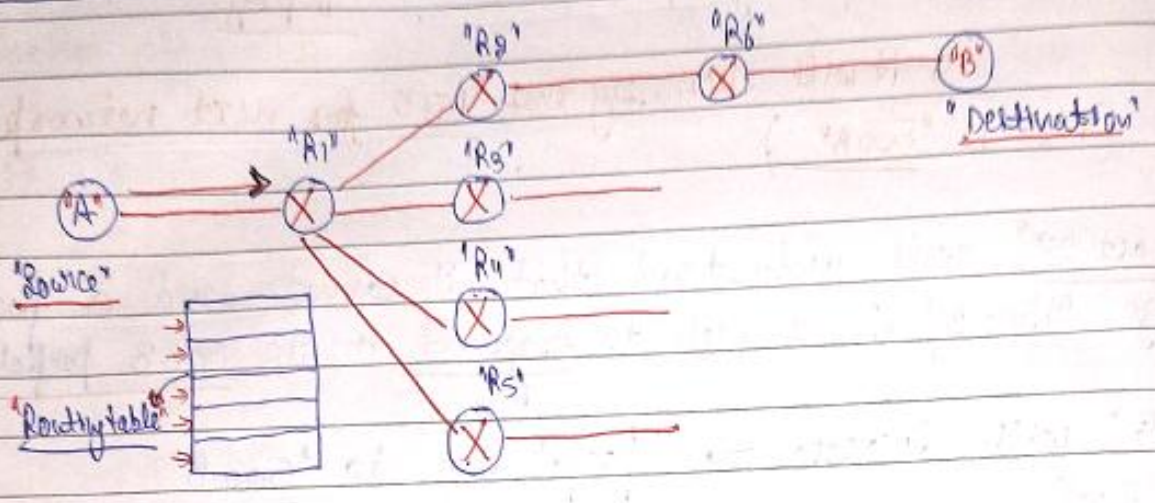
"ROUTING"

Let us see how a "router" sends a "packet" :-



Now when we send a packet to "router "R_1"! How would it know which way to "send the packet"!

↳ whether to send it to "R2" or "R3" or "R4" or "R5".

So, how does a "router" come to know about a "destination" or how does it know about, among all the "outgoing paths" which one is the "best path to follow".

If we can answer it by a "Routing-Table" then this "procedure" is known as "Routing".

So, at routers "R_1" we have a "routing table" so we take the "packet" & search for all the "rows" in the "routing-table".

The process of constructing a "Routing-Table" itself is known as "Routing".

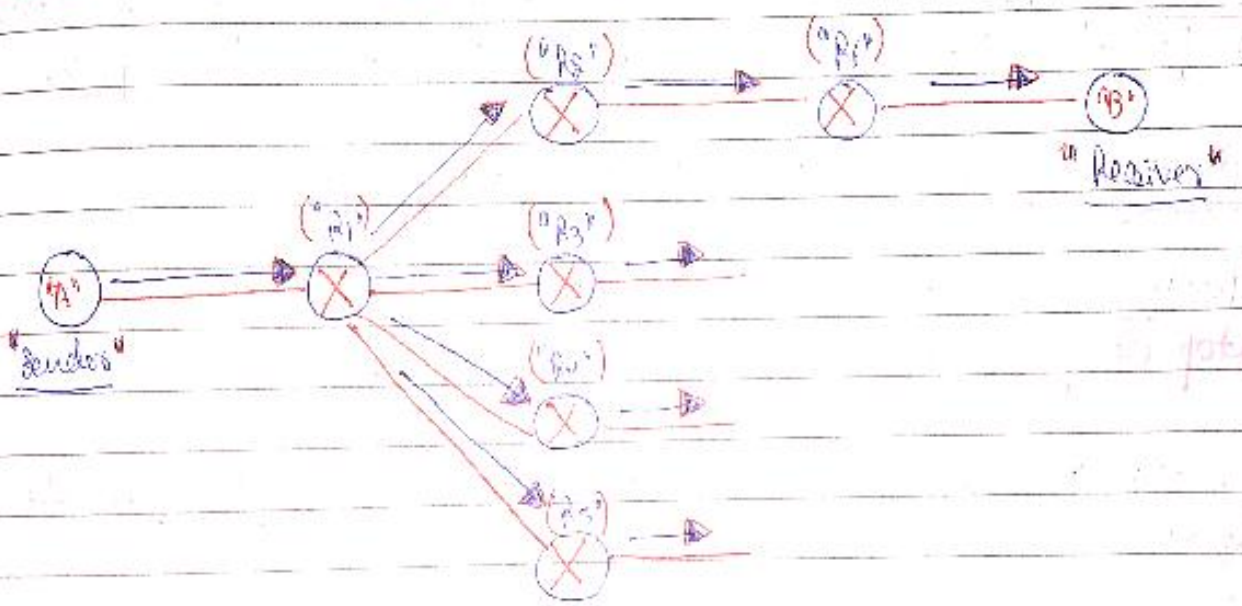
[Taking an "incoming packet" & sending it out is known as "switching" & not as "routing".]

→ "Routing" :→ The process of creating a "Routing table" at every router is known as "routing".

Even if we donot have a "routing table" still we would be able to "survive".

↳ There is another method known as "flooding".

"Flooding" means, if we donot know which way to "send a packet" Then the best way to send the "packet" is to send it to all the available "interfaces". (towards all possible directions).



So, if "R1" doesnot know which way to send the "packet", then it can send it in all the "possible directions".

So, if "every router" use this method of "flooding" then at-least one packet will reach the "receiver".

Let us now see the difference b/w "Routing & Flooding" :→

P.T.O

"Flooding" :→ It means taking a "packet" from incoming interface & putting it on all the "outgoing interfaces", except the one on which "it has arrived".

"Advantages of Flooding" :→

1> No. ('routing tables') are required.

2> (Shortest path) is always (guaranteed).
(because "packets" are send on all the "outgoing paths")

3> (Highly reliable) :→ Since we are "sending packets" towards destination through all the ways/paths even if one path is down the packet might reach the "destination" through "other path".

"Disadvantages of Routing" :→ (Compared to flooding)

1> "Routing tables" are required. (At every routers we must stop the packet & do some processing on it.)

2> Shortest path is not "guaranteed". (It depends on the algorithm we use.)

3> "May not be reliable" because if the chosen path for delivering a "packet" to "destination" is "itself down".

:→ "Disadvantages of flooding" :→

1> "Destination" is going to get many packets (Duplicate packets will arrive)

2> Since a single packet is multiplied many times, then traffic is increased.

"Various Protocols"

"Layer"	"Protocols"
"Application":	DHCP, DNS, Gopher, NNTP, NTP, SIP, SSI, NFS, Netconf, SMTP, HTTP, FTP, Telnet, IMAP, POP, BOOTP
"Presentation":	MIME, TLS, SSL, XDR
"Session":	Named pipes, Net BIOS, SAP, SIP, ZIP, PPTP, L2TP
"Transport":	TCP, UDP, SCTP, DCCP
"Network":	IPv4 (4,6), ICMP, IPSec, IGMP, IPX, Apple Talk, All routing protocols, RARP, ARP, DHCP, BOOTP
"Physical":	ATM, X.25, Frame Relay, Ethernet, 802.11, 802.15, 802.16 USB, Bluetooth, DSL, SONET/SDH

→ "Advantages of Routing" :→

1) No duplicate packets.
2) low traffic.

∵ in "internet" we are going to use "Routing"

⇒ "Types of Routing-Algorithms" :→

"Routing"

- "Static"
  - "Not scalable"
  - (Everything has to be done manually)
- "Dynamic"

"Static-Routing-Algorithms" :→ If "routing-tables" are manually prepared & uploaded offline, then it is known as "static" i.e. it is done manually.

The problem with this is that there is no such entity which can keep track of all the "routers" in the "world".

Depending upon the changes in "topology of the Network" & traffic these algorithms donot change the routing tables themselves.

"Dynamic-Routing-Algorithms" :→ There is no manual intervention. Everything is done by the "router" itself.

Depending upon changes in the "Network topology" and "traffic" these algorithms will change the "routing tables" automatically.

Note :→ "Dynamic-algorithms" are generally used in Internet whereas "Static-Routing-Algorithms" are not used.

"Dynamic Routing Algorithms"

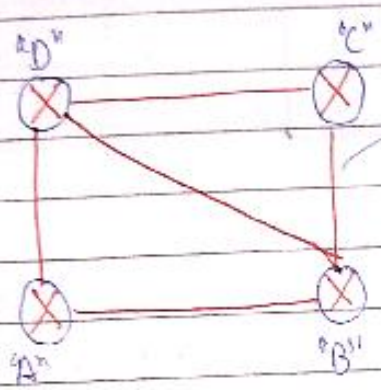
- "Distance vector Routing"
- "Link-State Routing"

"Distance-Vector-Routing Algorithm" :→

very popular in 1980's

Explanation of "Distance Vector Routing Algorithm" algorithm with the help of an example.

Here in "Distance-Vector-Routing" we are worried about the "Routers" & "shortest distance b/w routers".



→ These are the "links" which are "connecting the routers".

There will be some number associated with these links which will either be :→

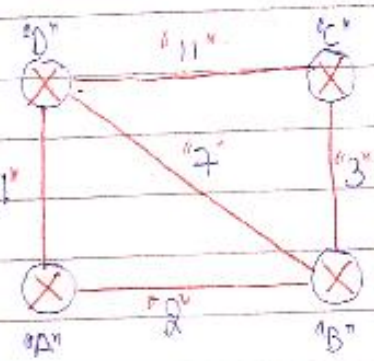
↳ "Cost" of the "edge" or "link".

↳ "Distance" or traffic or queuing delay etc

It means we can "interpret" the number associated with these "links" in many ways.

Our motive is to find the path having "least edge weight / cost" b/w every pair of "vertices".

Eg :→



First step is at "each router" we will maintain a "local-routing-table".

Note :⇒ In "DVR" every "node" knows only about its "neighbour". It doesnot know about a "node" which is not its neighbour". (& "every node" knows how many routers are there in total)

Each "local-routing-table" will have "three fields" :⇒

"Destination"	"Distance"	"Next Hop"

"Local Routing Table" at "B" is as :⇒

"Dest"	"Distance"	"N.H"
A	2	A
B	0	B
C	3	C
D	7	D

This is the "table" prepared by node "B" using its local knowledge.

"Local Routing Table" prepared by "A" is as :⇒

"Dest"	"Distance"	"N.H"
A	0	A
B	2	B
C	∞	—
D	1	D

(becoz "A" is not a neighbour of "C")

"Local-Routing-Table" prepared by router "C" is as :⇒

"Dest"	"Distance"	"N.H"
A	∞	—
B	3	B
C	0	C
D	11	D

"Vector in Computer Science means an Array"

→ "Distance vector"

"Local-Routing-Table" prepared by router "D" is as :→

"Dest"	"Distance"	"NxH"
A	1	A
B	7	B
C	11	C
D	0	D

"Step no 2" :→ Now every router will take its "distance vector" & exchange it with their "neighbours".

Eg :→ "Router "C"" will take its "distance-vector" & exchange it with "B" & "D".

So "Every node" is going to get a "distance-vector" from all of its "neighbours".

Let us see what happens at "router "A"", At "A" it will get a "distance-vector" from "B" & "D".

At "A" 'DV' are received from B, & D

At "B" 'DV' are received from A, C & D

At "C" 'DV' are received from B, & D

&

At "D" 'DV' are received from A, B & C

Everything happens here in parallel.

"P.T.O"

⇝ ("many nodes" in b/w)
→ ("single node" directly)

Now at "every-Router", using these distance vectors new routing tables will be created.

Let us see what happens at "A".

"From "B" "DV" is" ⇒

2
0
3
7

"From "D" "DV" is" ⇒

1
7
11
0

"Note" ⇒ Every "Router" will use the information from its "neighbours" to compute "new routing tables" & not based on "old" routing table".

New routing table at 'A'

A → A directly connected ∴ '0'

"Dest"	"Distance"	"N-H"
A	0	A
B	2	B
C	5	B
D	1	D

New Routing table at "A"

send it to "D".

Ask "D" to send it to "B".

$$\text{'}A \rightsquigarrow B\text{'} = \min \begin{cases} \overset{①}{A \to D} + \overset{②}{D \rightsquigarrow B} \\ \underset{②}{A \to B} + \underset{⓪}{B \rightsquigarrow B} \end{cases}$$

$$= \min \begin{cases} 8 \\ 2 \end{cases} = \boxed{2}$$ direct hop = "B"

Date ___/___/___
Page _____

"edge" "path"

Now "$A \rightsquigarrow C$" $= \min \begin{Bmatrix} A \to D + D \rightsquigarrow C \\ A \to B + B \rightsquigarrow C \end{Bmatrix}$

$= \min \begin{Bmatrix} 1 + 11 \\ 2 + 3 \end{Bmatrix} = \min \begin{Bmatrix} 12 \\ 5 \end{Bmatrix} = \boxed{5}$

Next Hop = "B"

Now "$A \rightsquigarrow D$" $= \min \begin{Bmatrix} A \to D + D \rightsquigarrow D \\ A \to B + B \rightsquigarrow D \end{Bmatrix}$

$= \min \begin{Bmatrix} 1 + 0 \\ 2 + 7 \end{Bmatrix} = \min \begin{Bmatrix} 1 \\ 9 \end{Bmatrix} = \boxed{1}$

Next Hop = "D"

Easy way to "Construct" "new routing table" (Short cut Method) :=>

Note "A to A" is always "zero" & "next Hop" is "A" itself
"It is an exception".

From "B"

2
0
3
7

From "D"

1
9
11
0

"Now" AB = 2 AD = 1

"New Routing table at A"

"Dest"	"Distance"	"N-H"
A	0	A
B	2	B
C	5	B
D	1	D

Now for every action zero except row 1.
take the "row 1" & add it with corresponding "row" then take the minimum
& "next hop" will be, where u get "min-value".

"P.T.O"

Now let us see the New routing table at router "B"

"B" will receive "Distance vectors" from "A" "C" & "D".

"A"	"C"	"D"
0	∞	1
2	3	7
∞	0	11
1	11	0

from figure (BA = "2" BC = "3" BD = "7")

["New routing table" of "B"]

"Dest."	"Distance"	"N.H"
A	2	A
B	0	B
C	3	C
D	3	A

Note :→ Take "distance vectors from previous round".

New "routing-table" at router "C" :→

"C" will receive "Distance-vectors" from "B" & "D"

"B"	"D"
"2"	"1"
"0"	"7"
"3"	"4"
"7"	"0"

(CB = "3" CD = "11")

["New routing table" at "C"]

"Dest"	"Distance"	"N-H"
A	5	B
B	3	B
C	0	C
D	10	B

New "routing-table" at "router "D" :→

"D" will receive "Distance-vectors" from "A" "B" "C"

["New routing table" of "D"]

	"A"	"B"	"C"
DA = "1"	0	2	∞
DB = "7"	2	0	3
DC = "11"	∞	3	0
	1	7	11

"Dest"	"Distance"	"N-H"
A	1	A
B	3	A
C	10	B
D	0	D

Note:→ If no of nodes is "n", then shortest path b/w two nodes will never contain more than "(n-1) edges".

Note In "Round ①" we get set of all shortest paths where shortest path contains at most "One edge".

In "Step 2" or "round 2" we get set of all shortest paths where shortest path contains at most "two edges".

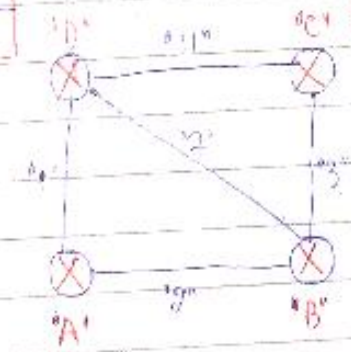
So for this example we have to run this "algorithm" one more time (for third time) :→

But in exam we will solve it by our "intuition" :→ "for gate"

Given a "graph" how to find "final "routing tables" :→ Use "intuition"

"Dest"	"Dist"	"NH"
"A"	1	A
"B"	3	A
"C"	6	A
"D"	0	D

"Dest"	"Dist"	"NH"
"A"	5	B
"B"	3	B
"C"	0	C
"D"	5	B



"Through Intuition"

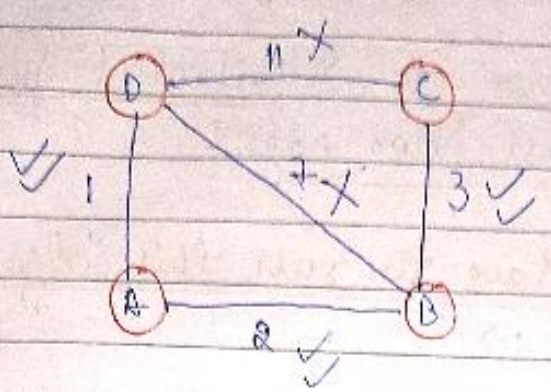
"Dest"	"Dist"	"NH"
"A"	0	A
"B"	8	B
"C"	5	B
"D"	1	D

"Dest"	"Dist"	"NH"
"A"	8	A
"B"	0	B
"C"	3	C
"D"	8	A

Other Question they can ask about "Routing tables" is

Given a "graph" what are the "edges" that will not be used after "routing tables converged"?

for "an edge" if there is an "alternate better path available" then that "edge" is not used.



So "only edges" used are:→

[AD, or DA
AB or BA
& BC or CB]

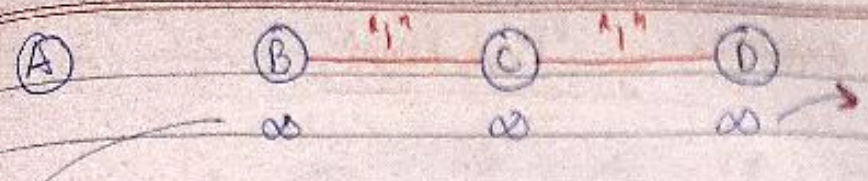
"Count to Infinity Problem" :→

One problem with "Distance Vector Routing" is "Count to Infinity Problem" :→

"Count to Infinity" :→ ("Bad News spreads slow")
("Good News spreads fast.")

Eg:→ ("A", "B", "C" & "D") are "four routers" as shown below :→

Hop distance b/w [B & C] is "1" & b/w [C & D] is also "1".
Initially assume "A" is not connected
i.e Distance from "B" to "A" is "∞" & similarly distance from "C" to "A" & "D" to "A" is also ["∞"].

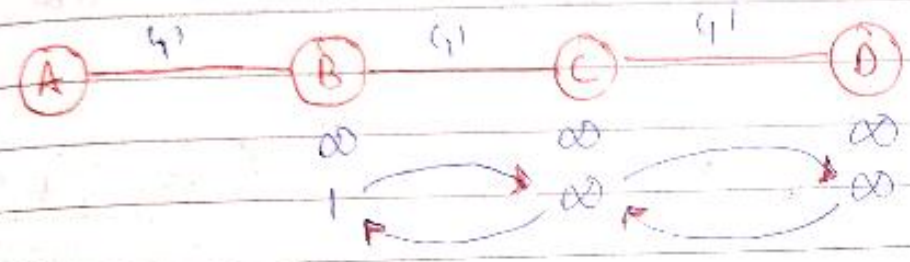


These are "distance to "A" from respective routers

Here we are talking about "distance" of all other 'nodes' to node "A" only.

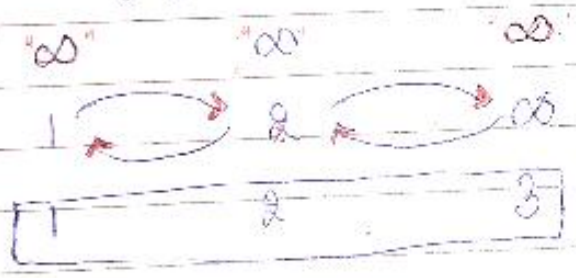
Now Assume that "A" is suddenly connected to "B".

→ Now "A" gets connected & "Hop distance" b/w "A" & "B" is "1" so immediately routing table of its immediate neighbour i.e "B" will get updated, as shown below



Now in next round there will be an exchange of distance vectors.

"B" is saying "C" that my distance to "A" is "1" & "C" knows its distance to "B" is "1" ∴ it ("C") is going to update ∞ to 2.

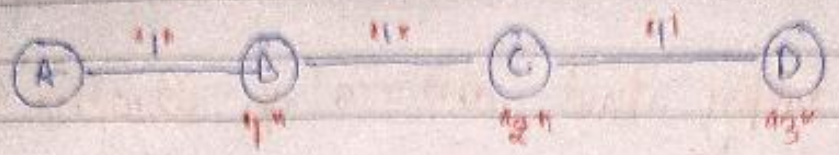


"C" will say "D" that it can reach "A" in "2" & "D" knows that its distance from "C" is "1", so it will update ∞ to "3"

There is a "Good news" & "Good news" is that "A" is working, & "Good news" is "spreading fast".

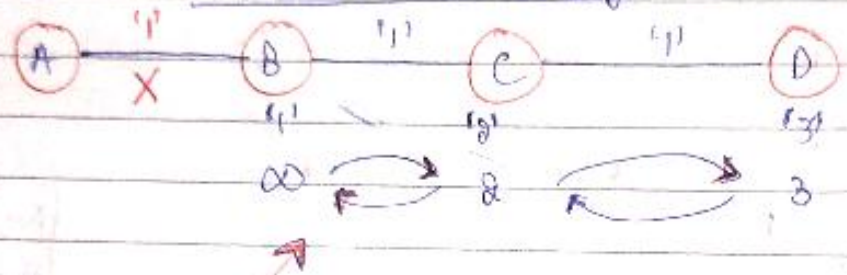
P.T.O

Now let us see an example of "Bad news"



Distance from "B" to "A" is = '1'
Distance from "C" to "A" is = '2'
& Distance from "D" to "A" is = '3'

Now suddenly the link b/w "A" & "B" is down. So "B" will know about this bad news immediately & it will update '1' to '∞'

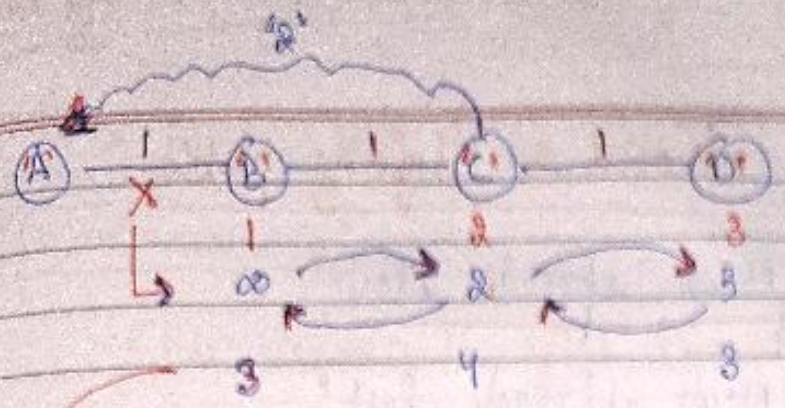


(becoz 'C' & 'D' doesn't know anything about "A")

We know that "routing tables" are updated at a particular router based on the information it received from its neighbours!

When exchange happens, "C" is saying to "B" that I can take you to "A" in "2 hop distance" but it is not saying it that its path is through "B" only (that is a mistake) because we exchange only the distances, but not the "next hops"

Then "B" will accidentally "falsely" think that "C" must be having some path to reach "A" but do not know that it itself is the next hop. (this is an error here)

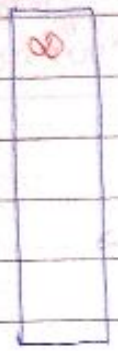


[At "D" it will remain 3 bcoz 3+1 = 3]

At "C" from one side it is getting '∞' & from other side it is getting '3', so it will update '2' to "4"

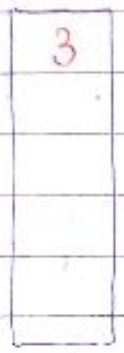
At "C"

"Dv" From "B"



"CB=1"

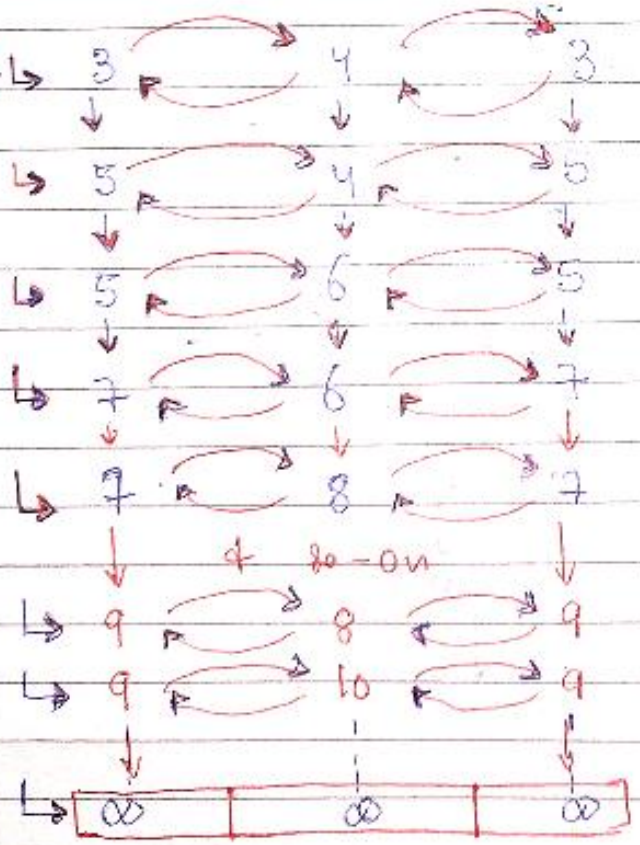
"Dv" from "D"



"CD=1"

"Now updated Routing table at C"

"Dest."	"Dist."	"N-H"
A	4	D



[Even though "previous no" is less, we should not stick to it bcoz we consider routing tables of our neighbors for updation.]

This keep on happening until we reach infinit

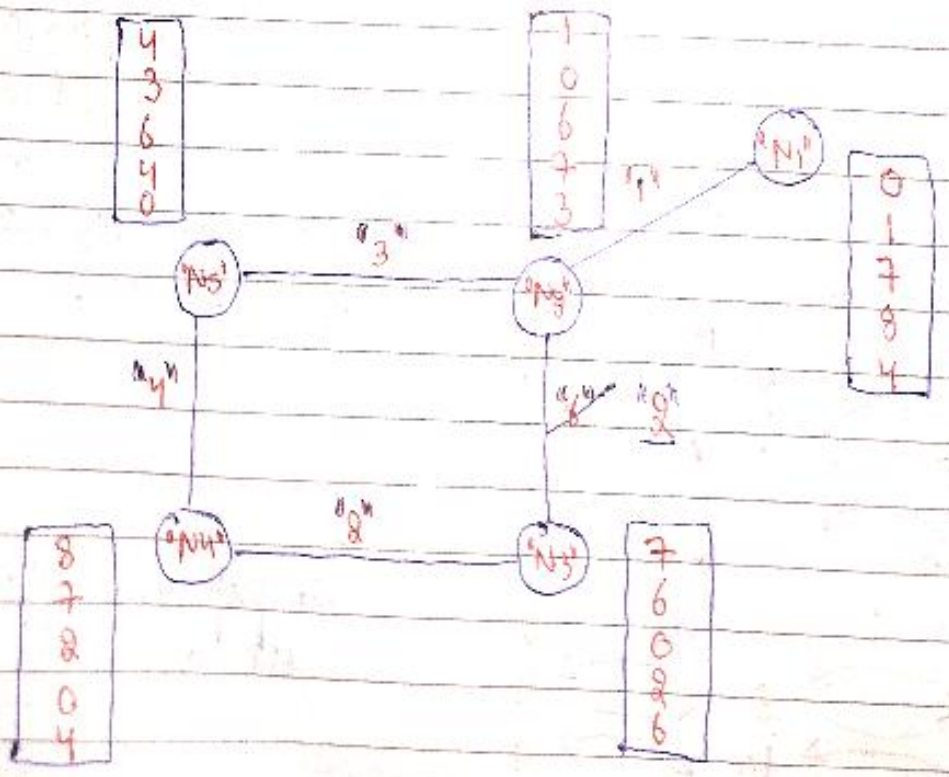
This is known as "Count to Infinity problem"

∴ in "DVR" "Bad News spreads slow"
&
"Good News spread fast"

This is becoz in "DVR" we are sending "Distance vectors" only & not the "next hops"

This "Algorithm" will never "stop" becoz we want our "algo" to be "dynamic in nature" in order to find out that some change might have taken place in the "network".

Example of "DVR" from "GATE point of view" :→
"GATE-2003"



Note:> "Convergence rate of DVR is very slow"

Date / /
Page

"Five Routers" ("N1", "N2", "N3", "N4" & "N5") are given along with their final "Distance-Vectors".

This scenario is after "Convergence".

Here in this question they are asking that if distance b/w "N2" & "N3" changes from "6" to "2" then what would be the new distance vector at "N3" after "one round of exchange."

Sol:> Let us see what happens at "N3"

At "N3" we are going to get "distance-vectors" from its immediate neighbours i.e. "N2" & "N4".

from "N2"

1
6
6
7
3

from "N4"

9
7
2
0
4

New routing table at "N3"

"Dest"	"Dist"	"Next Hop"
"N1"	3	N2
"N2"	2	"N2
"N3"	0	N3
"N4"	2	N4
"N5"	5	N2

"N3N2" = "2" "N3N4" = "2"

∴ "distance vector" at "N3" after "one round" is:>

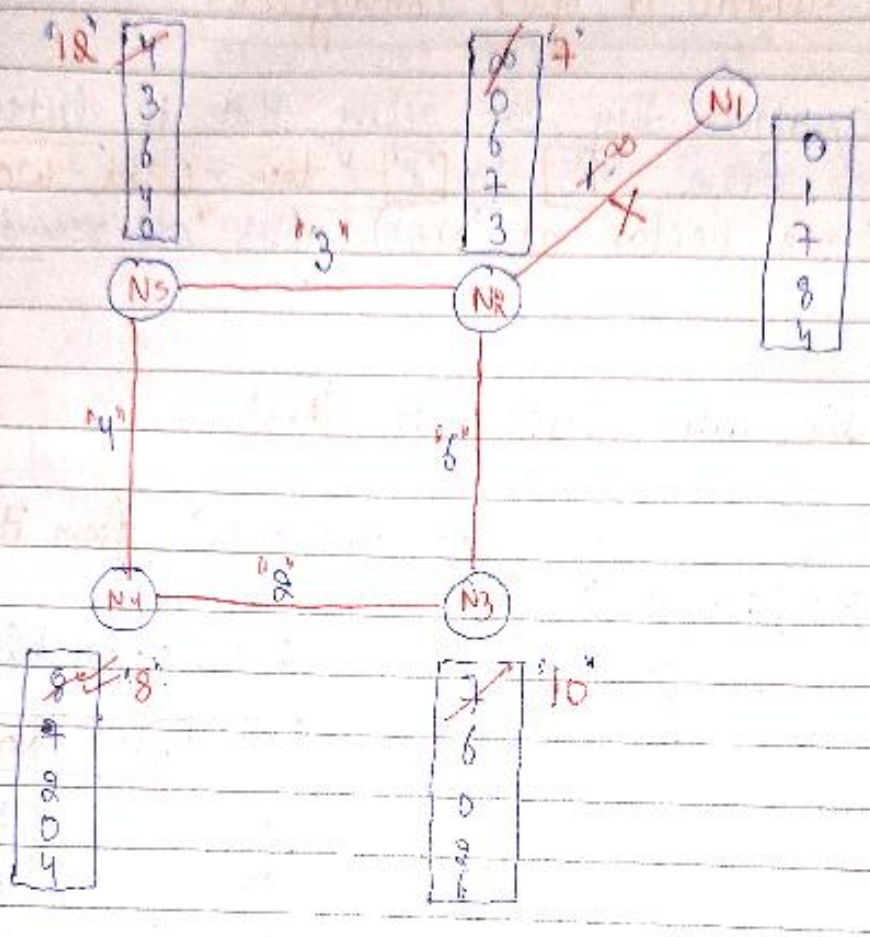
3
2
0
2
5

"Answer"

PTO

Note:→ In "DVR" updates are sent periodically so that if there is any change in the "network topology" every node would get the information about the change.

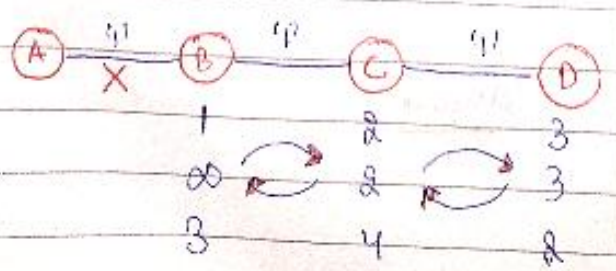
Now suppose one of link b/w [N2] & [N1] goes down. Now count to infinity scenario after 1 round at all the routers. It are



[In "next round" again the same thing will happen till "∞" (infinity) is reached.]

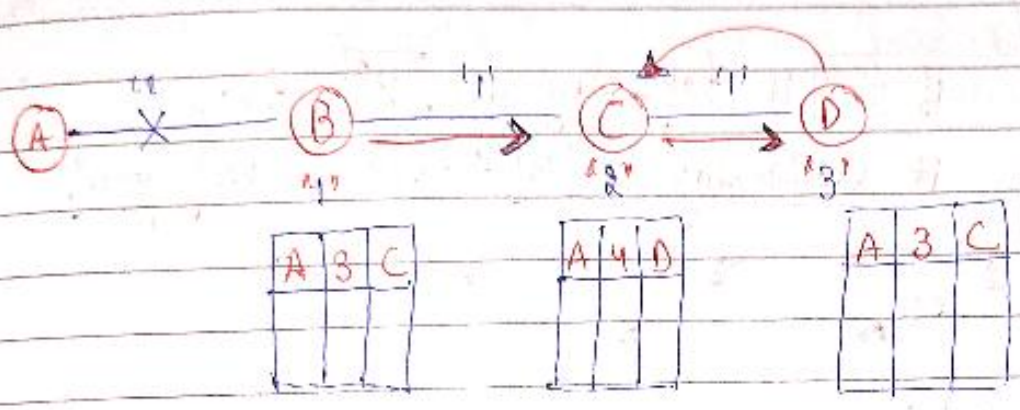
The problem with "count to infinity" is that it not only gives wrong routing tables, but will also "create loops".

eg:→



It means at round '3' if 'B' wants to send a packet to 'A' it will send it to 'C'. If 'C' wants to send a packet to 'A' it will send it to 'D' & if 'D' wants to send a packet to 'A' it will send it to 'C'.

Now 'packet' will continuously fallen into a 'loop'. This is the disadvantage of 'count to infinity problem'.



The problem of "Infinite loop falling" can be solved by "Split-horizon-concept".

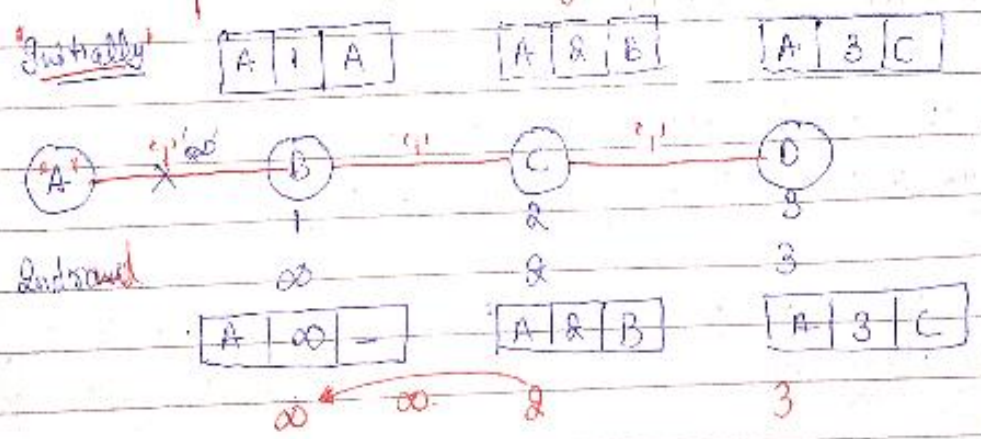
It is as explained below :-

[If someone is already dependant on you then donot entertain him depending on you]

'or'

[If you are 'dependant' on 'someone', then donot let that 'person' depend on you.]

'Initially'

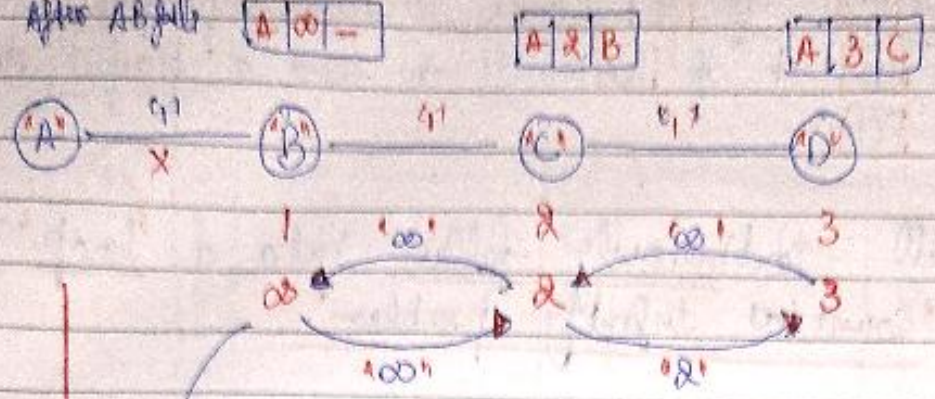


[Since 'C' is already depending on 'B' to send a packet to 'A' so it should not let 'B' depend on 'C']

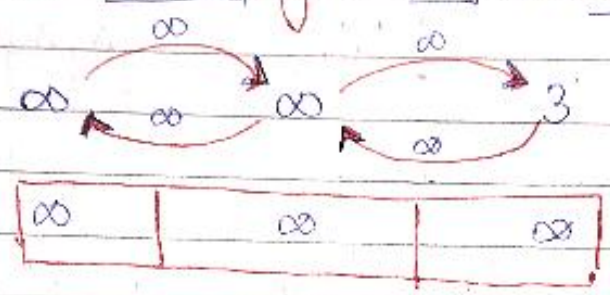
Hence send ∞ instead of ~~it a real value~~

"Split horizon solution" is as follows

After AB fails



If 'C' is depending on 'B' then it should say to 'B' that "I can help you".



∴ By using "split horizon" "convergence" is "very fast" & there will be "no loops".

"Split horizon" is used to solve "Count to Infinity problem" & we can even avoid falling into infinite loops

"LINK-STATE-ROUTING" :→

Since 1980's as the "bandwidth" was very limited, people were sending only "distance vectors" i.e. limited data.

Coming to 1990's the bandwidth got improved because of advancements in technology, hence we are able to send more data

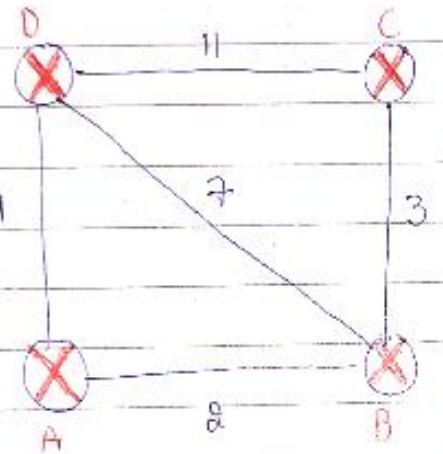
"Link" is an "edge" & "state" means if it is "up" or "down".
Now if it is up what is the cost associated with it.

The disadvantage of "distance vector Routing" was "count to infinity" problem.
& also convergence of "DVR" is very low.

"LSR" is somewhat "superior to DVR" but there are some disadvantages also.

Explanation of "LSR (Link State Routing)" with the help of an example is

In "DVR" every "router" has created a "local routing table" first.
But here in "LSR" every router will create a "link state packet".



"Link-state-packet" at "B" is ⇒

"B"	
"Seq. No"	
"TTL"	
'A'	'2'
'D'	'7'
'C'	'3'

"Link-state-packet" at "C" is ⇒

"C"	
"Seq. No"	
"TTL"	
'D'	'11'
'B'	'3'

"P.T.O"

"DVR" is based on "Local Knowledge" whereas "LSR" is based on "Global Knowledge"

"Link-State-Packet" at "A" is :→

"A"	
"Seq. No"	
"TTL"	
"D"	"1"
"B"	"2"

"Link-State-Packet" at "D" is :→

"D"	
"Seq. No"	
"TTL"	
"A"	"1"
"B"	"7"
"C"	"11"

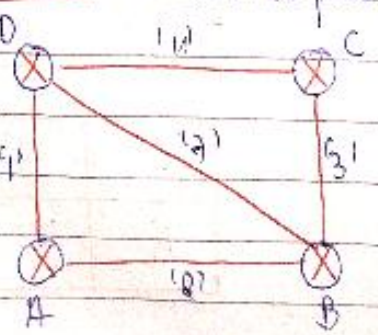
So, "first step is" every "Router" will find out who are its neighbors, what are the links & what are the "weights" associated with them, by sending "Hello Packets"

Next step is unlike "DVR", in "LSR" every "Router" is supposed to flood the information to every "other Router" in the "topology".

Note:→ Only for "Link-state-Packets" we are using "flooding".

"In Second step":→

Let us see what happens at "A" :→ It will receive "LS packet" from every other Router in the topology i.e "B", "C" & "D".



"A" has Global database

Now "A" at this point will apply "Single source shortest path algorithm" i.e. [Dijkstra's algorithm].

Then "A" will find out what is the "shortest path" from "A" to every other "node".

Now using the "Global Data," "A" will construct its "Local-Routing Table" :→

"Dest."	"Distance"	"N.H"
"A"	0	A
"B"	2	B
"C"	5	B
"D"	1	D

Using "Dijkstra's algo"

"Convergence" is "fast" compared to "DVR".

Similarly for other "routers" we can construct their "Local Routing Table".

Some Problems with LSR (Link-State-Routing) :→

▷ "Heavy Traffic" due to "flooding".

Every "LSR packet" will have a seq no. to distinguish b/w latest packet & delayed packet

& it also has a "TTL"

"P.T.O"

"Difference b/w "LSR" & "DVR" :->

"DVR" ("Distance Vector Routing")	"LSR" ("Link State Routing")
1> Invented in 1980's	1> Invented in 1990's
2> Bandwidth req. is less "flooding not used"	2> Bandwidth req. is more becoz "flooding is used"
3> Based on "Local-Knowledge" (update "routing tables" only using "local knowledge")	3> Based on "Global-Knowledge" (Each "router" knows about the entire network).
4> Bellman Ford algorithm is used here. But (Not mentioned anywhere)	4> "Dijkstra Algorithm" is used.
5> "Traffic" is very less.	5> "Traffic is very high".
6> Periodic updates are done	6> Periodic updates are done
7> Converges slowly (to obtain final answer)	7> Converges faster
8> Problem of Count to infinity	8> No problem of "Count to infinity" (becoz it is based on global knowledge.)
9> Problem of persistent looping. (loop will remain forever)	9> Transient looping
10> It is implemented using a Protocol called "RIP" ("Routing Information Protocol")	10> It is implemented using OSPF protocol.

[Computation is very simple in "DVR" as compared to "LSR". (because in "LSR" "Dikstra's algo" has to be applied on a "very very big graph")]

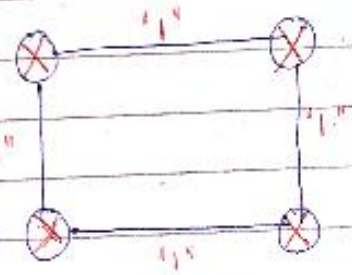
→ RIP ("Routing Information Protocol") ⇒ It is the Implementation of DVR (Distance Vector Routing).

"∞" in "DVR" means that a "host" is "unreachable", but "∞" is not used in practice.

practically we use "16" in place of "∞"

↳ "Metric is" → "Hop-Count".

"edge weight" used is "1"



"∞" is represented by "16"

Rest everything is same as "DVR"

→ "OSPF" ⇒ It is an Implementation of "LSR" ("Link-State-Routing")

"Traffic" is very high in case of "LSR" becoz of "flooding".

— o —

P.T.O

P.T.O