* Unit 3 - Network layer

* Network layer Design issues

→ The network layer is the lowest layer that deals with end-to-end transmission, which is concerned with getting packets from the source all the way to the destination.

→ Following are the design issues of the network layer.

① Store-and-Forward packet switching
② services provided to the transport layer
③ Implementation of connectionless service
④ Implementation of connection oriented service.
⑤ Comparision of virtual-Circuit and Datagram subnets.

* ① Store-and-forward packet switching

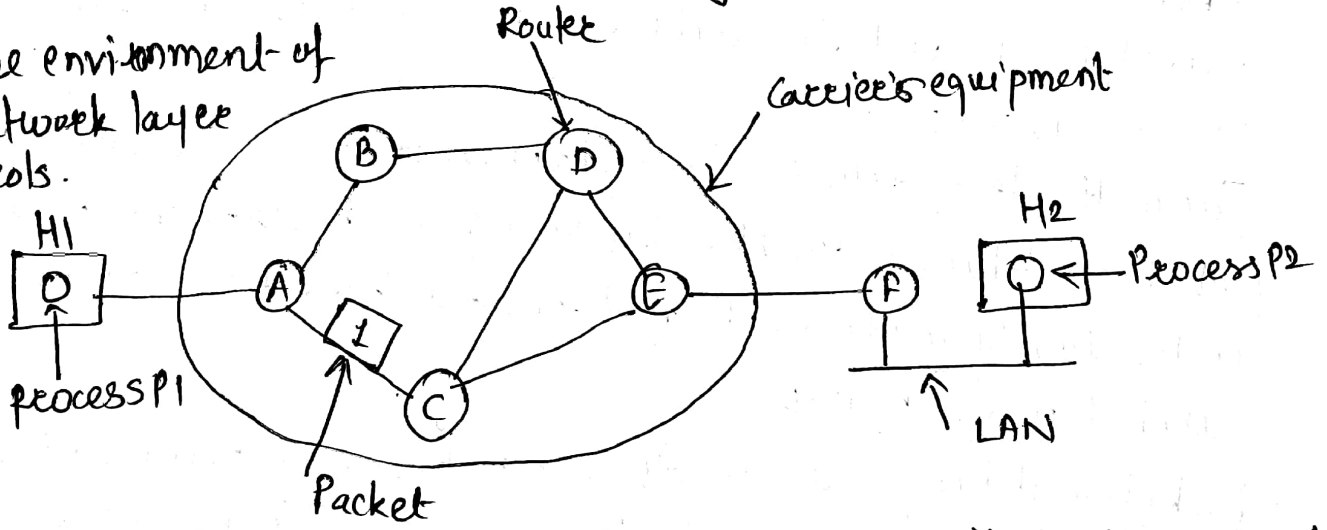


Fig: The environment of the network layer protocols.

→ A host with a packet to send transmits it to the nearest router, either on its own LAN or over point-to-point link to the carrier.

→ The packet is stored there until it has fully arrived so the checksum can be verified.

→ Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered.

→ This mechanism is known as store-and-forward packet switching.

*② Services provided to the transport layer

→ the network layer provides services to the transport layer at the network layer/transport layer interface.

→ the network layer services have been designed with the following goals in mind.

→ 1. the services should be independent of router technology.
2. the transport layer should be shielded from the number, type, and topology of the routers present.
3. the network addresses made available to the transport layer should use a uniform numbering plan, even across LAN's or WAN's.

→ Connection oriented service - is a network communication mode, where a connection is established before any data can be transferred and the data is delivered in the same order as it was sent.

→ Connection-less service - is a data transmission method used in packet-switching networks, by which each data unit is individually addressed and routed based on information carried in each unit.

→ Packet-switching - is a digital networking communication method that groups all transmitted data into suitably sized blocks, called packets.

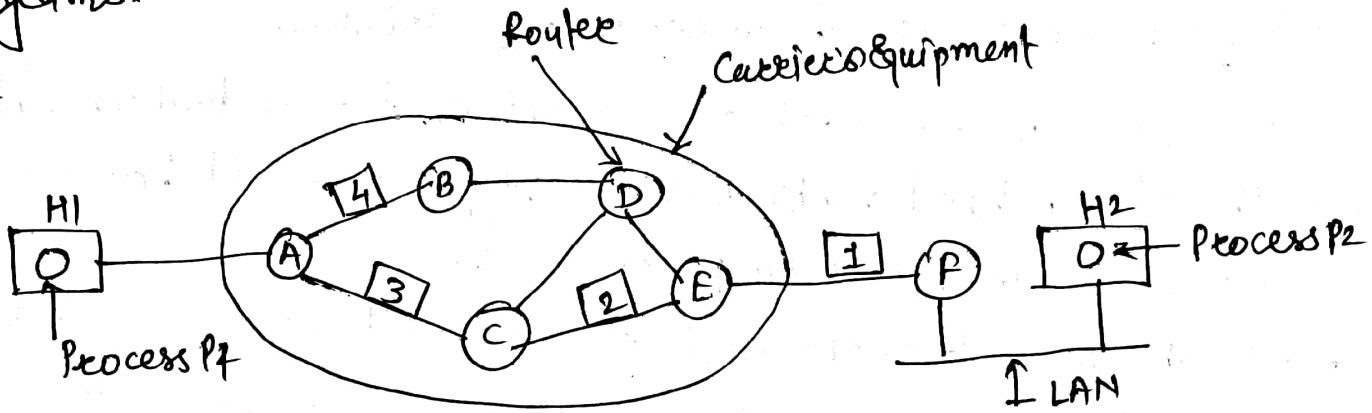
→ A datagram - is a basic transfer unit. the delivery, arrival time, and order of arrival need not be guaranteed by the network.

→ A virtual circuit - is a means of transporting data over a network in such a way that it appears as though there is a dedicated link between source and destination.

③ Implementation of Connectionless service

→ If connectionless service is offered, packets are injected into the network individually and routed independently of each other.

→ No advance set-up is needed. The packets are frequently called datagrams.



→ Suppose that the process P1 has a long message for P2. It hands the message to the transport layer with instructions to deliver it to process P2 on Host H2.

→ the transport layer code runs on H1, adds a transport header to the front of the message and hands the result to the network layer.

A's table ⇒

initially	
A	-
B	B
C	C
D	B
E	C
F	C

later

A	-
B	B
C	C
D	B
E	B
F	B

C's table

A	A
B	A
C	-
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	-
F	F

Fig: Routing within datagram subnet

→ Let us assume, the message is four times longer than maximum size of the packet, so the network layer has to break it into four packets 1, 2, 3, and 4 and sends each of them in turn to router A using some point-to-point protocol.

→ Every router has an internal table telling it where to send packets for each possible destinations.
→ Each table entry is a pair consisting of a destination and the outgoing line to use for that destination.

④ Implementation of Connection oriented service

→ For connection-oriented service, we need a virtual circuit network.
→ The idea behind virtual circuits is to avoid having to choose a new route for every packet sent.
→ With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.
→ Assigns a different connection identifier to the outgoing traffic for the second connection. to avoid any conflicts during transmission.
→ In some contexts, this process is called label switching.

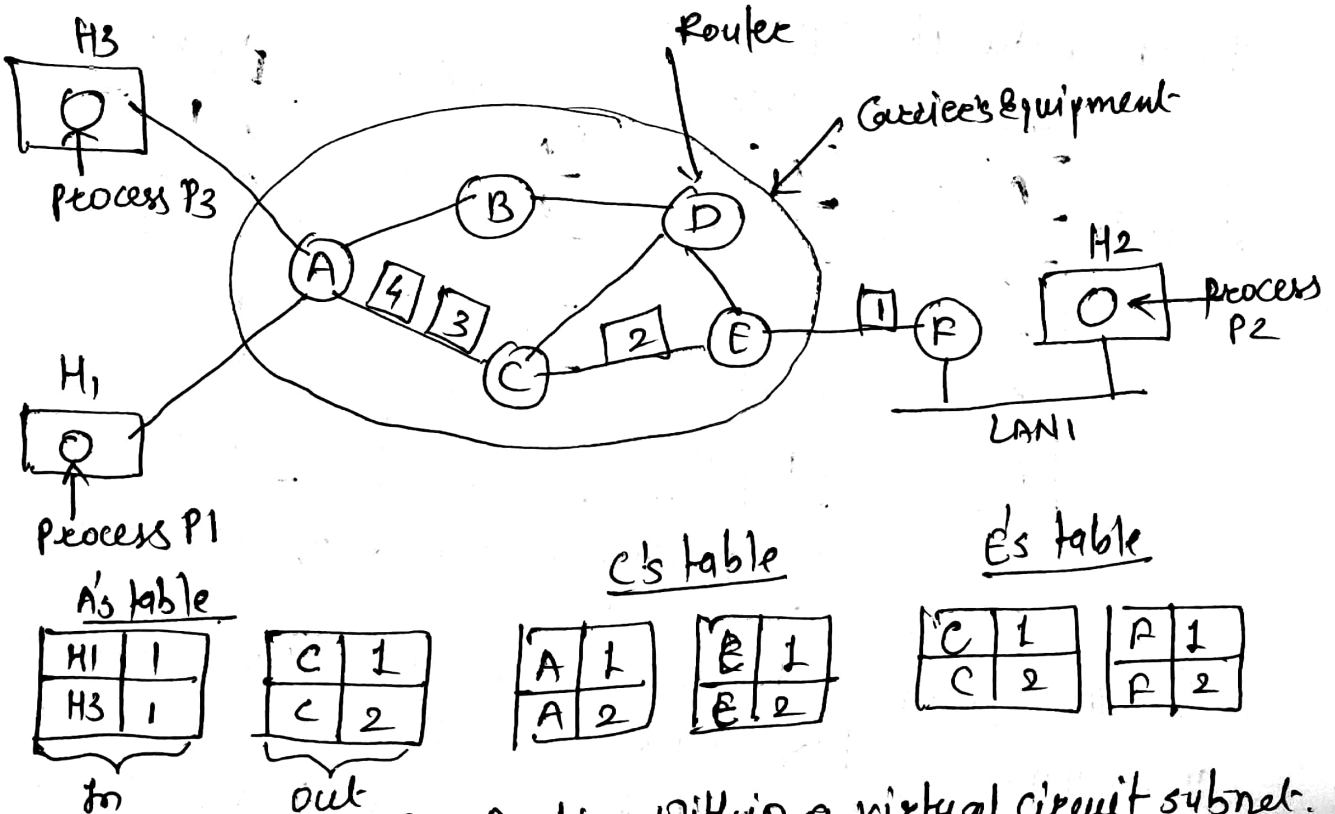


Fig: Routing within a virtual circuit subnet.

(5) Comparison of Virtual-Circuit and Datagram subnets

Issue	Datagram Subnet	Virtual-Circuit Subnet
(1) Circuit setup	Not needed	Required
(2) Addressing	Each packet contains the full source and destination address.	Each packet con a short VC number.
(3) State information	Routers do not hold state information about connections.	Each VC requires router table space per connection.
(4) Routing	Each packet is routed independently.	Route chosen when VC is set up; all packets follow it.
(5) Effect of router failure	None, except for packets lost during the crash.	All VC's that passed through the failed router are terminated.
(6) Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC.
(7) Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC.

* Routing Algorithms

→ the main function of network layer is routing packets from source to destination.

→ the routing algorithm is the part of the network layer software responsible for deciding input and output line.

→ the process of forwarding of a packet is totally depends on routing tables, where the routing algorithm comes into play.

→ Routing algorithms can be grouped into two major classes nonadaptive and adaptive.

→ Nonadaptive algorithms do not depends on routing tables. Instead the choice of the route is computed in advance, off-line and downloaded to the routers when the network is booted. Also called static routing.

→ Adaptive Algorithms, change their routing decisions to reflect changes in the topology and traffic. Also called dynamic routing.

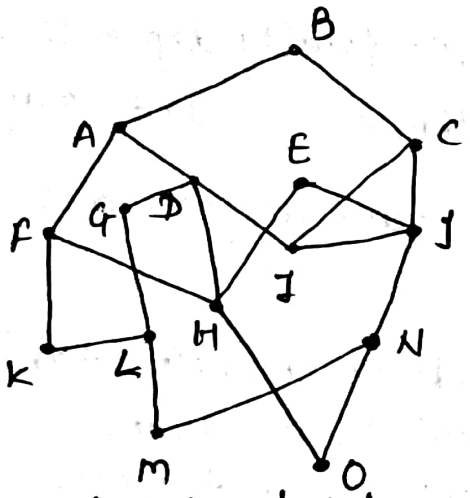
* Optimality Principle

→ Desirable properties of Routing Algorithms.

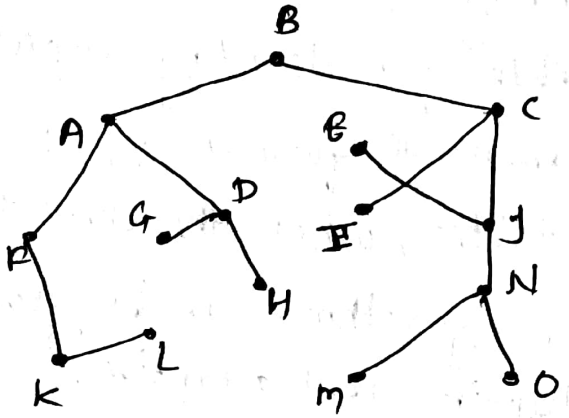
- Correctness – should get packets eventually to the correct destination.
- Simplicity – this usually implies faster
- Robustness – should be able to handle new routers coming online, as well as, handle other going off.
- stability – under constant conditions should converge to some equilibrium.
- Fairness and Optimality – these are hard to satisfy simultaneously.
  - fairness – every node connected to the n/w should get fair chance of transmitting packets
  - Optimality – Optimal in the terms of throughput and minimizing packet delays.

*(1) The Optimality Principle:

→ One can make a general statements about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle.



(a) A subnet



(b) A sink tree for router B

→ It states that, if router J is on the optimal path from router I to router k, then optimal path from J to k also falls along the same route.

→ The set of all optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a sink tree.

→ It does not contain any loops so each packet will be delivered within a finite and bounded number of hops.

② Shortest Path Routing

→ The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line. the algorithm just finds the shortest path between them on the graph.

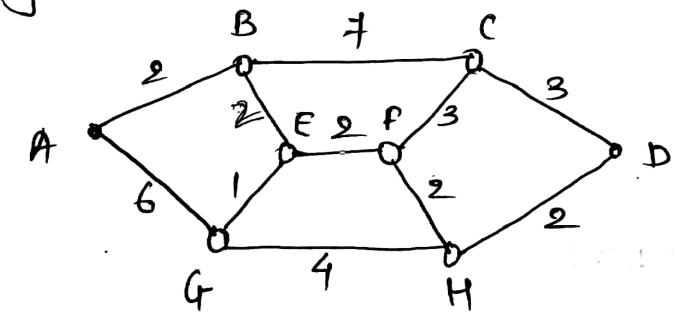
→ the algorithm compute shortest path which is measured according to the cost allocated on each arc of a graph.

→ One of the algorithm to compute shortest path is Dijkstra algorithm in which each node is labeled with its distance from the source node along the best known paths.

→ Node for which no path is known are labeled initially with infinity.

→ A label may be either tentative or permanent.

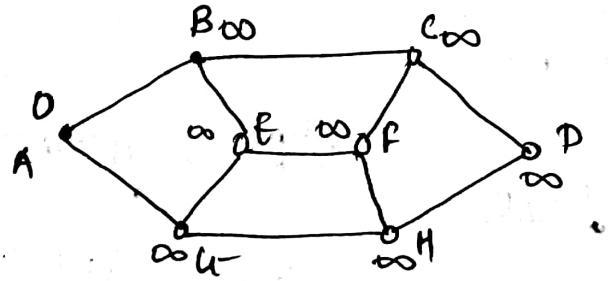
→ Initially, all labels are tentative. when it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.



Source: A Dest: D.

(a)

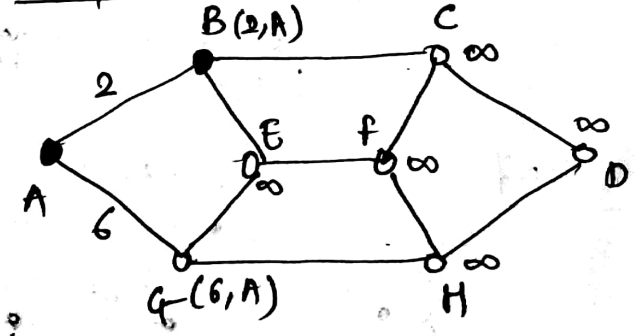
step 1



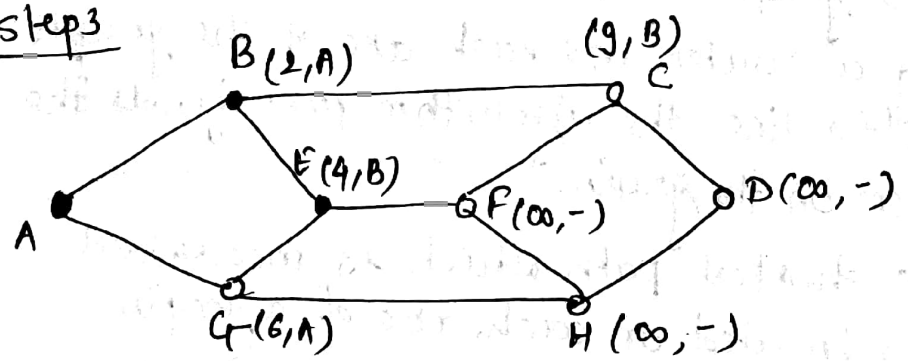
start node = 0

All other nodes = ∞

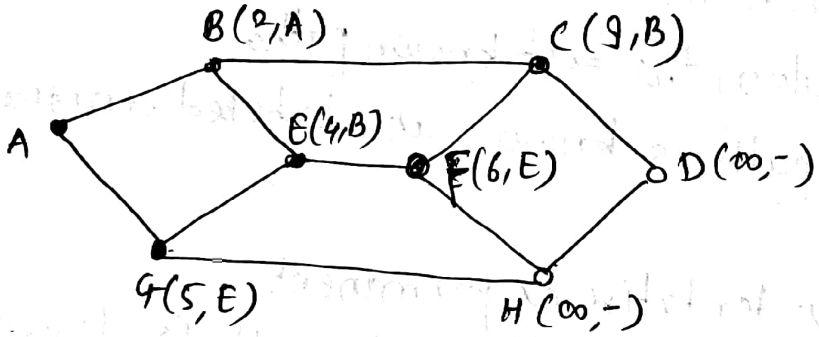
step 2



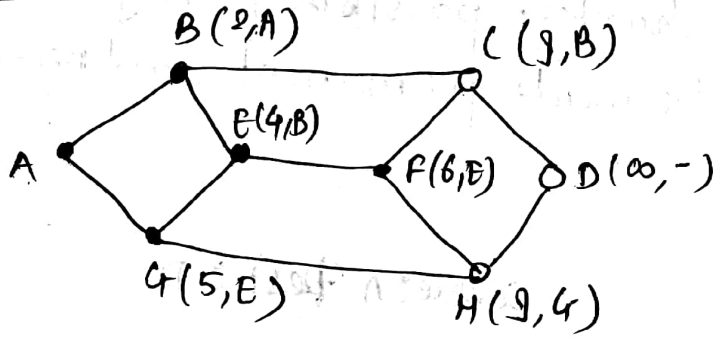
step3



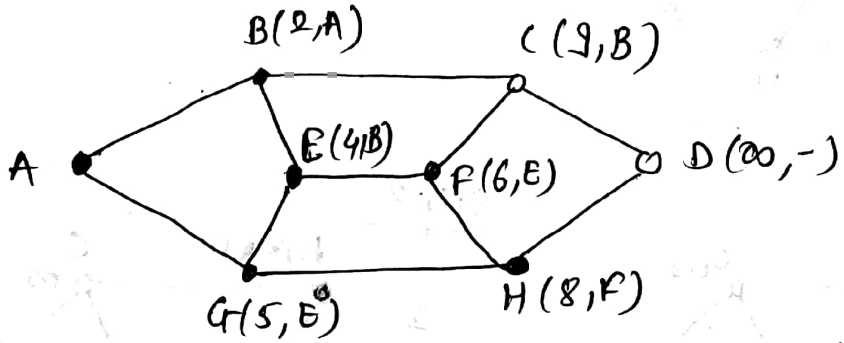
step 4



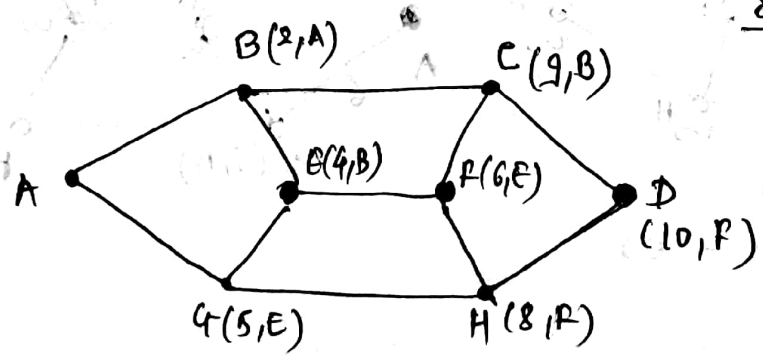
step 5



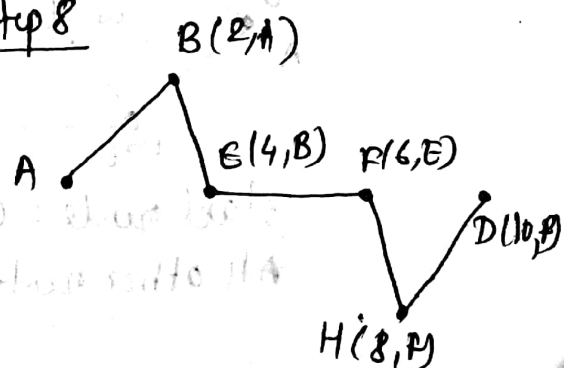
step 6



step 7



step 8



Flooding

* A static algorithm in which every incoming packet is sent-out on every outgoing line except the one it arrived on.
* It generates vast numbers of duplicate packets, to control this some measures are taken to damp the process.

①Hop-Count

→ A hop counter may be contained in the packet header which is decremented at each hop, with the packet being discarded when the counter reaches zero.

→ Ideally, the hop counter should be initialized to the length of the path from source to destination, if the sender does not know how long the path is, it can initialize the counter to the full diameter of the subnet.

* ② An alternative technique for damming the flood is keeping track of the packets which are responsible for flooding using a sequence number. And avoid them out a second time.
* Selective-flooding is an variation of flooding in which the routers do not send every incoming packet out on every line; only on those lines that are going approximately in the right direction of the destination.

* Advantages
  → Highly Robust
  → Works on Virtual Circuit
  → Always chooses shortest path
  → Broadcast messages to all the nodes.

* Applications
  → military applications.
  → distributed database applications
  → Wireless networks

* Distance Vector Routing

→ Distance vector routing is an dynamic routing protoc

→ Distance vector routing algorithms operates by having each router maintain a table giving the best known distance to each destination and which line to use to get there.

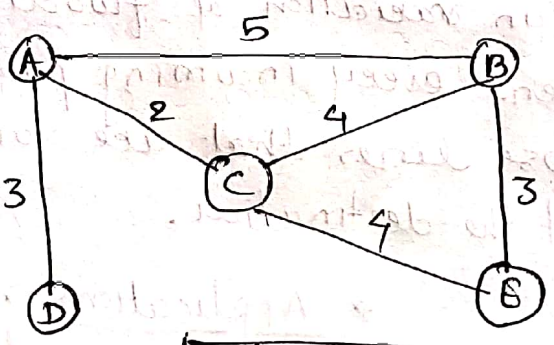
→ These tables are updated by exchanging information with the neighbours.

→ The distance vector routing algorithm is called as distributed Bellman-Ford routing algorithm.

→ The metric used might be the number of hops, time delay in milliseconds, bandwidth etc.

→ The router is assumed to know the "distance" to each of its neighbours.

→ Consider the following example with five routers connected to each other.



Dest	Cost	Next
A	0	–
B	5	–
C	2	–
D	3	–
E	∞	–

A's table

Dest	Cost	Next
A	5	–
B	0	–
C	4	–
D	∞	–
E	3	–

B's table

Dest	Cost	Next
A	3	–
B	∞	–
C	∞	–
D	0	–
E	∞	–

D's table

Dest	Cost	Next
A	2	–
B	4	–
C	0	–
D	∞	–
E	4	–

C's table

Dest	Cost	Next
A	∞	
B	3	B
C	4	C
D	∞	
E	0	–

E's table

Fig: Initialization of tables in DVR

→ At the beginning, each node know the cost of itself and its immediate neighbor. ⑦

→ the distance of any entry that is not a neighbor is marked as infinite.

Sharing → the idea is to share the information between neighbors.

→ the node A does not know the distance about E, but node C does, if node C does share its Routing table with A, node A can also know how to reach node E.

→ while sharing, a node can send only the first two column of its table to any neighbor.

updating → when node receives a two-column table from a neighbor, it needs to update its own routing table.

→ updating takes three steps.

① the receiving node needs to add the cost between itself and the sending node to each value in the second column.

② the receiving node needs to add the name of sending node to each row as the third column.

③ the modified table entries are compared with old table entries and the entries with smallest cost kept in table.

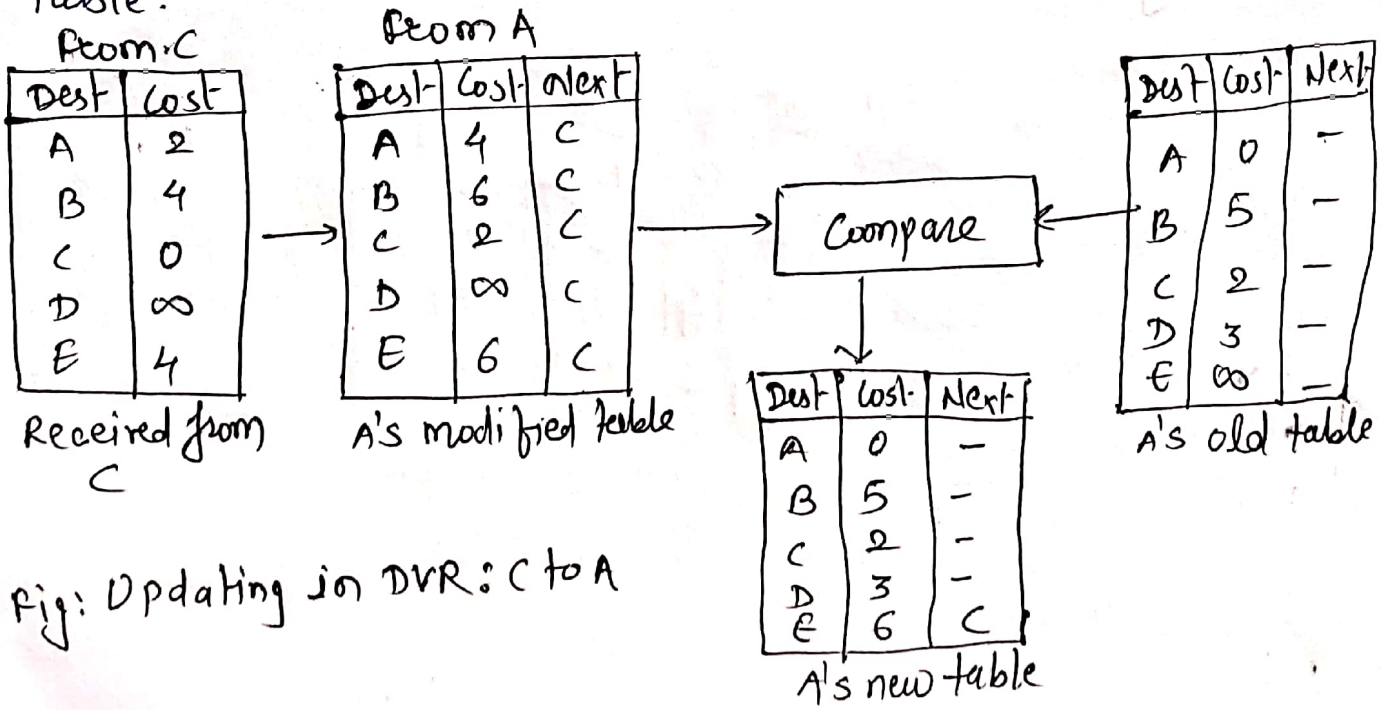
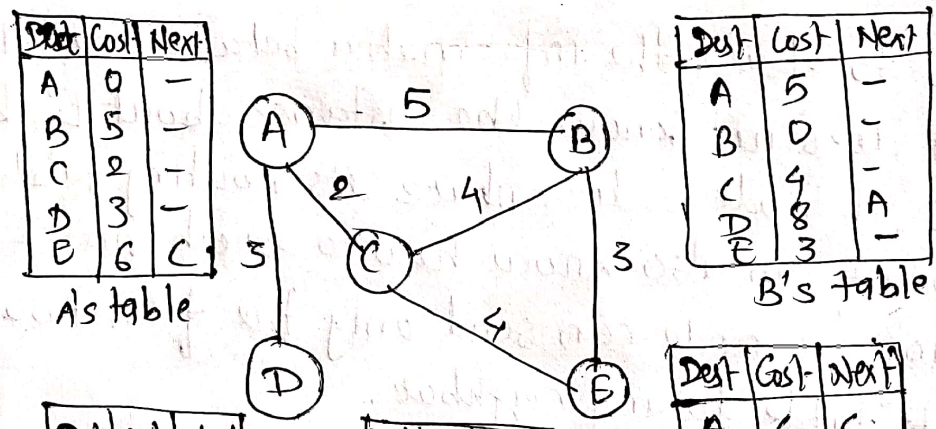


Fig: Updating in DVR: C to A

→ The sharing of tables may be periodic which updates every 30s or it may be triggered update which happens if there is any change in any routing table.



A's table

Dest	Cost	Next
A	0	–
B	5	–
C	2	–
D	3	–
E	6	C

B's table

Dest	Cost	Next
A	5	–
B	0	–
C	4	–
D	8	A
E	3	–

D's table

Dest	Cost	Next
A	3	–
B	8	A
C	5	A
D	0	–
E	9	A

C's table

Dest	Cost	Next
A	2	–
B	4	–
C	0	–
D	5	A
E	4	–

E's table

Dest	Cost	Next
A	6	C
B	3	–
C	4	–
D	9	C
E	0	–

fig: Final Distance vector Routing tables.

→ Disadvantage

→ Tedious comparing/updating process

→ slow response to infinite loop problem

→ huge list to be maintained.

	1	2	3	4	5
1	0	5	2	3	99
2	5	0	4	99	3
3	2	4	0	99	4
4	3	99	99	0	99
5	99	3	4	99	0

* Count to infinity problem

→ One of the important issue in Distance Vector Routing is Count to infinity problem.

→ Count to infinity is just another name for a routing loop.

→ In DVR, routing loops usually occur when an interface goes down.

→ Consider the following example:

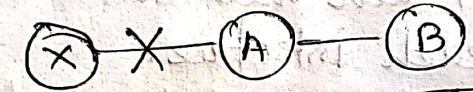
① Before failure

X	2	-

X	6	A

⇒ Both nodes A and B know how to reach node X.

② After failure

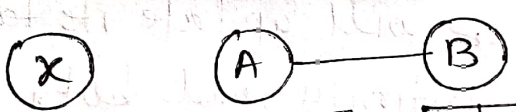


X	∞	-

X	6	A

⇒ After failure, system becomes unstable

⑤ After A receives update from B

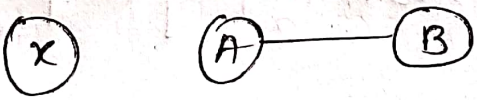


X	10	B

X	6	A

⇒ Node A, receives update and assuming that B has found a way to X, immediately update its routing table.

④ After B receives update from A

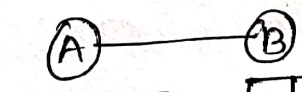


X	10	B

X	14	A

⇒ Node B, receives update from A, and thinks something has changed and immediately updates its routing table.

⑤ Finally - X



X	∞	-

X	∞	-

⇒ The cost of reaching X increases gradually until it reaches infinity.

⇒ Packet bounce between A and B, creating loop problem, known as count to infinity problem.

⇒ Solutions

→ ① Defining infinity

⇒ The first obvious solution is to redefine infinity to a smaller number.

⇒ most implementation of DVR define the distance between each node to be 1 and 16 as an infinity

⇒ That means the size of the network, in each direction can not exceed 15 hops.

→ ② Split Horizon

⇒ In this strategy instead of flooding the table through each interface, each node sends only part of its table through each interface.

⇒ In given example in the first update itself if A shares the cost to reach x as infinity without sharing the third column, so he will update its table cost to reach to x as infinity and both node A and B known that x is not reachable and will be prevented from looping the packet.

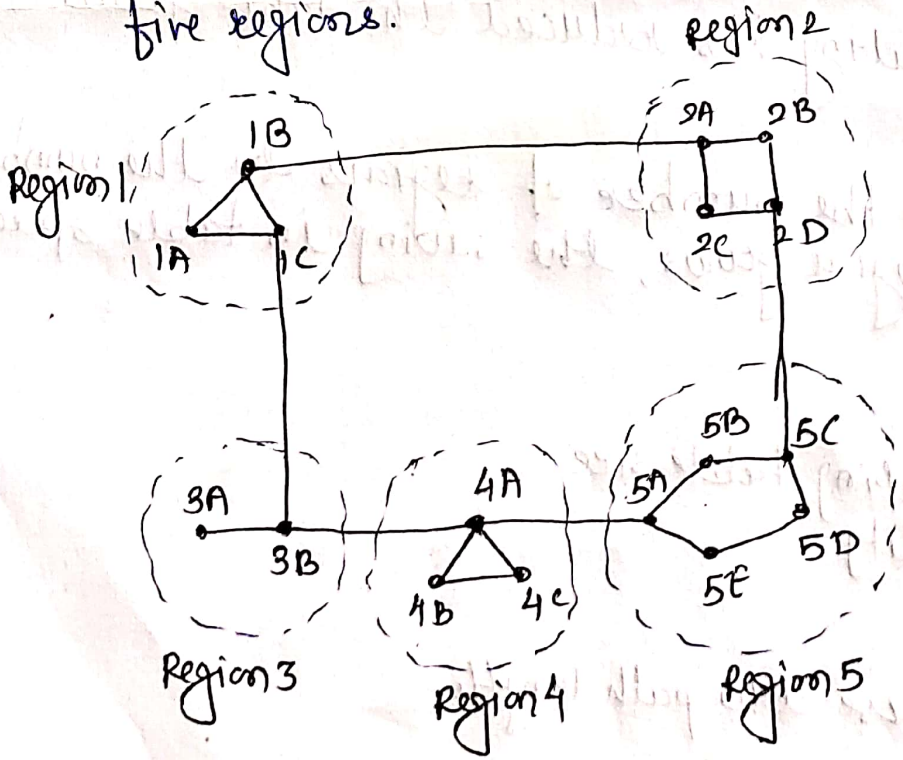
Hierarchical Routing

→ As network grow in size, the routing tables grow proportionally. At a certain point the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically.

→ In Hierarchical routing, the routers are divided into regions, with each router knowing all the details about how to route a packet to destinations within its own region. but knowing nothing about the internal structure of other regions.

→ Hierarchical routing is a multi-level hierarchical routing protocol that employs clustering at different levels. A leader is elected at every level.

→ Example:
The following example is a two level hierarchy with five regions.



(a) Hierarchical Regions

Table of A

Dest	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b) Full table of A without Hierarchical Routing

→ The full routing table of router 1A has 17 entries.
→ When routing is done hierarchically, there are entries for all the local routers as before, but all other regions have been condensed into single router.
→ So according to example all traffic for region 2 goes via 1B-2A line and rest of the remote traffic goes via 1C-3B line.

→

Dest	line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c) Hierarchical table for 1A.

→ Hierarchical Routing has reduced the table from 17 to 7 entries.
→ As the ratio of the number of regions to the number of routers per region grows, the saving in table space increase.
→ Advantages
* Reduction in Routing table size.
* Better scalability
→ Disadvantage
* It may increase the path length.

* Congestion Control and Admission Control Algorithms

* Leaky Bucket Algorithm

→ Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. Once the bucket is full any additional water entering it spills over the sides and is lost. The same idea is applied to packets.

→ Conceptually each network interface contains a leaky bucket.

→ When the host has to send a packet, the packet is thrown into the bucket, the bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.

→ Bursty traffic is converted to a uniform traffic by the leaky bucket, the bucket is a finite queue that outputs at a finite rate.

→ If there is room in the queue it is queued up and if there is no room then the packet is discarded.

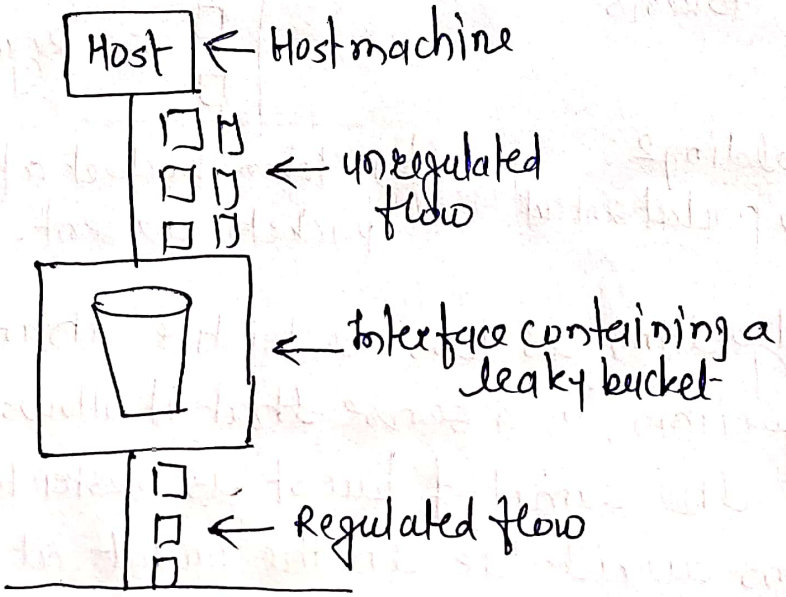


Fig: Leaky Bucket Implementation.

* Token Bucket Algorithm

→ The leaky bucket algorithm, enforces a rigid pattern at the output stream, irrespective of the pattern of the input.

→ For many applications it is better to allow the output to speed up somewhat when a larger burst arrives than to loose the data.

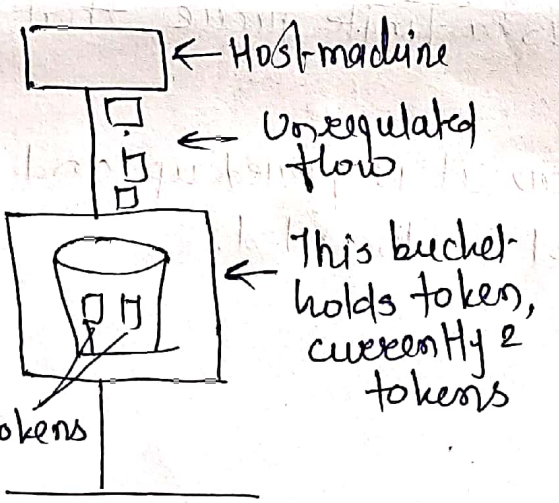
→ Token bucket will provide the solution for this, in which the leaky bucket holds token, generated at regular intervals.

→ In regular intervals token are thrown into the bucket.

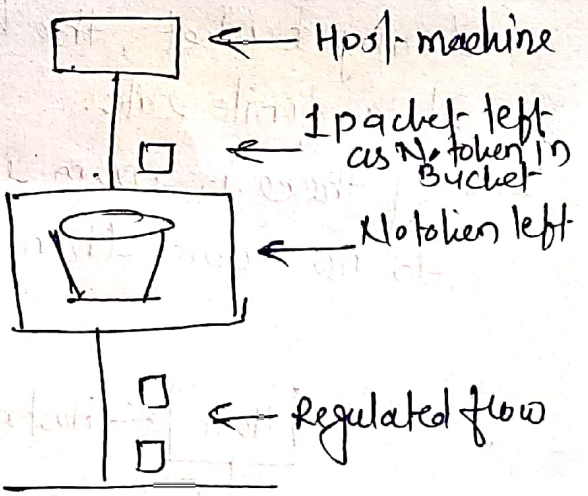
→ The bucket has maximum capacity.

→ If there is a ready packet, a token is removed from the bucket, and the packet is sent.

→ If there is no token in the bucket, the packet cannot be send.



(a) Token Bucket holding 2 tokens, before packet sent out.



(b) Token Bucket after two packets are sent.

→ Token Bucket algorithm is less restrictive than the leaky bucket algorithm, in a sense that it allows bursty traffic, however the limit of burst is restricted by the number of tokens available in the bucket at a particular instant of time.

→ The implementation of basic token bucket algorithm is very simple, a variable is used just to count the tokens. This counter is incremented every t seconds and is decremented whenever a packet is sent. whenever this counter reaches zero, no further packet is sent.

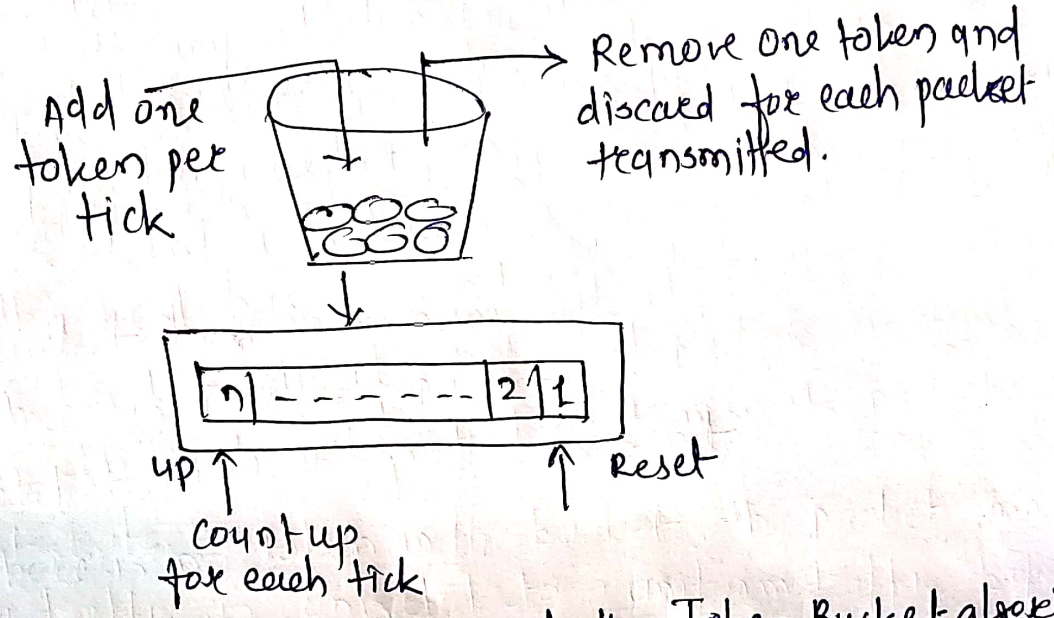


Fig: Implementation of the Token Bucket algorithm.