UNIT IV ROUTING 7

Routing and protocols: Unicast routing - Distance Vector Routing - RIP - Link State Routing – OSPF – Path-vector routing - BGP - Multicast Routing: DVMRP – PIM.

4.1Routing and protocols

- The network layer is responsible for routing the packet from its source to the destination.
- The network layer is responsible for finding the best one among these possible routes.
- The network layer needs to have some specific strategies for defining the best route.
- Routing is the concept of applying strategies and running routing protocols to create the decision-making tables for each router.
- These tables are called as routing tables

4.2UNICAST ROUTING

- Routing is the process of selecting best paths in a network.
- In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.
- Routing a packet from its source to its destination means routing the packet from a *source router* (the default router of the source host) to a *destination router* (the router connected to the destination network).
- The source host needs no forwarding table because it delivers its packet to the default router in its local network.
- The destination host needs no forwarding table either because it receives the packet from its default router in its local network.
- Only the intermediate routers in the networks need forwarding tables.

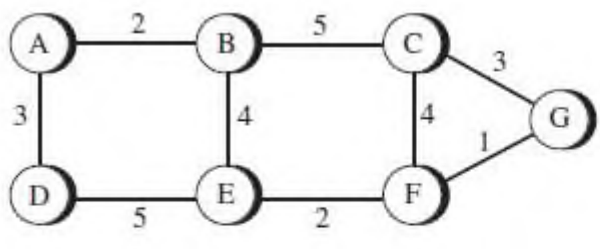
NETWORK AS A GRAPH

➢The Figure below shows a graph representing a network.

➢The nodes of the graph, labeled A through G, may be hosts, switches, routers, or networks.

➢The edges of the graph correspond to the network links.

➢Each edge has an associated *cost*



UNICAST ROUTING ALGORITHMS

➢There are three main classes of routing protocols:

1) Distance Vector Routing Algorithm – Routing Information Protocol

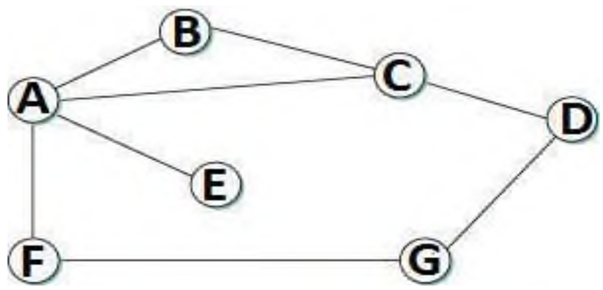
2) Link State Routing Algorithm – Open Shortest Path First Protocol

3) Path-Vector Routing Algorithm - Border Gateway Protocol

4.3 DISTANCE VECTOR ROUTING (DSR) & ROUTING INFORMATION PROTOCOL (RIP)

➢Distance vector routing is *distributed*, i.e., algorithm is run on all nodes.
➢Each node *knows* the distance (cost) to each of its directly connected neighbors.
➢Nodes construct a *vector* (Destination, Cost, NextHop) and distributes to its neighbors.
➢Nodes compute routing table of *minimum* distance to every other node via NextHop using information obtained from its neighbors.

Initial State



➢In given network, *cost* of each link is 1 hop.
➢Each node sets a distance of 1 (hop) to its *immediate* neighbor and cost to itself as 0.
➢Distance for non-neighbors is marked as *unreachable* with value ∞ (infinity).
➢For node *A*, nodes *B*, *C*, *E* and *F* are *reachable*, whereas nodes *D* and *G* are *unreachable*.

Destination	Cost	NextHop
A	0	A
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Node A's initial table

Destination	Cost	NextHop
A	1	A
B	1	B
C	0	C
D	1	D
E	∞	—
F	∞	—
G	∞	—

Node C's initial table

Destination	Cost	NextHop
A	1	A
B	∞	—
C	∞	—
D	∞	—
E	∞	—
F	0	F
G	1	G

Node F's initial table

➢The initial table for all the nodes are given below

Initial Distances Stored at Each Node (Global View)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

- ✓ Each node *sends* its initial table (distance vector) to neighbors and receives their estimate.
- ✓ Node *A* sends its table to nodes *B*, *C*, *E* & *F* and receives tables from nodes *B*, *C*, *E* & *F*.
- ✓ Each node *updates* its routing table by comparing with each of its neighbor's table For each destination, Total Cost is computed as:
- ✓ ***Total Cost*** = Cost (*Node* to *Neighbor*) + Cost (*Neighbor* to *Destination*)
- ✓ If Total Cost < Cost then
- ✓ ***Cost*** = Total Cost and NextHop = *Neighbor*

- ✓ Node *A learns* from *C*'s table to reach node *D* and from *F*'s table to reach node *G*.
  Total Cost to reach node *D* via *C* = Cost (*A* to *C*) + Cost(*C* to *D*)
  Cost = 1 + 1 = 2.
- ✓ Since 2 < ∞, entry for destination *D* in *A*'s table is changed to (*D*, 2, *C*)
- ✓ Total Cost to reach node *G* via *F* = Cost(*A* to *F*) + Cost(*F* to *G*) = 1 + 1 = 2
- ✓ Since 2 < ∞, entry for destination *G* in *A*'s table is changed to (*G*, 2, *F*)
- ✓ Each node builds *complete* routing table after few exchanges amongst its neighbors.

Node A's final routing table

Destination	Cost	NextHop
A	0	A
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

➢System stabilizes when all nodes have complete routing information, i.e., **convergence.**

➢Routing tables are exchanged *periodically or* in case of *triggered update*

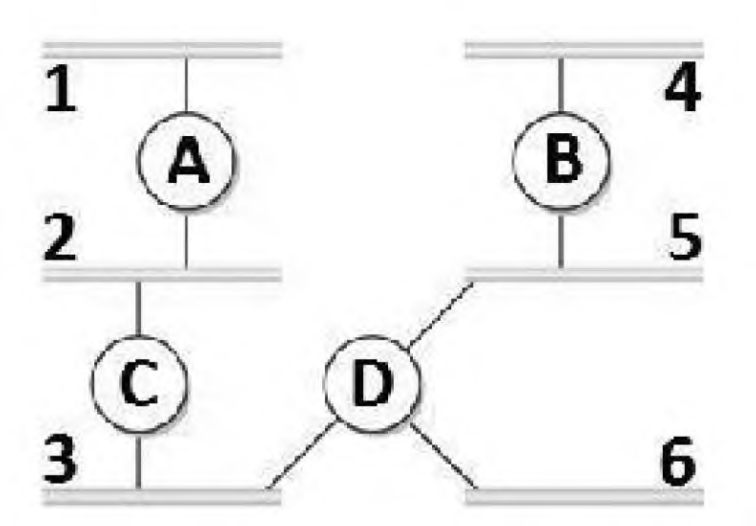
Final Distances Stored at Each Node (Global View)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

ROUTING INFORMATION PROTOCOL (RIP)

- RIP is an intra-domain routing protocol based on distance-vector algorithm.

Example



- Routers *advertise* the cost of reaching networks. Cost of reaching each link is 1 hop. For example, router *C* advertises to *A* that it can reach network 2, *3* at cost 0 (directly connected), networks *5*, *6* at cost 1 and network *4* at cost 2.
- Each router *updates* cost and next hop for each network number.
- Infinity is defined as 16, i.e., any route cannot have more than 15 hops. Therefore RIP can be implemented on small-sized networks only.
- Advertisements are sent every 30 seconds or in case of triggered update
- ***Command*** **-** It indicates the packet type. Value 1 represents a request packet. Value 2 represents a response packet.
- ***Version*** **-** It indicates the RIP version number. For RIPv1, the value is 0x01.
- ***Address Family Identifier*** **-** When the value is 2, it represents the IP protocol.
- ***IP Address*** **-** It indicates the destination IP address of the route. It can be the addresses of only the natural network segment.
- ***Metric*** **-** It indicates the hop count of a route to its destination

0	7	15	31
command	version	must be zero	
address family identifier		must be zero	
IP address			
must be zero			
must be zero			
metric			

Count-To-Infinity (or) Loop Instability Problem

- Suppose link from node *A* to *E* goes *down*.
- Node *A* advertises a distance of ∞ to *E* to its neighbors
- Node B receives periodic update from C before A's update reaches B
- Node *B* updated by *C*, concludes that *E* can be reached in 3 hops via *C*
- Node *B* advertises to *A* as 3 hops to reach
- Node *A* in turn updates *C* with a distance of 4 hops to *E* and so on
- Thus nodes update each other until cost to *E* reaches *infinity*, i.e., *no convergence*.
- Routing table does not stabilize.
- This problem is called *loop instability* or *count to infinity*