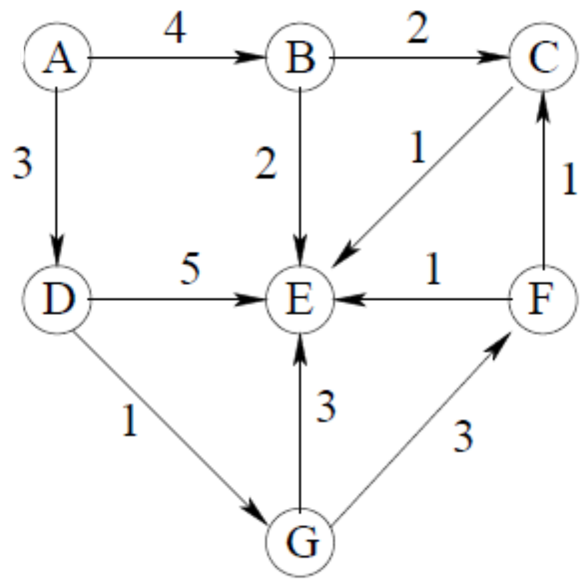
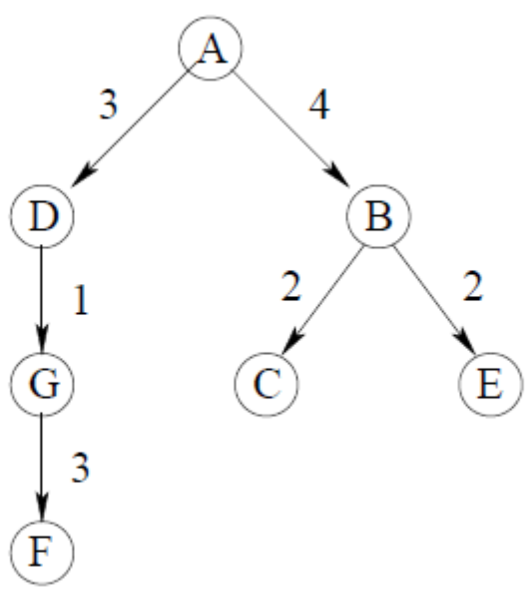
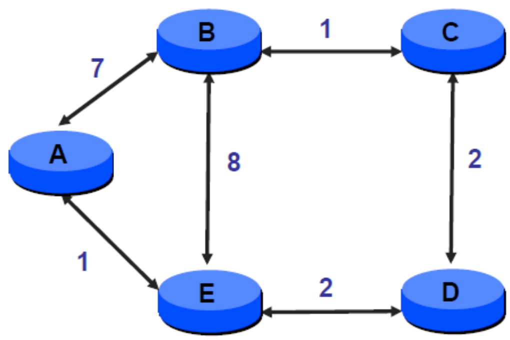
1. Find the shortest path from node A to every other router in the network. Illustrate the SPT using Dijkstra algorithm (Show them the steps (Table) similar to chart #27 of the Routing Algorithm charts posted)



Answer:

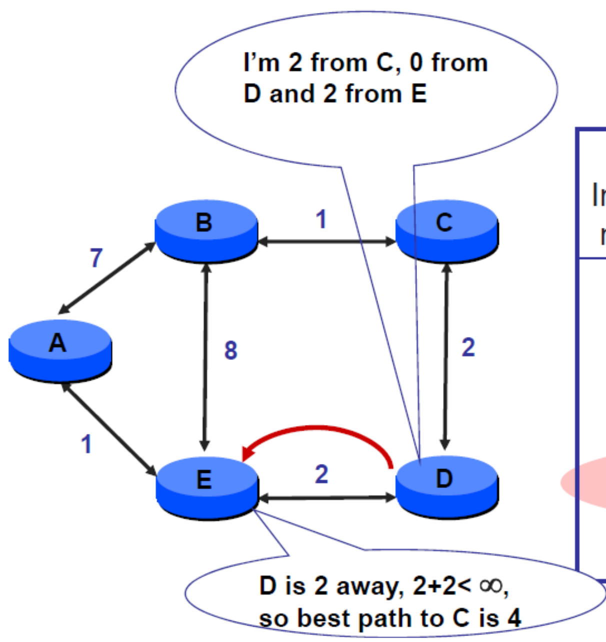


2. Apply the Bellman ford algorithm (Distance Vector) to the following network.



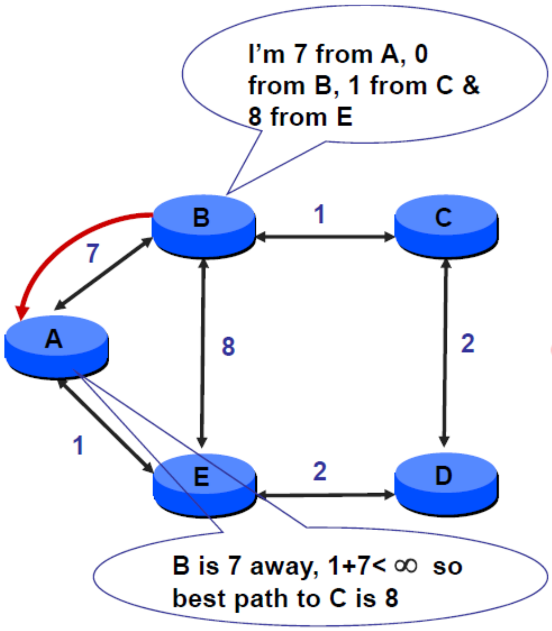
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	∞	2	0

Example of iteration: Router D send his DV to E



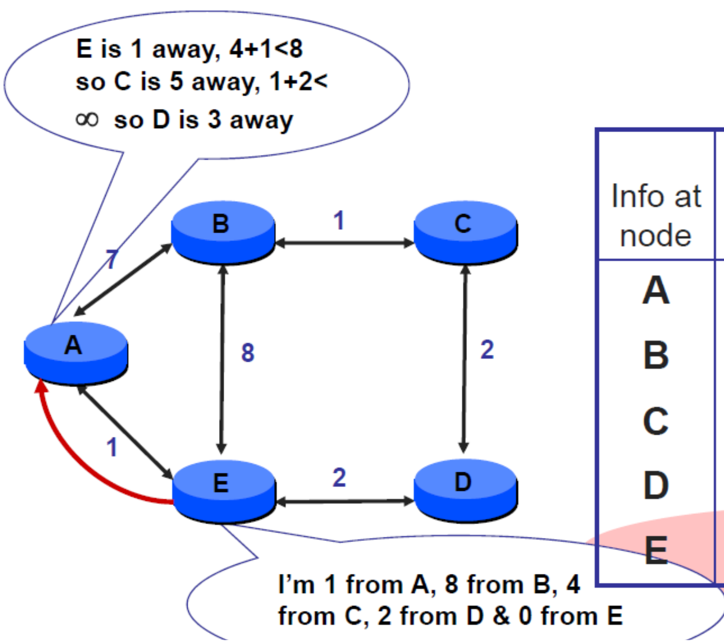
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	**4**	2	0

Example of iteration: Router B sends his DV to A



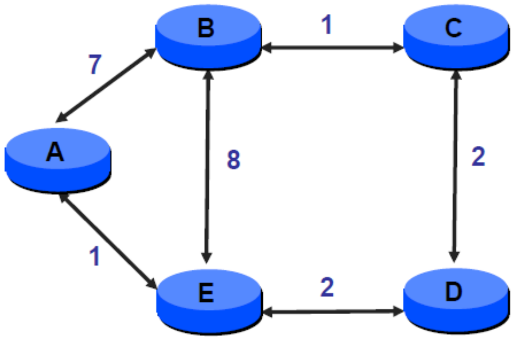
Info at node	Distance to Node A	B	C	D	E
A	0	7	8	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

Example of iteration: Router E sends his updated DV (i.e. after the first iteration) to A



Info at node	Distance to Node A	B	C	D	E
A	0	7	5	3	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

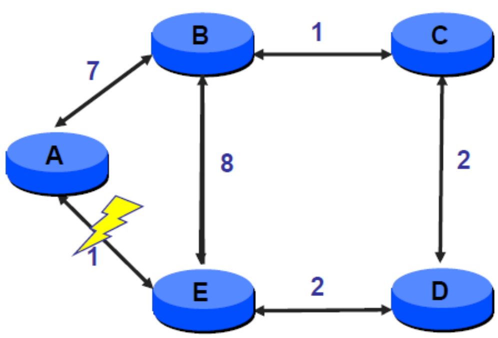
The final DV is table is shown below



Info at node	Distance to Node A	B	C	D	E
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

If link between A and E fails

- A marks distance to E as ∞ , and tells B
- E marks distance to A as ∞ , and tells B and D
- B and D recompute routes and tell C, E and E
- etc… until converge



Info at node	Distance to Node A	B	C	D	E
A	0	7	8	10	12
B	7	0	1	3	5
C	8	1	0	2	4
D	10	3	2	0	2
E	12	5	4	2	0

Distance-Vector Routing Protocol

- One of the two main classes of routing protocols used in the computer networks (e.g. RIP).
- Uses the *Bellman-Ford algorithm* to calculate paths
- A distance-vector routing protocol requires that
  - A router informs its neighbors of topology changes
    - Periodically
    - Whenever a change is detected in the topology of a network
  - Unlike link-state protocols doesn't require the router to inform all the nodes in a network of topology changes
    - Hence less computational complexity and message overhead

Bellman-Ford Algorithm

- A distributed version of Bellman–Ford algorithm is used in distance-vector routing protocols, (e.g. RIP).
  - Each node (i.e. router) calculates the distances between itself and all other nodes and stores this information as a table.
  - Each node sends its table to all neighboring nodes.
  - When a node receives distance tables from its neighbors, it calculates the shortest routes to all other nodes and updates its own table to reflect any changes.
- Disadvantages of the Bellman–Ford algorithm in this context
  - Scalability
  - Slow convergence
  - Count-to-infinity problem
    - If failure of a link or a node renders a node unreachable from some other nodes, those nodes may indefinitely and gradually increase their cost estimates of the distance to the unreachable node, and routing loops may also be formed.

Count-to-Infinity Problem

- Consider an example topology A—B—C—D—E (hop-count is the metric)
  - A goes down.
  - B does not receive the vector update from A so it concludes that its route of cost 1 to A is no longer available.
  - C doesn't know yet that A is down and tells B that A is 2 hops away form it
  - The wrong info propagates until it reaches infinity.
  - The algorithm then corrects itself using the "Relax property" of Bellman Ford.

- **Partial Solutions**
  - ***Split Horizon:*** prohibits a router from advertising a route back out the interface from which it was learned
  - ***Split Horizon with poison reverse:*** Allows a router to advertise the route back to the router that is used to reach the destination, but marks the advertisement as unreachable.
  - More theoretical than practical, allows more scalable and complex DV protocols

Problem #2: How the *Distance-Vector* Algorithm Builds the Routing Table for Node z

- Consider the network shown below and assume that each node initially knows the costs to each of its neighbors. Consider the distance vector algorithm and show the distance tables entries at node z.

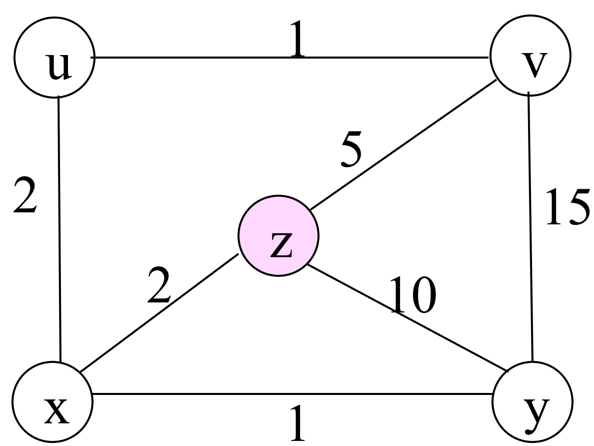
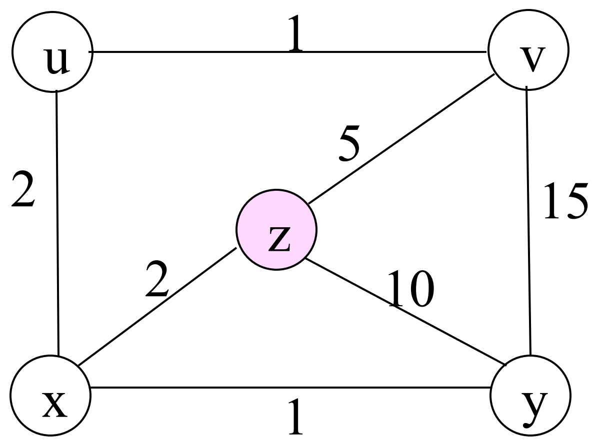


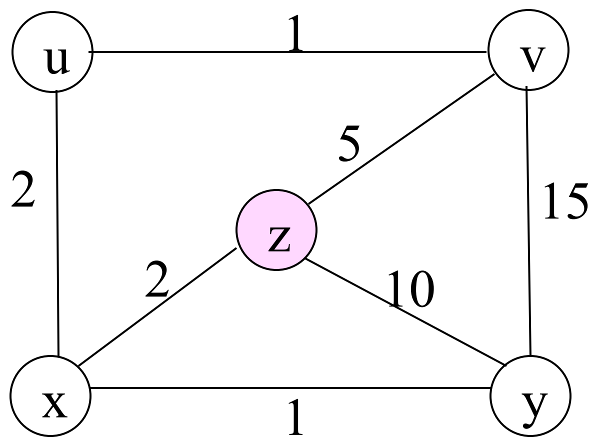
Table at Node Z: Step 1



Cost to

		u	v	x	y	z
	v	∞	∞	∞	∞	∞
From	x	∞	∞	∞	∞	∞
	y	∞	∞	∞	∞	∞
	z	∞	5	2	10	0

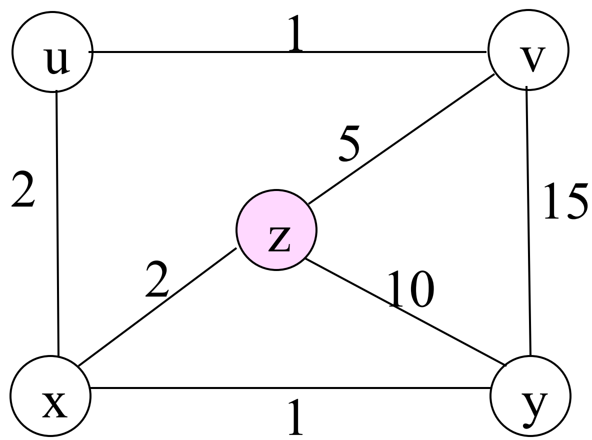
Table at Node Z: Step 2



Cost to

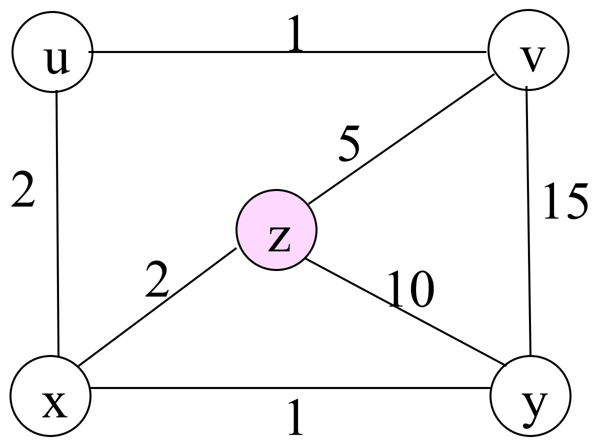
From		u	v	x	y	z
	v	1	0	∞	15	5
From	x	2	∞	0	1	2
	y	∞	15	1	0	10
	z	4	5	2	3	0

Table at Node Z: Step 3



		Cost to				
		u	v	x	y	z
	v	1	0	3	15	5
From	x	2	3	0	1	2
	y	3	15	1	0	3
	z	4	5	2	3	0

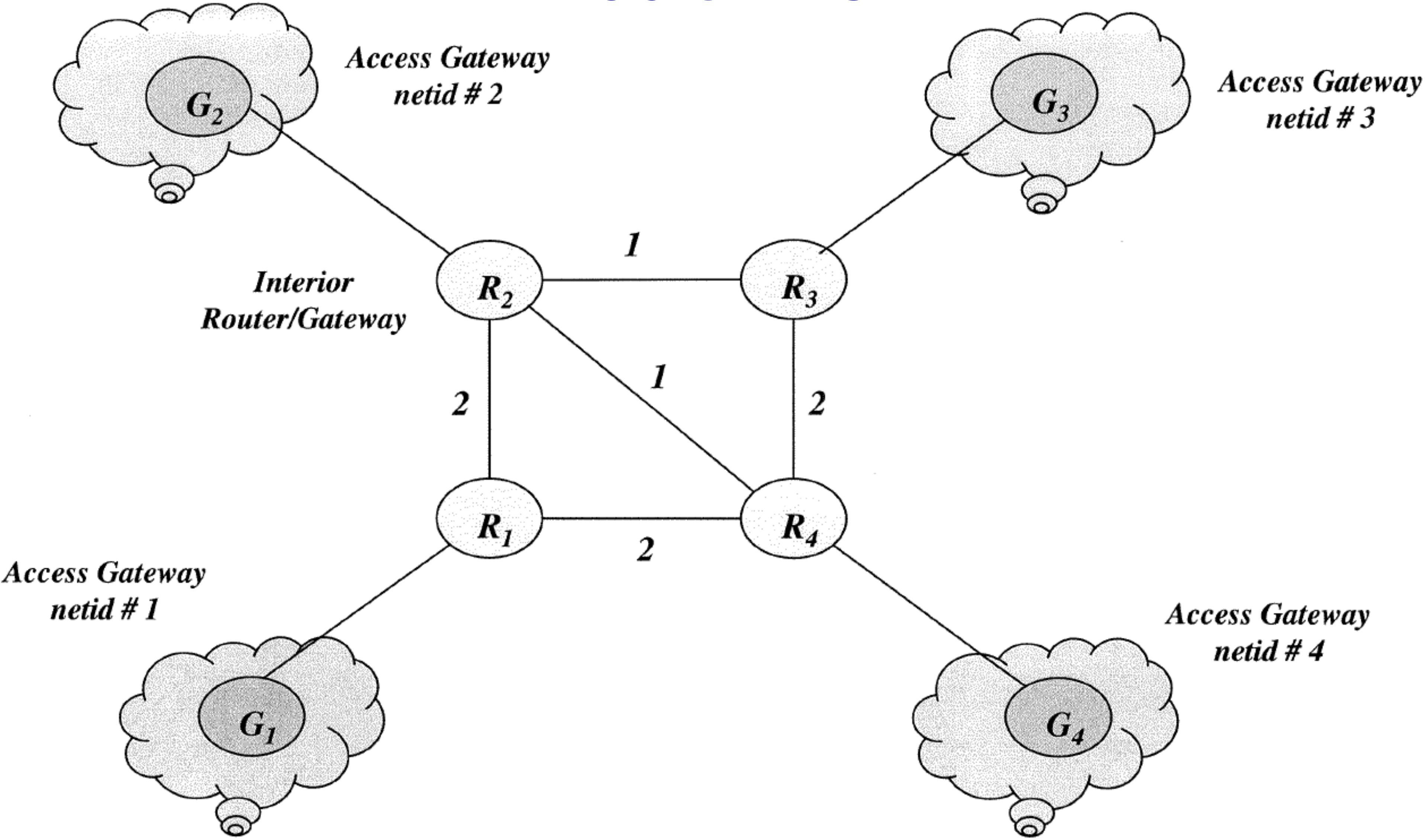
Table at Node Z: Step 4

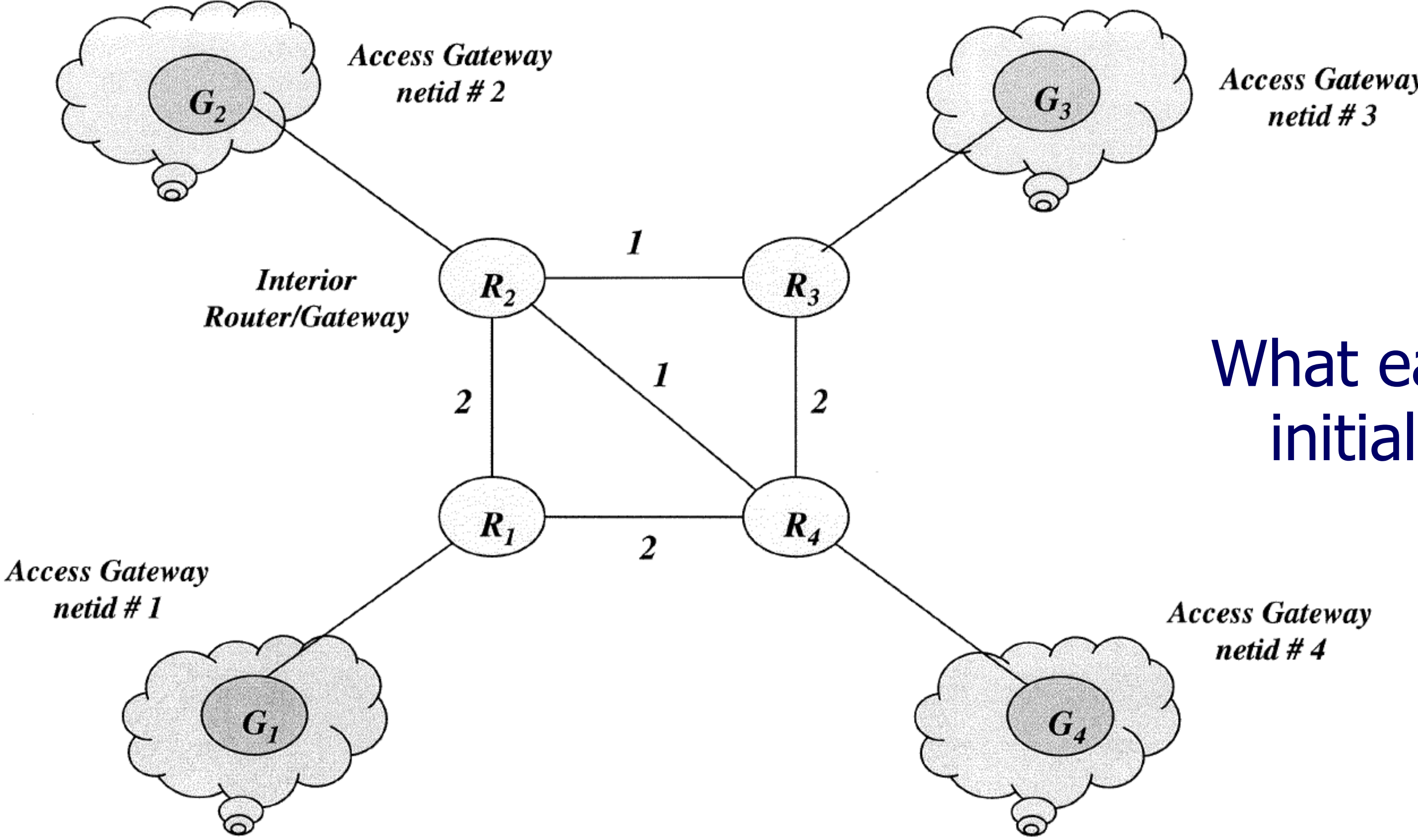


		Cost to				
		u	v	x	y	z
	v	1	0	3	4	5
From	x	2	3	0	1	2
	y	3	4	1	0	3
	z	4	5	2	3	0

Example of a DV Routing

Problem#3





What each router initially knows

Info at	(Next Hop, Distance to)			
	netid1	netid2	netid3	netid4
R1	R1,0	∞	∞	∞
R2	∞	R2,0	∞	∞
R3	∞	∞	R3,0	∞
R4	∞	∞	∞	R4,0

Initial/Intermediate/Final Routing Tables

@ R_1

Netid	Next Hop, D
1	R_1, 0

R_2 R_4

Netid	Next Hop, D
1	R_1, 0
2	R_2 , 2
4	R_4 , 2

R_2 R_4

Netid	Next Hop, D
1	R_1, 0
2	R_2 , 2
3	R_2 , 3
4	R_4, 2

@ R_2

Netid	Next Hop, D
2	R_2 , 0

R_1 R_3 R_4

Netid	Next Hop, D
1	R_1 , 2
2	R_2 , 0
3	R_3 , 1
4	R_4 , 1

R_1 R_3 R_4

Netid	Next Hop, D
1	R_1 , 2
2	R_2 , 0
3	R_3 , 1
4	R_4 , 1

@ R_3

Netid	Next Hop, D
3	R_3 , 0

R_2 R_4

Netid	Next Hop, D
2	R_2 , 1
3	R_3 , 0
4	R_4 , 2

R_2 R_4

Netid	Next Hop, D
1	R_2 , 3
2	R_2 , 1
3	R_3 , 0
4	R_4 , 2

@ R_4

Netid	Next Hop, D
4	R_4 , 0

R_1 R_2 R_3

Netid	Next Hop, D
1	R_1 , 2
2	R_2 , 1
3	R_3 , 2
4	R_4 , 0

R_1 R_2 R_3

Netid	Next Hop, D
1	R_1 , 2
2	R_2 , 1
3	R_3 , 2
4	R_4 , 0