

Intradomain Routing



Hui Chen, Ph.D.
Dept. of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgements

- ❑ Some pictures used in this presentation were obtained from the Internet
- ❑ The instructor used the following references
 - Larry L. Peterson and Bruce S. Davie, Computer Networks: A Systems Approach, 5th Edition, Elsevier, 2011
 - Andrew S. Tanenbaum, Computer Networks, 5th Edition, Prentice-Hall, 2010
 - James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach, 5th Ed., Addison Wesley, 2009
 - Larry L. Peterson's (<http://www.cs.princeton.edu/~llp/>) Computer Networks class web site

Forwarding vs. Routing

□ Forwarding:

- to select an output port based on destination address and routing table

□ Routing:

- to process by which routing table is built

Forwarding Table vs. Routing Table

□ Forwarding table

- Used when a packet is being forwarded and must contain enough information to accomplish the forwarding function
- A row in the forwarding table contains the mapping from a network number to an outgoing interface and some MAC information, such as Ethernet Address of the next hop

□ Routing table

- Built by the routing algorithm as a precursor to build the forwarding table
- Generally contains mapping from network numbers to next hops

Forwarding Table vs. Routing Table: Example

- Example rows from (a) routing and (b) forwarding tables

(a)

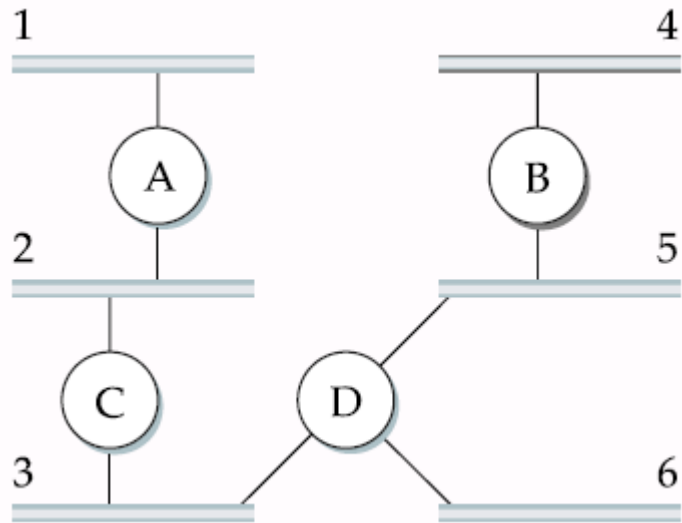
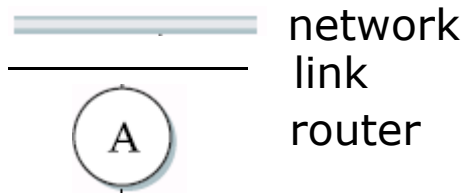
Prefix/Length	Next Hop
18/8	171.69.245.10

(b)

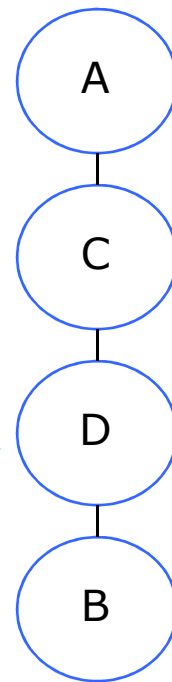
Prefix/Length	Interface	MAC Address
18/8	if0	8:0:2b:e4:b:1:2

Modeling Internetworks as Graph for Routing

Legends:




Routers as nodes
Connection
(network/link) as
edges

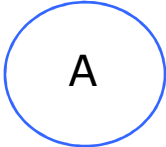


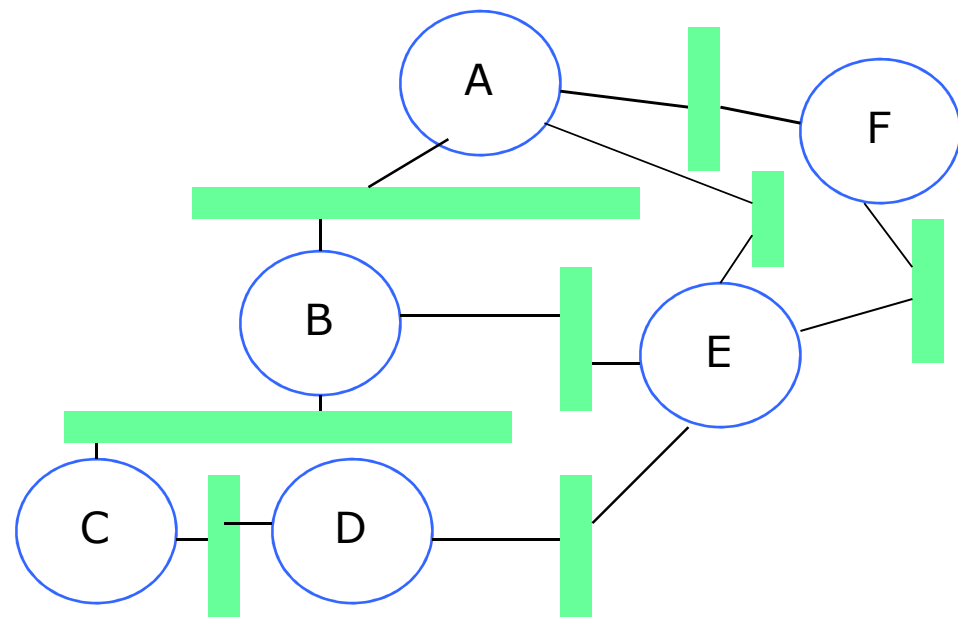
Exercise L12-1

- Use routers as nodes, connections between routers as edges, please construct the graph of the internet shown below

Legends:

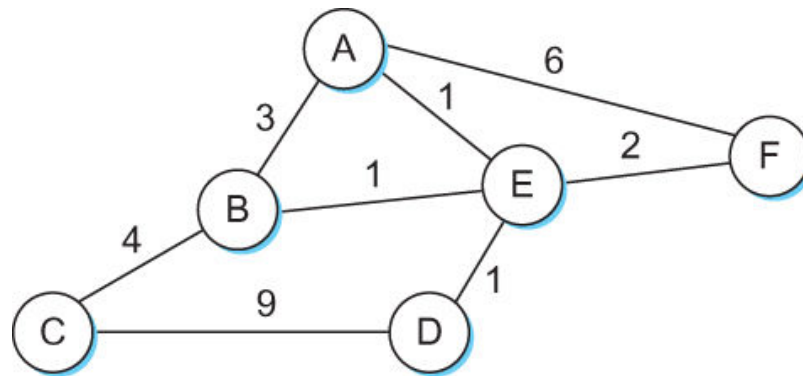
 network Link

 Router



Routing

□ Model Network as a Graph



□ Routing problem

- To find the lowest-cost path between any two nodes
- where the cost of a path equals to the sum of the costs of all the edges that make up the path

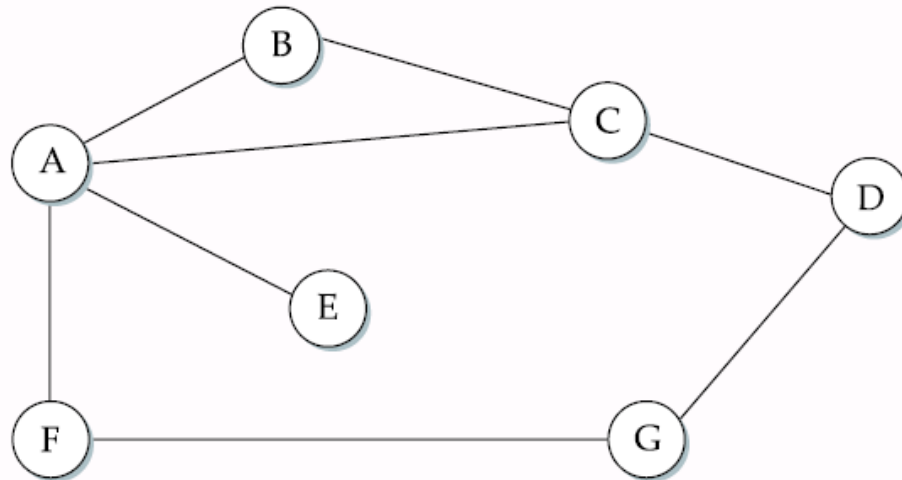
Routing

- ❑ Calculate all shortest paths and load them into some nonvolatile storage on each node
 - Such a static approach has several shortcomings
 - ❑ It does not deal with node or link failures
 - ❑ It does not consider the addition of new nodes or links
 - ❑ It implies that edge costs cannot change
- ❑ What is the solution?
 - Need a distributed and dynamic protocol
 - ❑ Two main classes of protocols
 - Distance Vector
 - Link State

Distance Vector

- ❑ Each node constructs a one dimensional array (a vector) containing the “distances” (costs) to all other nodes and distributes that vector to its immediate neighbors
- ❑ Starting assumption is that each node knows the cost of the link to each of its directly connected neighbors

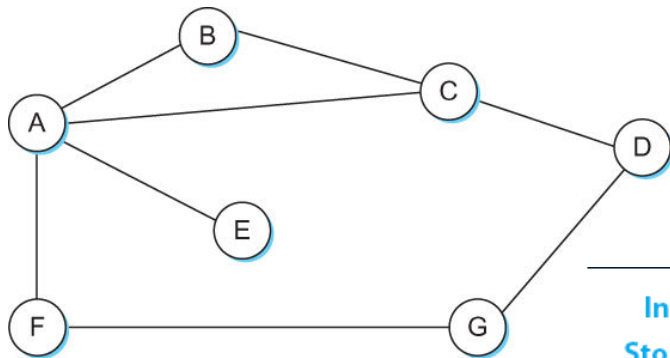
Distance From a Node to Other Nodes



- What is the (shortest) distance from A to B?
- What is the (shortest) distance from A to C?
- What is the (shortest) distance from A to D?

Distance Vector: Example

- Initial distances stored at each node (*global view*)

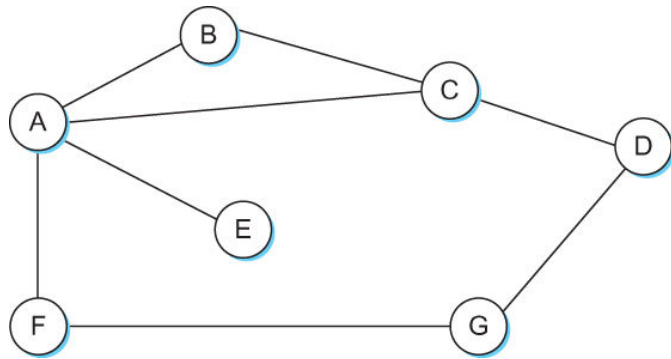


Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

- No node has this global view!

Distance Vector: Example of Initial Routing Table

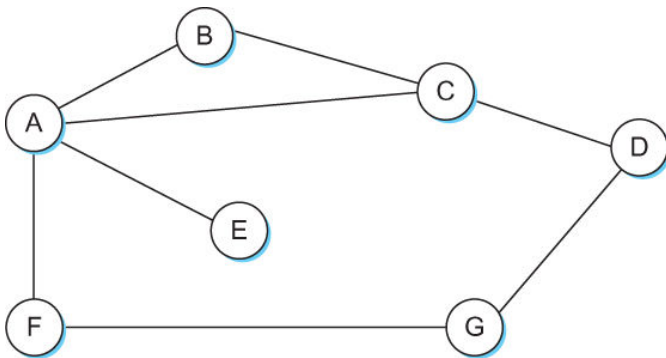
□ Initial routing table at node A



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

Distance Vector: Example of Final Routing Table

Final routing table at node A

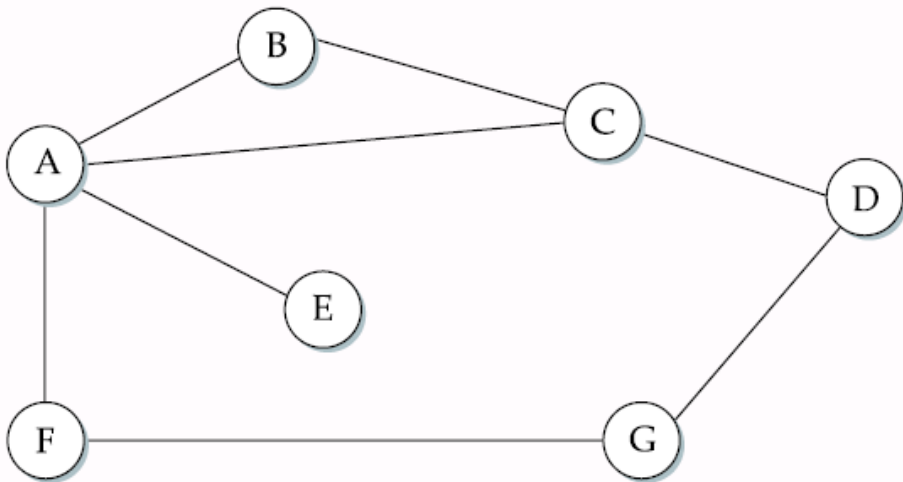


Distance vector: distances from A to the other nodes

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

Exercise L12-2

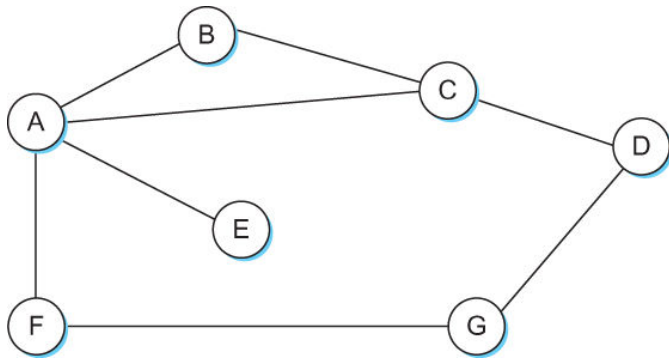
- Given an internetwork below, construct the *initial* routing table for the distance vector routing algorithm at *router C* (by filling the provided table below)



Destination	Cost	Next Hop
A		
B		
D		
E		
F		
G		

Distance Vector: Example

- Final distances stored at each node (*global view*)

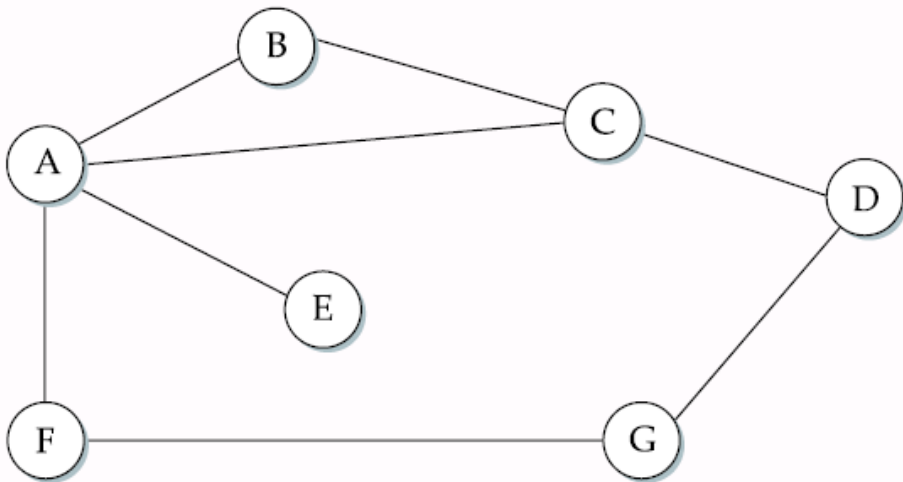


Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

- No node has this global view!

Exercise L12-3

- Given an internetwork below, construct the *final* routing table for the distance vector routing algorithm at *router C* (by filling the provided table below)



Destination	Cost	Next Hop
A		
B		
D		
E		
F		
G		

Distance Vector Routing Algorithm

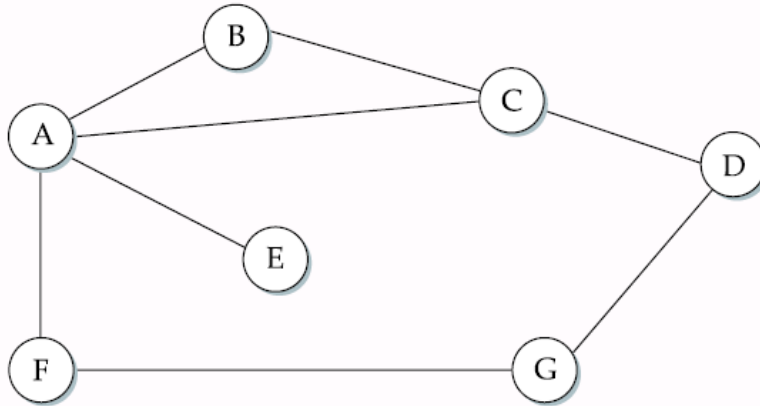
- ❑ Sometimes called as *Bellman-Ford* algorithm
- ❑ Main idea
 - Every T seconds each router sends its table to its neighbor each router then updates its table based on the new information
- ❑ Problems
 - Fast response to good news, but slow response to bad news
 - Also too many messages to update

Distance Vector Routing Algorithm: More Details

- ❑ Each node maintains a routing table consisting of a set of triples
 - (Destination, Cost, NextHop)
- ❑ Exchange updates directly connected neighbors
 - periodically (on the order of several seconds)
 - whenever table changes (called *triggered update*)
- ❑ Each update is a list of pairs:
 - (Destination, Cost): from sending router to destination
 - Update local table if receive a “better” route
 - ❑ smaller cost
 - ❑ came from next-hop
- ❑ Refresh existing routes; delete if they time out

Table Update

- Example: Exchange updates between A and C



- Then A sends an update to C

Destination	Cost
B	1
C	1
D	∞
E	1
F	1
G	∞

C's initial routing table

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	∞	-
F	∞	-
G	∞	-

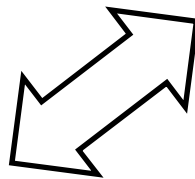
C's updated routing table

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	2	A
F	2	A
G	∞	-

Table Update from A at C

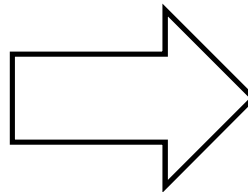
Destination	Cost
B	1
C	1
D	∞
E	1
F	1
G	∞

+ 1 =



Destination	Cost	Next Hop
B	2	A
C	2	A
D	∞	A
E	2	A
F	2	A
G	∞	A

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	∞	-
F	∞	-
G	∞	-



Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	2	A
F	2	A
G	∞	-

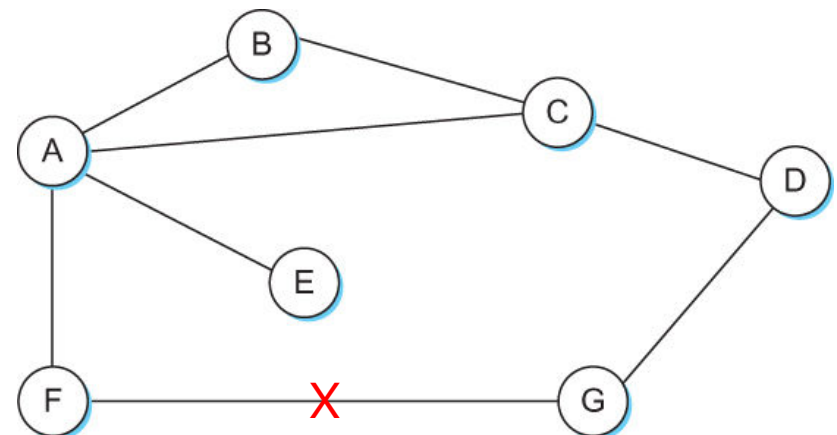
Convergence

- ❑ Process of getting consistent routing information to all the nodes
- ❑ Desired results: routing tables converges to a stable *global* table (no more changes upon receiving updates from neighbors)

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

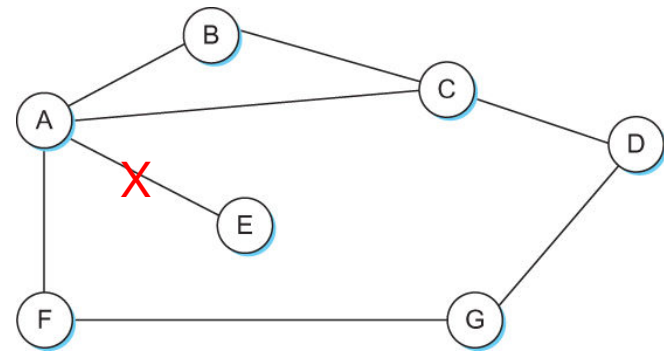
Link Failure: Example

- When a node detects a link failure
 - F detects that link to G has failed
 - F sets distance to G to infinity and sends update to A
 - A sets distance to G to infinity since it uses F to reach G
 - A receives periodic update from C with 2-hop path to G
 - A sets distance to G to 3 and sends update to F
 - F decides it can reach G in 4 hops via A



Count-to-infinity Problem

- ❑ Slightly different circumstances can prevent the network from *stabilizing*
 - Suppose the link from A to E goes down
 - In the next round of updates, A advertises a distance of infinity to E, but B and C advertise a distance of 2 to E
 - Depending on the exact timing of events, the following might happen
 - ❑ Node B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A
 - ❑ Node A concludes that it can reach E in 4 hops and advertises this to C
 - ❑ Node C concludes that it can reach E in 5 hops; and so on.
 - ❑ This cycle stops only when the distances reach some number that is large enough to be considered infinite
 - ❑ ***called count-to-infinity problem***



Count-to-infinity Problem: Solutions

- ❑ Use some relatively small number as an approximation of infinity
- ❑ For example, the maximum number of hops to get across a certain network is never going to be more than 16
 - Set infinity to 16
 - Stabilize fast, but not working for larger networks
- ❑ One technique to improve the time to stabilize routing is called *split horizon*

Split Horizon

- ❑ When a node sends a routing update to its neighbors, it does *not* send those routes it learned from each neighbor *back* to that neighbor
- ❑ For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update

Split Horizon with Poison Reverse

- In a stronger version of split horizon, called *split horizon with poison reverse*
 - B actually sends that back route to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E
 - For example, B sends the route (E, ∞) to A

Routing Information Protocol

- Routing Information Protocol (RIP)
 - Initially distributed along with BSD Unix
 - Widely used
- Straightforward implementation of distance-vector routing

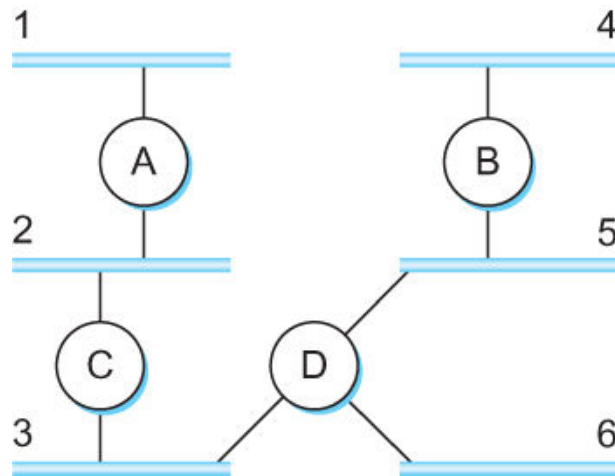
Routing Information Protocol (RIP)

Distance: cost (# of routers) of reach a network

C → A

Network 2 at cost 0; 3 at cost 0

Network 5 at cost 1, 4 at 2



Example Network

0		8		16		31	
Command		Version		Must be zero			
Family of net 1				Route Tags			
Address prefix of net 1							
Mask of net 1							
Distance to net 1							
Family of net 2				Route Tags			
Address prefix of net 2							
Mask of net 2							
Distance to net 2							

RIPv2 Packet Format

Link State Routing

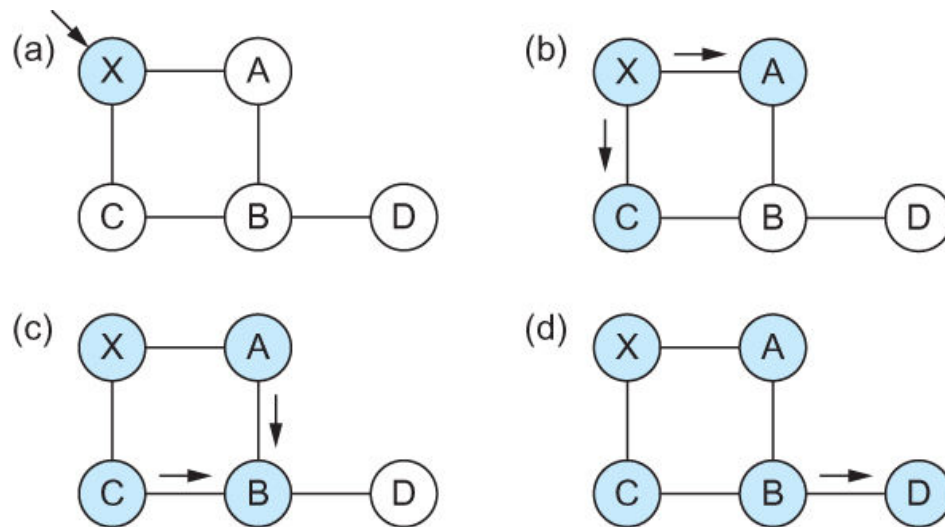
- ❑ Strategy: Send to all nodes (not just neighbors) information about directly connected links (not entire routing table).
- ❑ Link State Packet (LSP)
 - id of the node that created the LSP
 - cost of link to each directly connected neighbor
 - sequence number (SEQNO)
 - time-to-live (TTL) for this packet
- ❑ Reliable Flooding
 - store most recent LSP from each node
 - forward LSP to all nodes but one that sent it
 - generate new LSP periodically; increment SEQNO
 - start SEQNO at 0 when reboot
 - decrement TTL of each stored LSP; discard when TTL=0

Link State Routing

- Reliable flooding triggered by
 - Timer
 - Topology or link cost change
- increment SEQNO
 - start SEQNO at 0 when reboot
 - SEQNO does not wrap
 - e.g., 64 bits
 - decrement TTL of each stored LSP
- discard when TTL=0

Link State Routing: Example

□ Reliable Flooding



- Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) A and C flood LSP to B (but not X); (d) flooding is complete

Shortest Path Routing Algorithm

□ Dijkstra's Algorithm

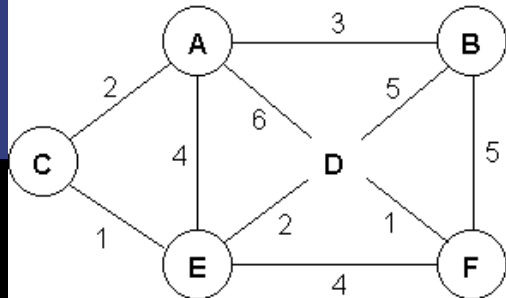
- Assume non-negative link weights
- N : set of nodes in the graph
- $l(i, j)$: the non-negative cost associated with the edge between nodes $i, j \in N$ and $l(i, j) = \infty$ if no edge connects i and j
- Let $s \in N$ be the starting node which executes the algorithm to find shortest paths to all other nodes in N
- Two variables used by the algorithm
 - M : set of nodes incorporated so far by the algorithm
 - $C(n)$: the cost of the path from s to each node n

Shortest Path Routing Algorithm

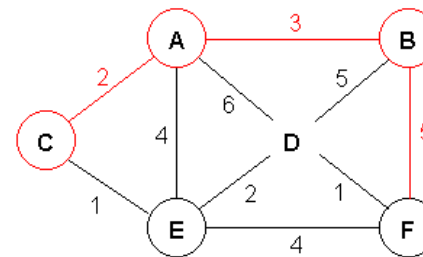
□ Dijkstra's Algorithm - Assume non-negative link weights

```
M = {s}
For each n in N - {s}
    C(n) = l(s, n)
while ( N ≠ M)
    M = M ∪ {w} such that C(w) is the minimum
                        for all w in (N-M)
    For each n in (N-M)
        C(n) = MIN (C(n), C(w) + l(w, n))
```

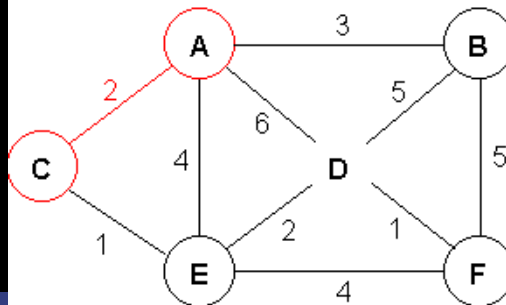
Dijkstra's shortest path algorithm



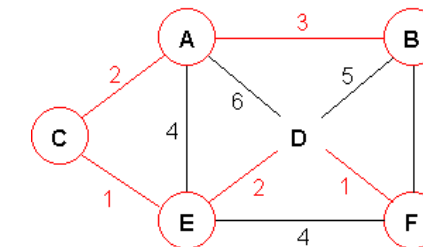
Step	Costs					visited?
	B	C	D	E	F	
Init	3	2	6	4	-	C



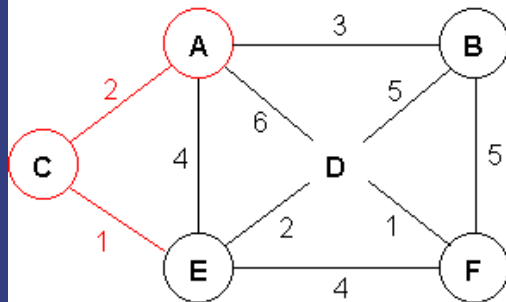
Step	Costs					visited?
	B	C	D	E	F	
Init	3	2	6	4	-	C
1	3	2	6	3	-	B
2	3	2	6	3	8	



Step	Costs					visited?
	B	C	D	E	F	
Init	3	2	6	4	-	C



Step	Costs					visited?
	B	C	D	E	F	
Init	3	2	6	4	-	C
1	3	2	6	3	-	B
2	3	2	6	3	8	E
3	3	2	5	3	7	D
4	3	2	5	3	6	F



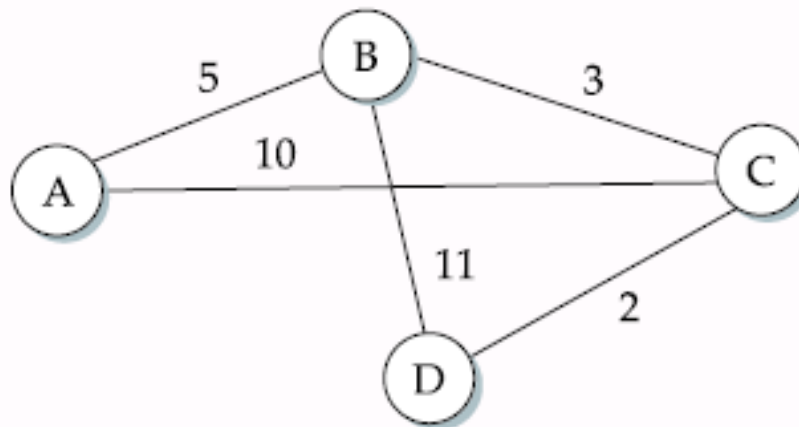
Step	Costs					visited?
	B	C	D	E	F	
Init	3	2	6	4	-	C
1	3	2	6	3	-	

Jens Brodowski, Animation of Dijkstra's shortest path algorithm, Available:

<http://www.animal.ahrgr.de/showAnimationDetails.php3?lang=en&anim=15>

Exercise L12-4

- Following the example illustrated and using the Dijkstra's shortest path algorithm, find the shortest path to all the other nodes from node D and show steps



Shortest Path Routing Algorithm

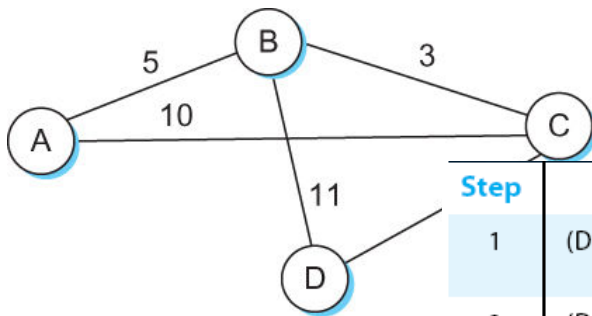
- ❑ In practice, each switch computes its routing table directly from the LSPs it has collected using a realization of Dijkstra's algorithm called the *forward search algorithm*
- ❑ Specifically each switch maintains two lists, known as **Tentative** and **Confirmed**
- ❑ Each of these lists contains a set of entries of the form (Destination, Cost, NextHop)

Shortest Path Routing Algorithm in Linked State Routing

- Each router runs the algorithm
 - Initialize the **Confirmed** list with an entry for myself; this entry has a cost of 0
 - For the node just added to the **Confirmed** list in the previous step, call it node **Next**, select its LSP
 - For each neighbor (Neighbor) of **Next**, calculate the cost (Cost) to reach this Neighbor as the sum of the cost from myself to Next and from Next to Neighbor
 - If Neighbor is currently on neither the **Confirmed** nor the **Tentative** list, then add (Neighbor, Cost, Nexthop) to the **Tentative** list, where Nexthop is the direction I go to reach Next
 - If Neighbor is currently on the **Tentative** list, and the Cost is less than the currently listed cost for the Neighbor, then replace the current entry with (Neighbor, Cost, Nexthop) where Nexthop is the direction I go to reach Next
 - If the **Tentative** list is empty, stop. Otherwise, pick the entry from the **Tentative** list with the lowest cost, move it to the **Confirmed** list, and return to Step 2.

Shortest Path Routing: Example

- Forward search algorithm: building routing table in D from received LSP's



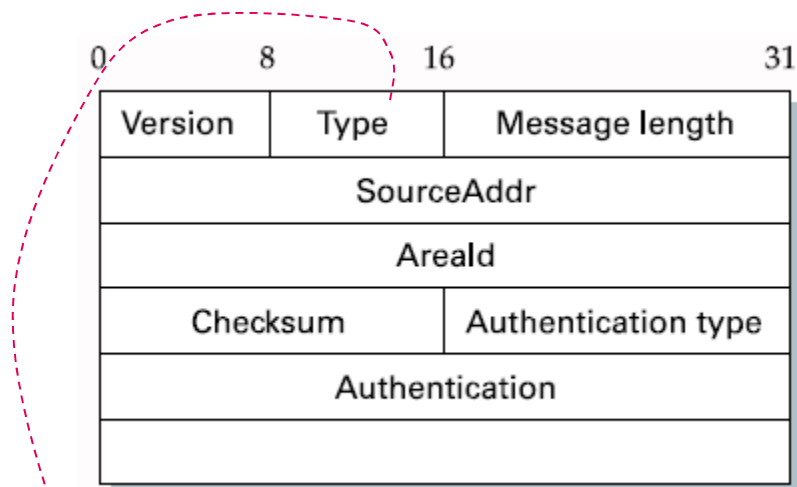
Step	Confirmed	Tentative	Comments
1	(D,0,-)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,-)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,-) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,-) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

Link State in Practice

- ❑ Open Shortest Path First Protocol (OSPF)
 - “Open” → open, non-proprietary standard, created under the auspices of the IETF
 - “SPF” → Shortest Path First, alternative name of link-state routing
- ❑ Implementation of Link-State Routing with added features
 - Authenticating of routing messages
 - ❑ Due to the fact too often some misconfigured hosts decide they can reach every host in the universe at a cost of 0
 - Additional hierarchy
 - ❑ Partition domain into areas → increase scalability
 - Load balancing
 - ❑ Allows multiple routes to the same place to be assigned the same cost → cause traffic to be distributed evenly over those routes

Open Shortest Path First Protocol

OSPF Header Format



OSPF Link State Advertisement

LS Age		Options		Type=1	
Link-state ID					
Advertising router					
LS sequence number					
LS checksum			Length		
0	Flags	0	Number of links		
Link ID					
Link data					
Link type	Num_TOS	Metric			
Optional TOS information					
More links					

Type	Packet name	Protocol function
1	Hello	Discover/maintain neighbors
2	Database Description	Summarize database contents
3	Link State Request	Database download
4	Link State Update	Database update
5	Link State Ack	Flooding acknowledgment

Metrics

- ❑ Original ARPANET metric
 - measures number of packets enqueued on each link
 - took neither latency or bandwidth into consideration
- ❑ New ARPANET metric
 - stamp each incoming packet with its arrival time (AT)
 - record departure time (DT)
 - when link-level ACK arrives, compute
- ❑ $\text{Delay} = (\text{DT} - \text{AT}) + \text{Transmit} + \text{Latency}$
 - if timeout, reset DT to departure time for retransmission
 - link cost = average delay over some time period
- ❑ Fine Tuning
 - compressed dynamic range
 - replaced Delay with link utilization

Summary

□ Distance Vector

- Algorithm
- Routing Information Protocol (RIP)

□ Link State

- Algorithm
- Open Shortest Path First Protocol (OSPF)

□ Metrics

- How to measure link cost?