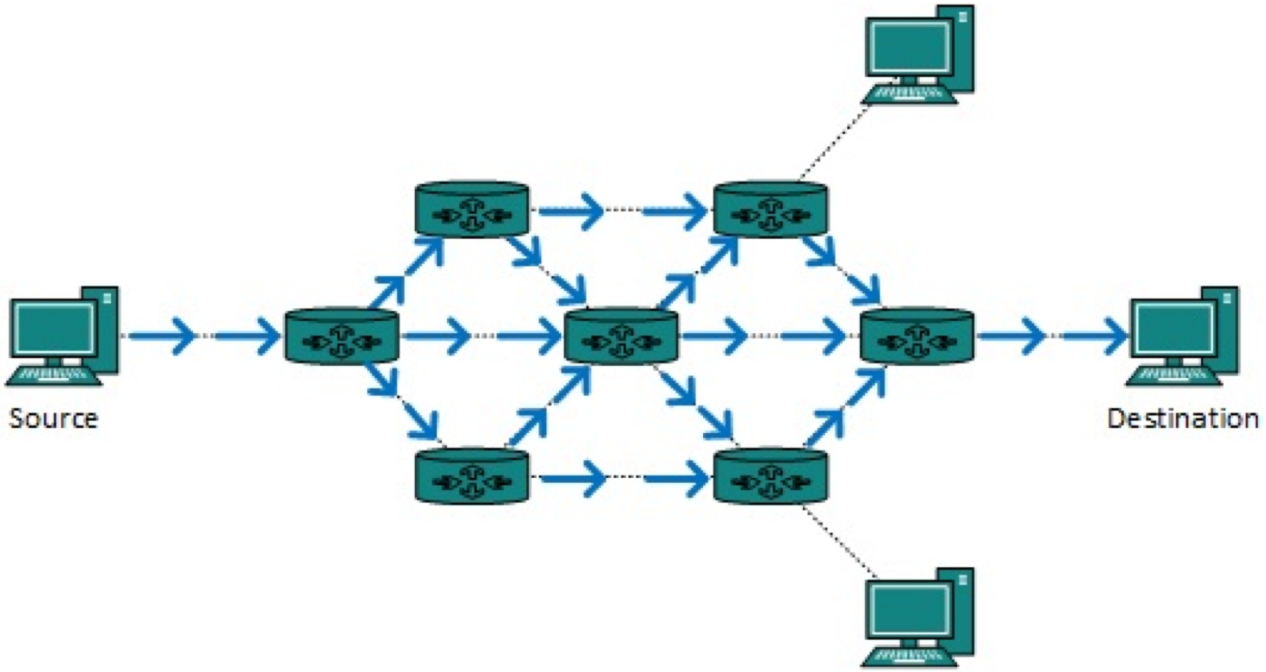
Lecture 16 Overview

- Last Lecture
  - Wide Area Networks (1)
- This Lecture
  - Wide Area Networks (2)
  - Source: chapters 20.1, 20.2
- Next Lecture
  - Internet Protocol (1)
  - Source: chapters 2.1, 18.4, 9.2

Today's Focus

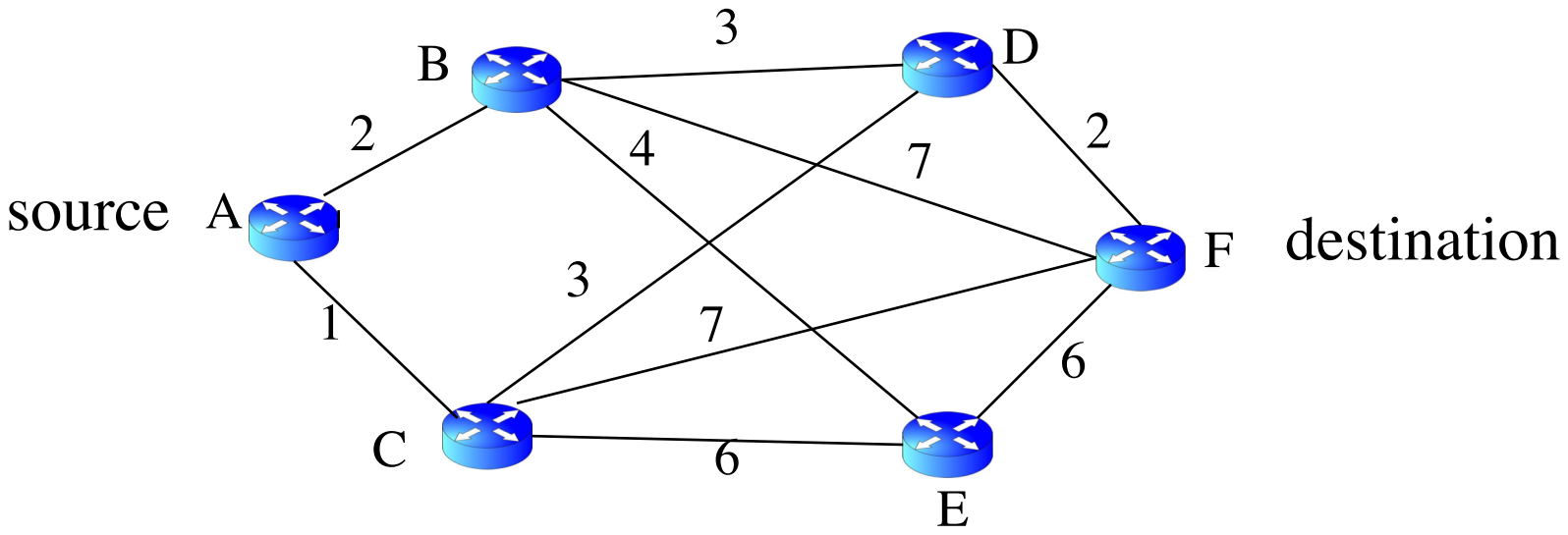
Routing Algorithm

How to choose the best path to route packets from a source to a destination?



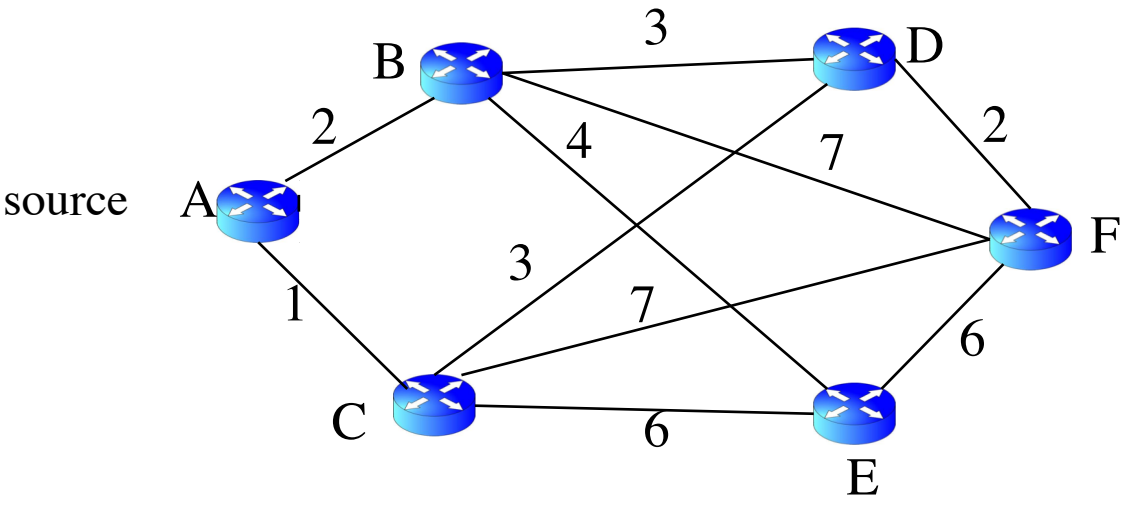
Routing Algorithm

- The process of selecting paths in a network along which to send network traffic
  - Determine the best route
  - Create routing tables



Shortest Path Routing

- Developed by Dijkstra in 1956
  - also called *Dijkstra's algorithm,* or *forward search algorithm*
  - Centralized algorithm
- Problem:
  - Find the shortest path from a source node to any other node in the graph.



Shortest Path Routing (cont.)

- Notations
  - *S*: the set of nodes for which the cheapest path from *A* is known
  - *W*: the set of nodes not in S, but connected to at least one node in *S* via a direct link
  - *Cost(X)*: the cost of the cheapest path from *A* to *X* for which intermediate nodes are in *S*
  - *Prior(X)*: is the node preceding X in the cheapest route

Shortest Path Routing (cont.)

- Algorithm

(1) Initialization: *S={A}; Cost(X)* is the cost of the link from *A* to *X*; if no such link exists, *Cost(X)* is infinite. For those nodes linked to *A,* define *Prior(X)=A.*

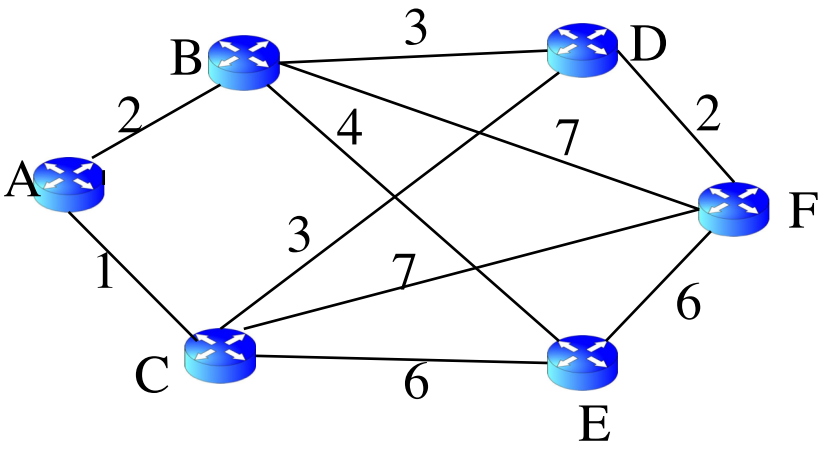
(2) Find W, the nodes which are not in S, but connected to a node in S.

(3) Choose a node X in W for which Cost(X) is minimum and add X to the set S.

(4) For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+cost of link connecting X to V}. If Cost(V) is changed, Prior(V)=X.

(5) Repeat steps (2), (3) and (4) until all nodes are in S.

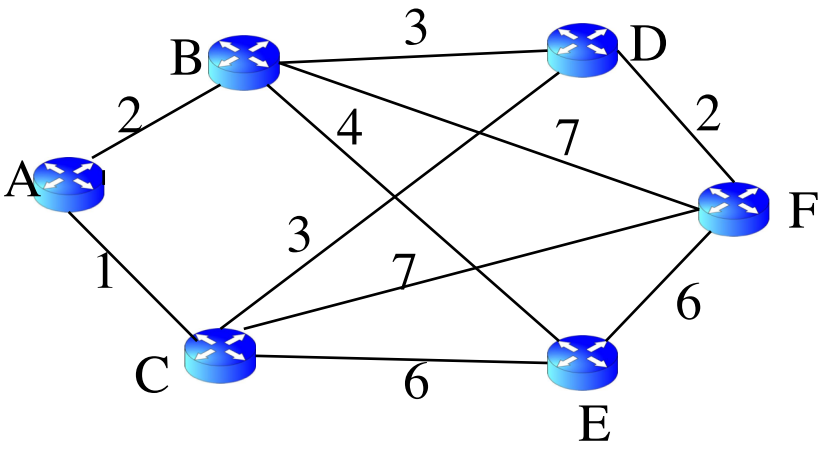
Shortest Path Routing (cont.)



Step 1:
S={A}; Cost(X) is the cost of the link from *A* to *X*; if no such link exists, *Cost(X)* is infinite. For those nodes linked to *A*, define *Prior(X)=A*.

	S	W	X	Cost(X)					Prior(X)				
				B	C	D	E	F	B	C	D	E	F
1	{A}			2	1	∞	∞	∞	A	A	–	–	–
2													
3													
4													
5													

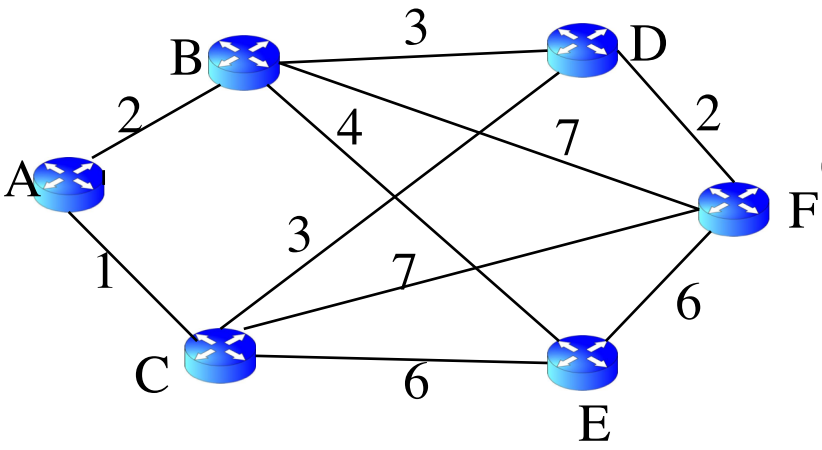
Shortest Path Routing (cont.)



Step 2:
Find *W*, the nodes which are not in *S*, but are connected to a node in *S*.

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}		2	1	∞	∞	∞	A	A	–	–	–
2													
3													
4													
5													

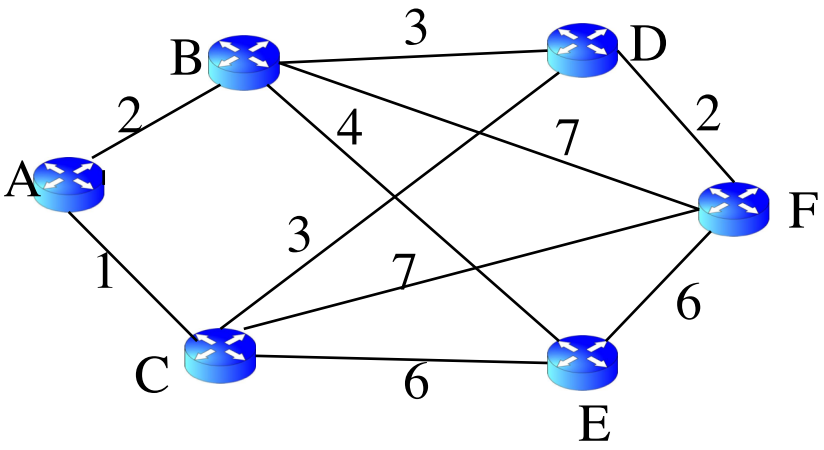
Shortest Path Routing (cont.)



Step 3:
Choose a node *X* in *W* for which *Cost(X)* is minimum and add *X* to the set *S*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}												
3													
4													
5													

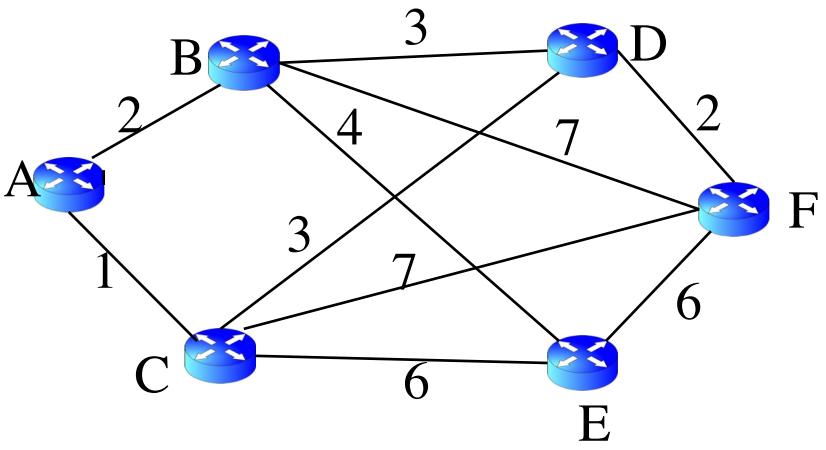
Shortest Path Routing (cont.)



Step 4:
For each *V* not in *S*,
Cost(V)=minimum{Cost(V), Cost(X)+cost of link connecting X to V}. If *Cost(V)* is changed *Prior(V)=X*.

	S	W	X	Cost(X)					Prior(X)				
				B	C	D	E	F	B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}			2	1	4	7	8	A	A	C	C	C
3													
4													
5													

Shortest Path Routing (cont.)



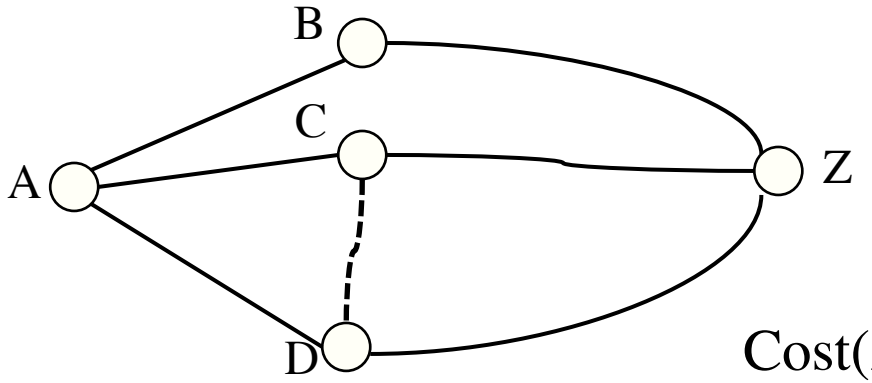
Step 5:
Repeat steps (2), (3), (4) until all nodes are in *S*.

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A,B,C}	{D, E, F}	D	2	1	4	6	8	A	A	C	B	C
4	{A,B,C,D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A,B,C,D,E}	{F}	F	2	1	4	6	6	A	A	C	B	D

Demo: http://www.unf.edu/~wkloster/foundations/DijkstraApplet/DijkstraApplet.htm

Distance Vector Routing

- Developed by Bellman-Ford, also called Bellman-Ford algorithm
- Used in ARPANET until 1979
- Basic idea: backward search



Cost(A, Z): cost of the cheapest route from A to Z

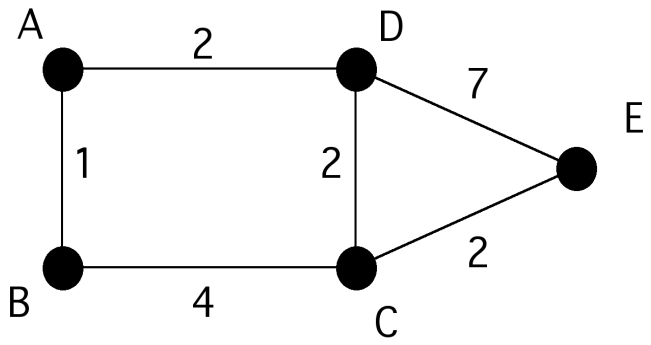
Neigh(A): neighbors of A

$$\text{Cost}(A, Z) = \min_{X \in \text{Neigh}(A)} \{\text{Cost}(A, X) + \text{Cost}(X, Z)\}$$

Distributed Bellman-Ford Algorithm

- Initialization
  - Initially, each router uses an ECHO packet to learn the cost to each of its neighbours.
- Sharing
  - Periodically each router sends each neighbour a list of estimated costs to each destination it knows about.
- Update
  - Each router updates its tables with the best cost to each destination according to that received from its neighbours.
    - If no route exists, insert a new entry
    - If a route exists and the new route has smaller cost, update to the new route

Bellman-Ford Example



$$Cost(A, Z) = \min_{X \in Neigh(A)} \{Cost(A, X) + Cost(X, Z)\}$$

		DESTINATION				
		A	B	C	D	E
SOURCE	A		(B, 1)	unknown	(D, 2)	unknown
	B	(A, 1)		(C, 4)	unknown	unknown
	C	unknown	(B, 4)		(D, 2)	(E, 2)
	D	(A, 2)	unknown	(C, 2)		(E, 7)
	E	unknown	unknown	(C, 2)	(D, 7)	

		DESTINATION				
		A	B	C	D	E
SOURCE	A		(B, 1)	(D,4)	(D, 2)	(D, 9)
	B					
	C					
	D					
	E					

Good news (decrease in cost) propagates quickly.
Bad news (increase in cost) propagates slowly.

Link State Routing

- Developed to replace Distance Vector in ARPANET
- Variants of Link State routing are now widely used.
- Similar to Distance Vector routing:
  - Each node communicates what it knows to its neighbours.
- Different from Distance Vector:
  - Information exchanged and how to store and process it.
  - Link information is more timely propagated

Idea Behind Link State Routing

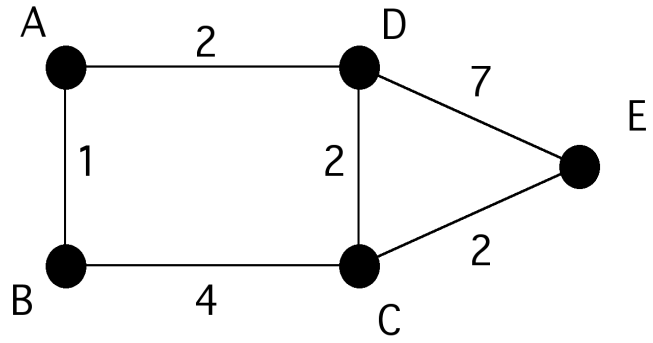
- A node gathers information on the status of each link to each neighbour.
  - bit rate, delay time reliability, number of queued packets
- The node builds a *link state packet* for each link.
  - Identifies the two nodes connected by the link.
  - Information about the link.
- A node receiving *link state packets* forwards them to all its neighbours.
  - Except the one from which it received the packet

Idea Behind Link State Routing (cont.)

- As *link state packets* are exchanged among nodes, each node eventually learns about the network topology and the cost and status of links between network nodes
- Each node can execute a cheapest route algorithm such as Dijkstra's algorithm to determine its routing table.

Link State Routing Example

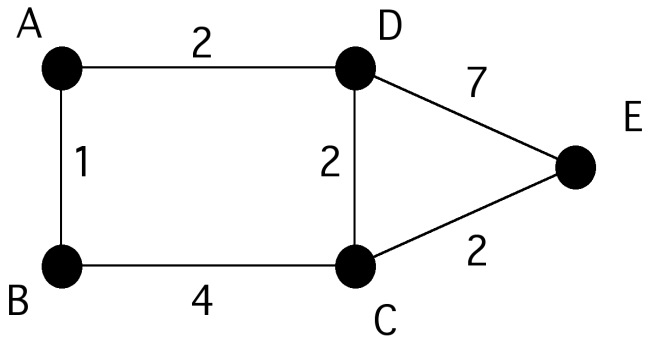
- After sending ECHO packets, the following is known:
  - A: AB=1, AD=2
  - B: BA=1, BC=4
  - C: CB=4, CD=2, CE=2
  - D: AD=2, DC=2, DE=7
  - E: EC=2, ED=7



- Each node creates a link state packet containing the information it knows.
- *Flood* the link state packets among the network, so that each node has the following information
  - AB=1, AD=2, BC=4, CD=2, CE=2, DE=7
- Each node executes a cheapest route algorithm based on the above information and creates a routing table.

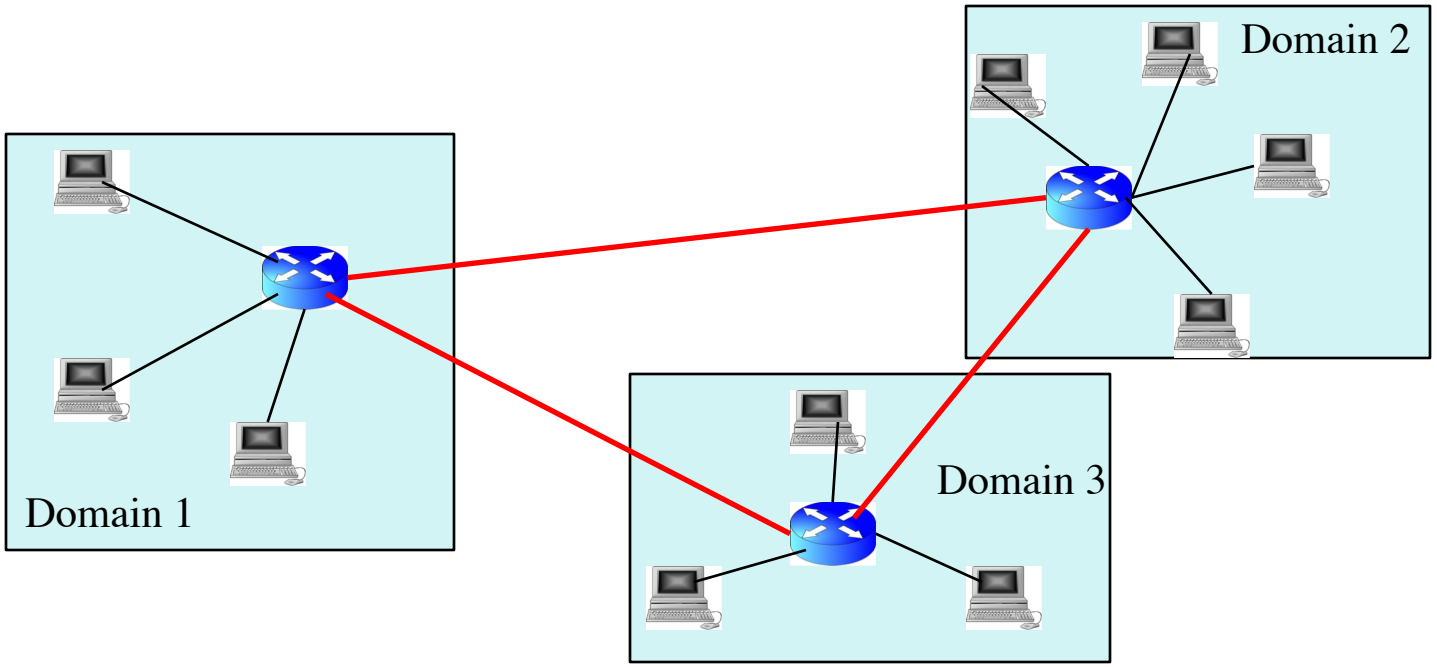
Changes in Link State Routing

- Assume *CD* link becomes 20
- *C and D* sends the information in the next round of *link state* packet exchanges.
- *A* will use this information and all other link information to execute the cheapest route algorithm and create a correct routing table.



Hierarchical Routing

- Previous algorithms require each node to keep proper routing information.
- There may be too many nodes for each to have complete routing tables.
  - Too much information to share in a large network.

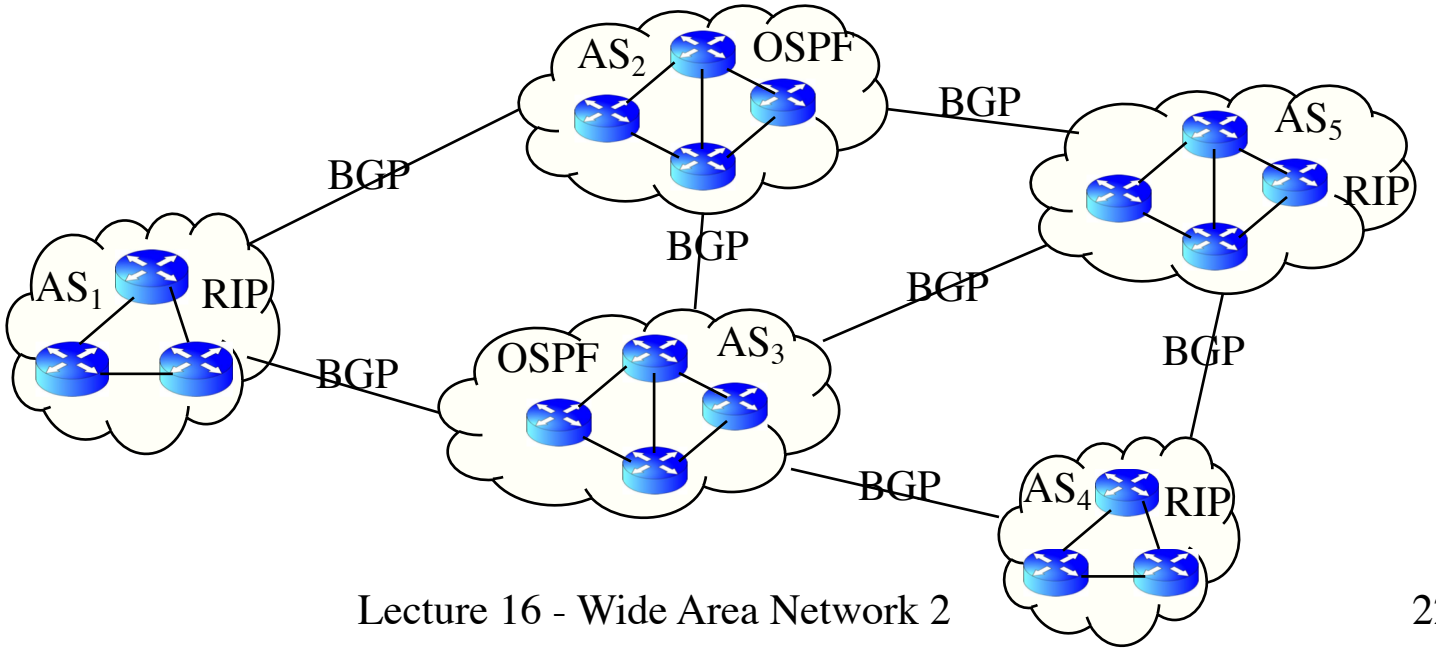


Hierarchical Routing (cont.)

- Nodes are divided into groups called domains.
  - considered a separate and independent network
- Routes between two nodes in a common domain are determined using the domain's or network's protocols.
- Each domain has one or more specially designated nodes called *routers* which determine routes between domains.
- Effectively the routers form a network.
- Domains may consist of subdomains.
  - Each has its own router

Routing in Internet

- Autonomous System (AS): a group of networks and routers controlled by a single administrative authority
- Intra-AS routing:
  - Routing Information Protocol (RIP): distance-vector routing
  - Open Shortest Path First (OSPF): link-state routing
- Inter-AS routing:
  - Border Gateway Protocol (BGP) : path-vector routing



Summary

- Routing algorithms
  - Shortest Path Routing
  - Distance Vector Routing
  - Link State Routing
  - Hierarchical Routing
- Routing in Internet