CEN445 – Network Protocols and Algorithms

Chapter 5 – Network Layer

5.2 Routing Algorithms

Dr. Mostafa Hassan Dahshan
Department of Computer Engineering
College of Computer and Information Sciences
King Saud University
mdahshan@ksu.edu.sa
http://faculty.ksu.edu.sa/mdahshan

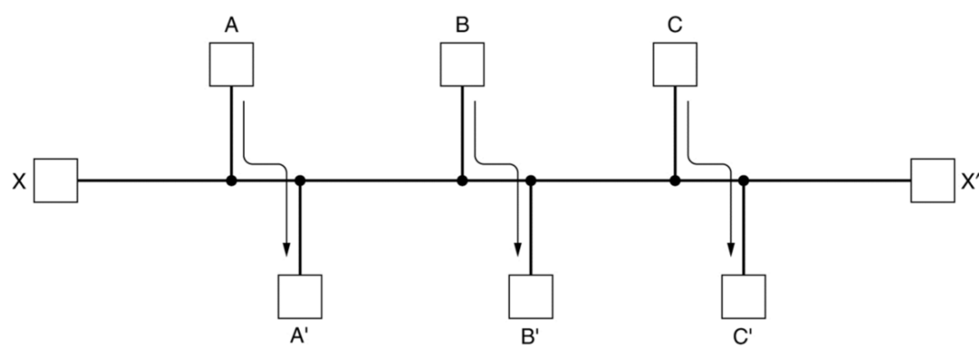
Routing Algorithms

- Routing – main function of network layer
- Routing algorithm
  - decides which output line incoming packet should be transmitted on
  - fills up and updates routing tables
- Forwarding
  - look up the routing tables and put the packet in the appropriate output line

Desired Properties

- Correctness
- Simplicity
- Robustness: ability to handle failures
- Stability: converge to equilibrium
- Fairness
- Optimality

Possible Conflict



Two Major Classes

- Non-adaptive/static routing
  - routing decisions not based on traffic, topology
  - instead, routes are computed in advance
- Adaptive routing
  - Change their decisions to reflect changes in the topology and traffic
  - Differ in: information source, update frequency and optimization metrics

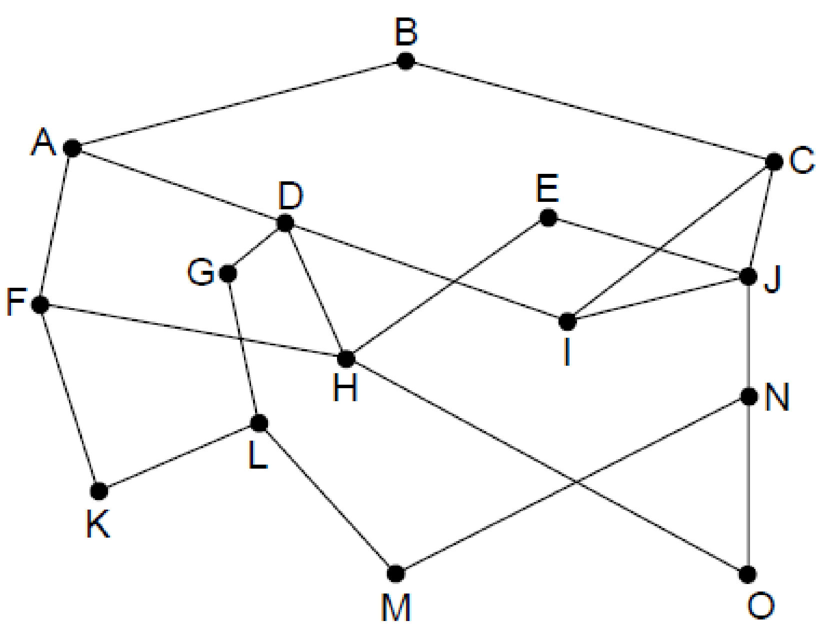
Optimality Principle

- If router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route
- Set of optimal routes from all sources to a given destination form a tree rooted at the destination "sink tree"
- Goal of all routing algorithms: discover and use sink tree for all routers

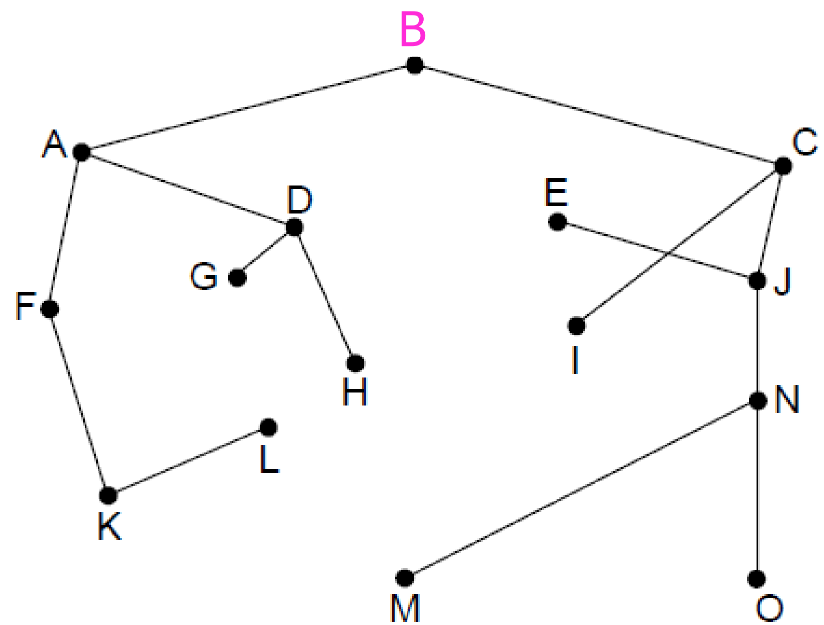
The Optimality Principle

Each portion of a best path is also a best path; the union of them to a router is a tree called the sink tree

- Best means fewest hops in the example



Network



Sink tree of best paths to router B

Shortest Path Routing

- Build a graph of network
- Each node represent a router
- Each arc represent a link
- Find shortest path between the two nodes

Shortest Path Routing

- Each arc is labeled with a weight
  - number of hops
  - geographic distance
  - mean queuing/transmission delay
  - bandwidth
  - cost

Dijkstra's Algorithm

- Finds shortest paths from given source node s to all other nodes
- Develops paths in order of increasing path length
- Runs in stages, each time adding node with next shortest path
- algorithm terminates when all nodes processed by algorithm (in set T)

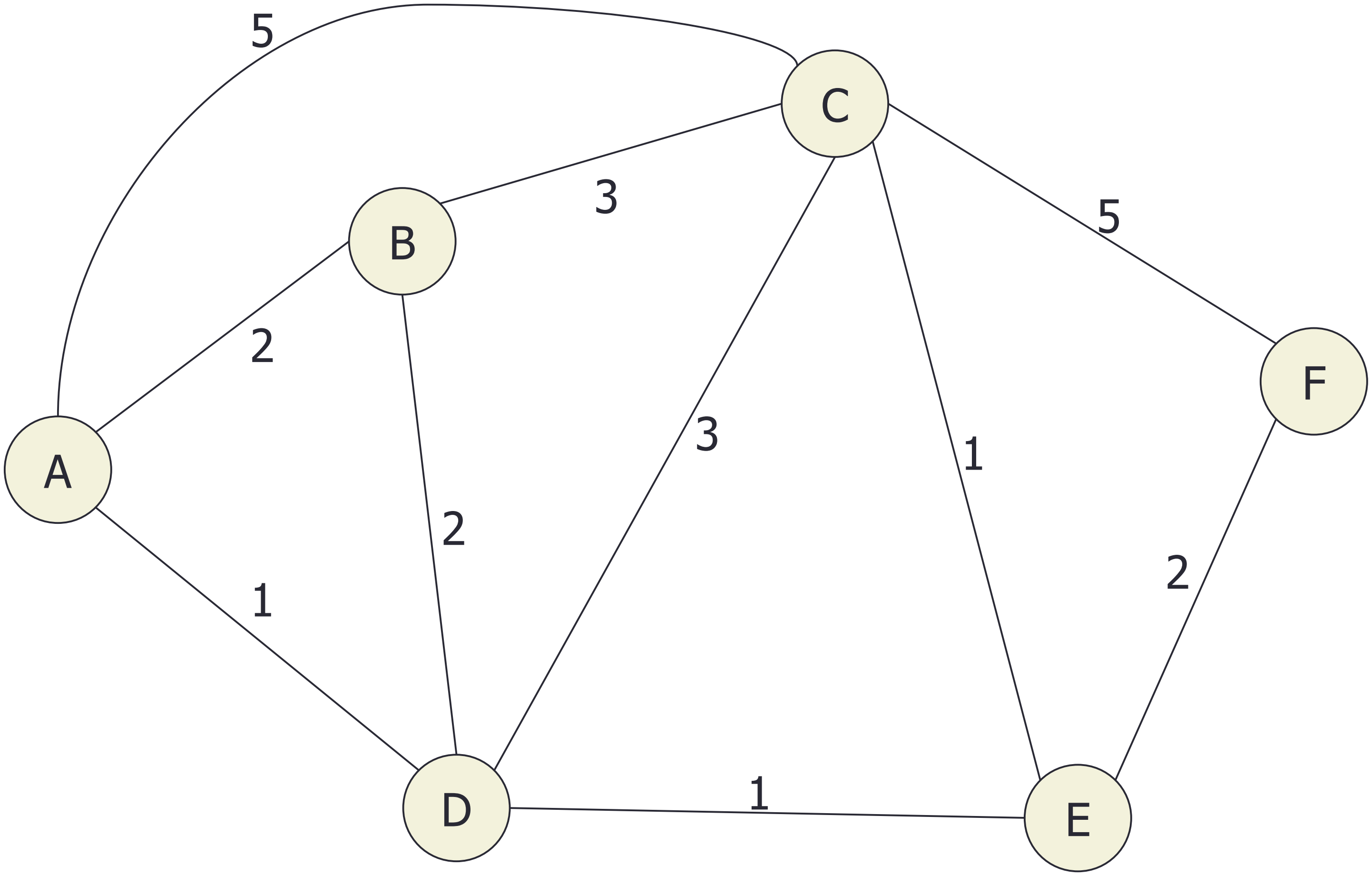
Dijkstra's Algorithm

- Step 1 [Initialization]
  - $T = \{s\}$ Set of nodes so far incorporated
  - $L(n) = w(s, n)$ for $n \neq s$
  - initial path costs to neighboring nodes are simply link costs

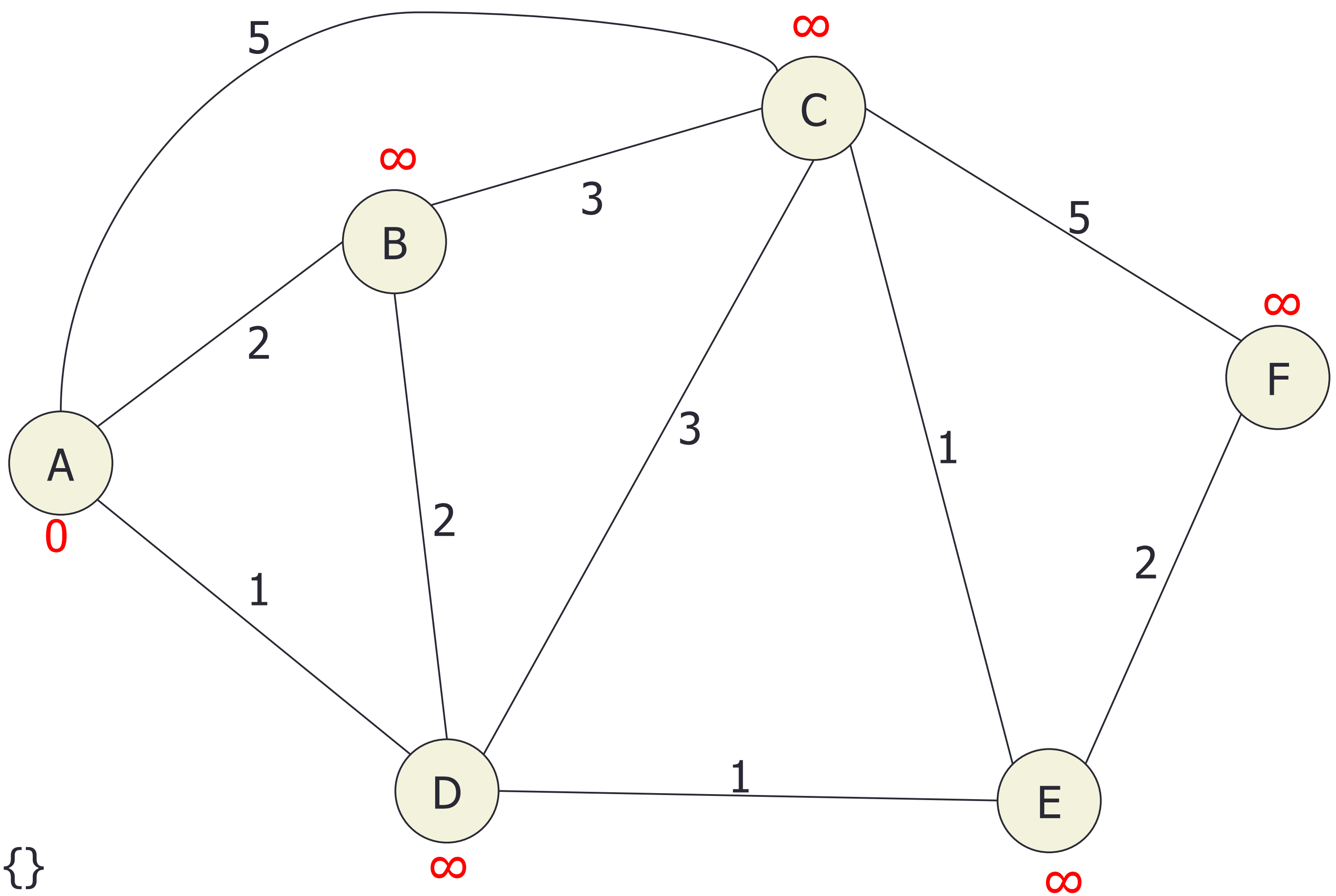
Dijkstra's Algorithm

- Step 2 [Get Next Node]
  - find neighboring node not in T with least-cost path from s
  - incorporate node x into T *(node marked as permanent)*
  - also incorporate the edge that is incident on that node and a node in T that contributes to the path
- Step 3 [Update Least-Cost Paths]
  - $L(n) = \min[L(n), L(x) + w(x, n)]$ for all $n \notin T$
  - if latter term is minimum, path from s to n is path from s to x concatenated with edge from x to n

Dijkstra's Algorithm

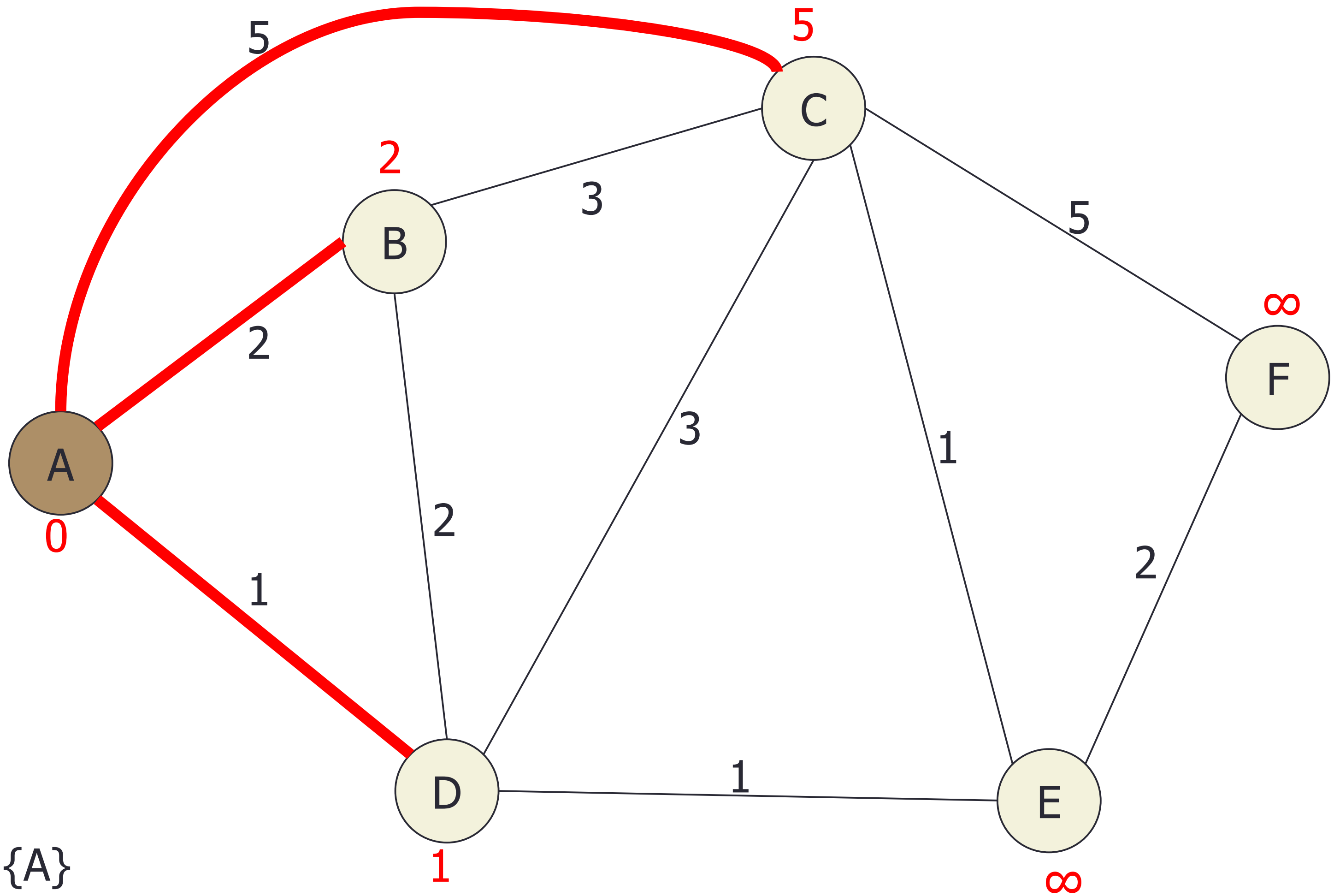


Dijkstra's Algorithm



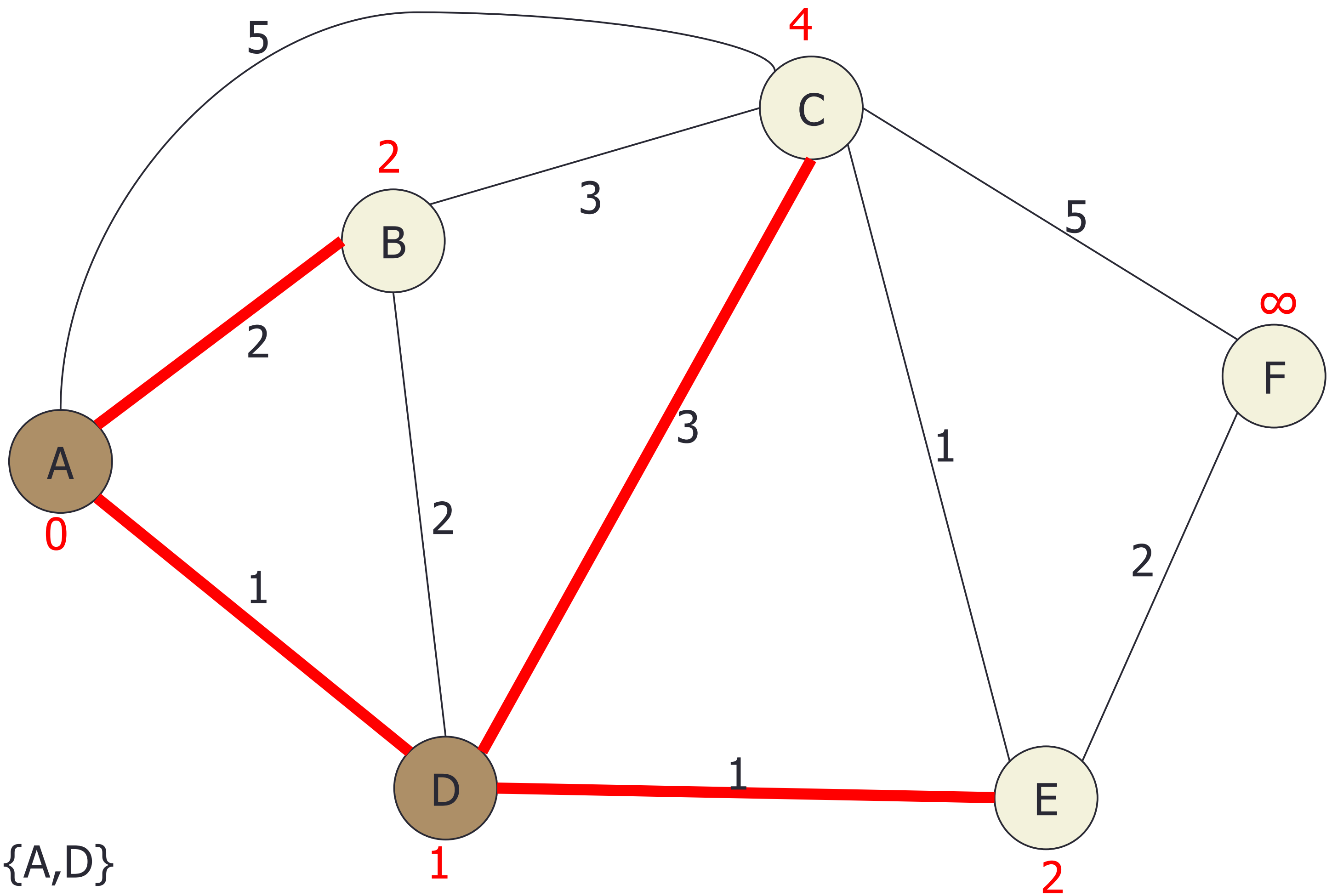
T={}

Dijkstra's Algorithm



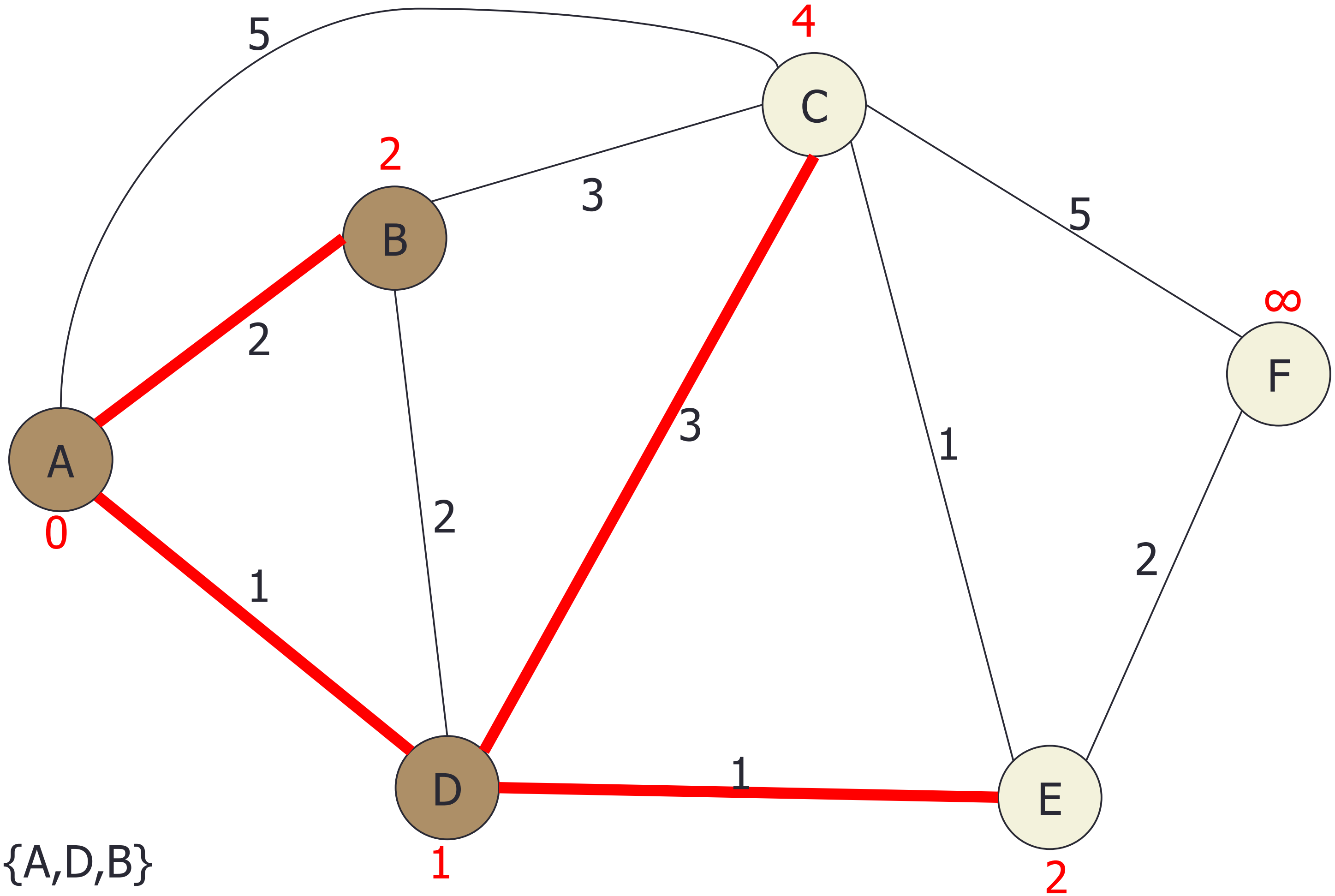
T={A}

Dijkstra's Algorithm



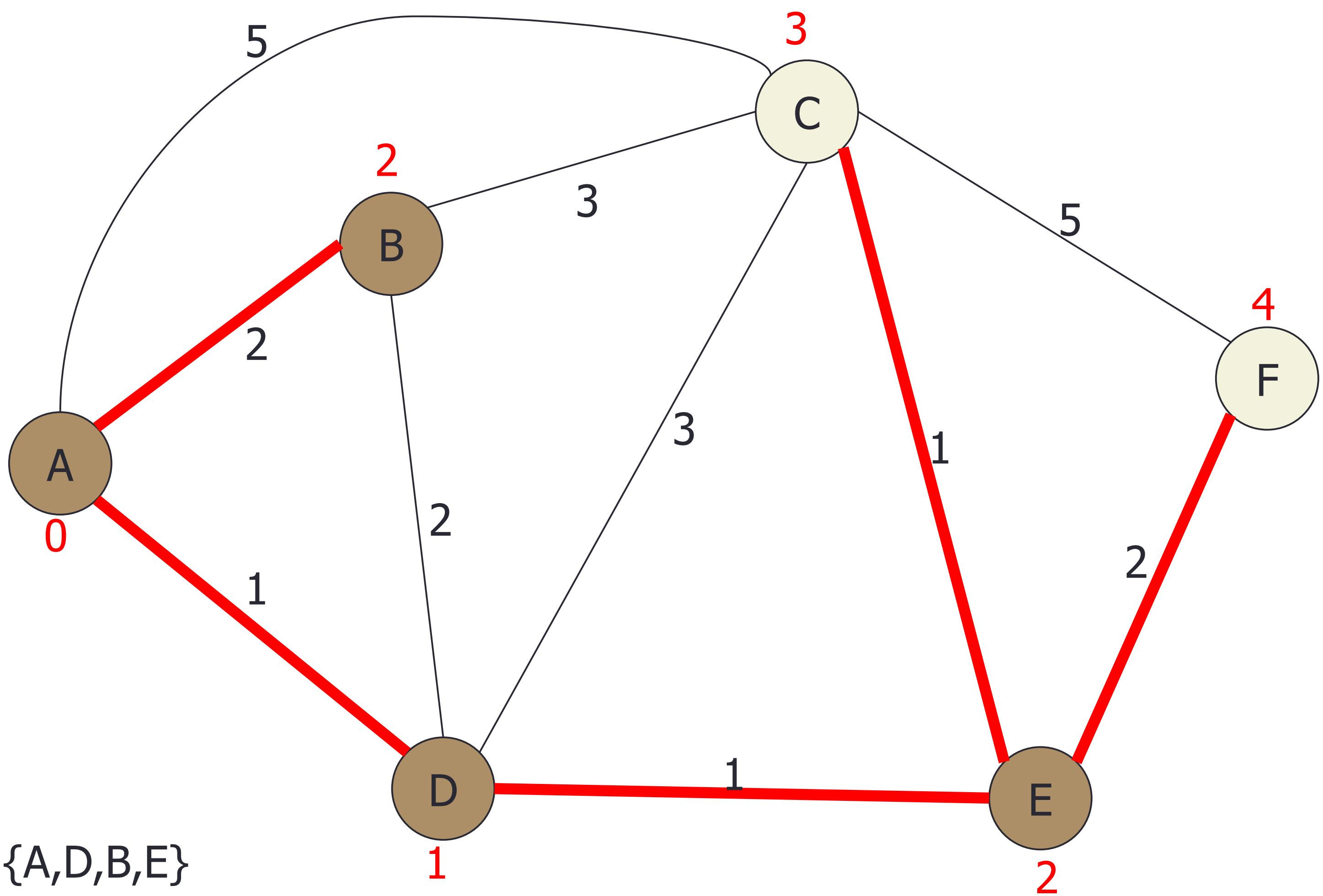
T={A,D}

Dijkstra's Algorithm



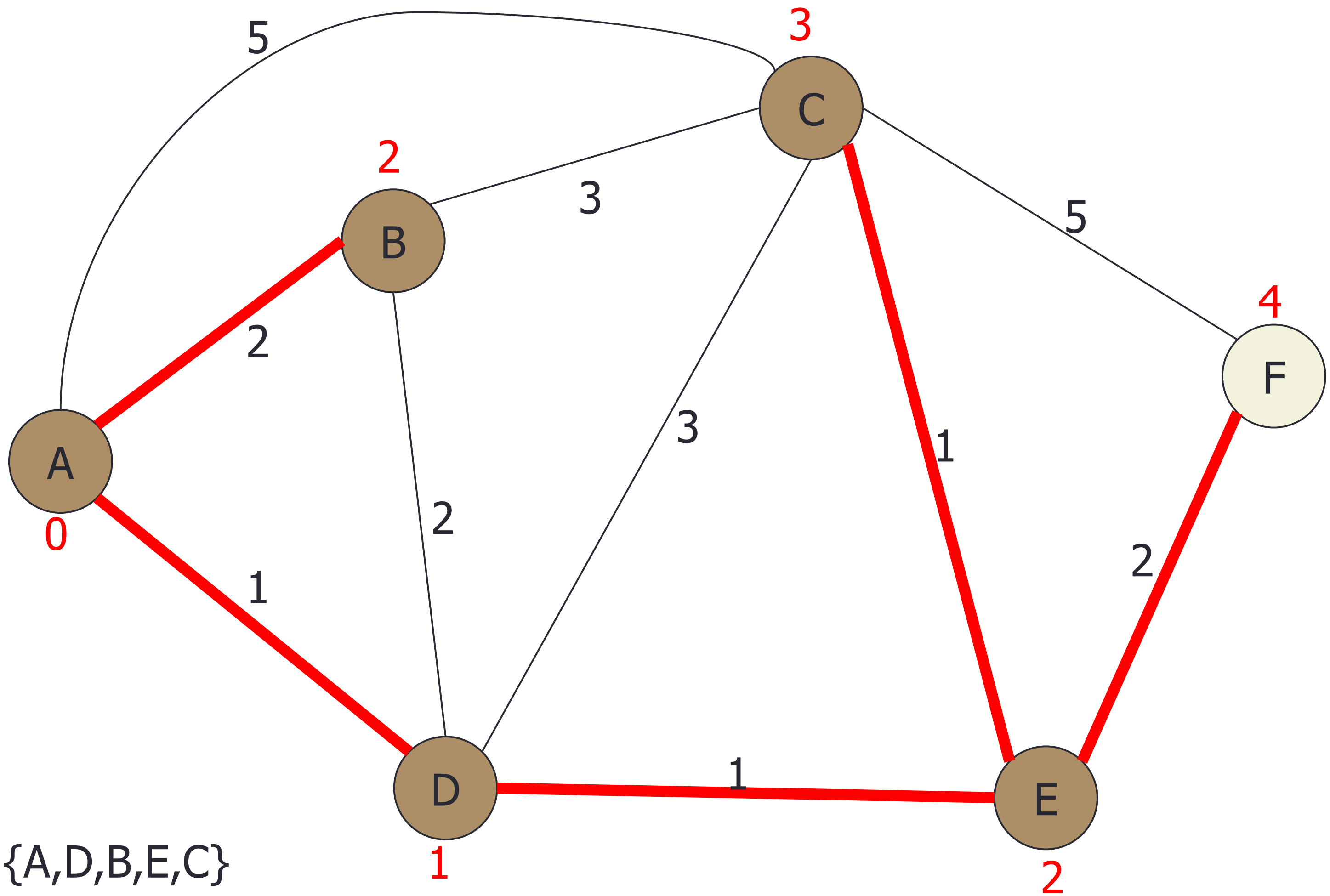
T={A,D,B}

Dijkstra’s Algorithm



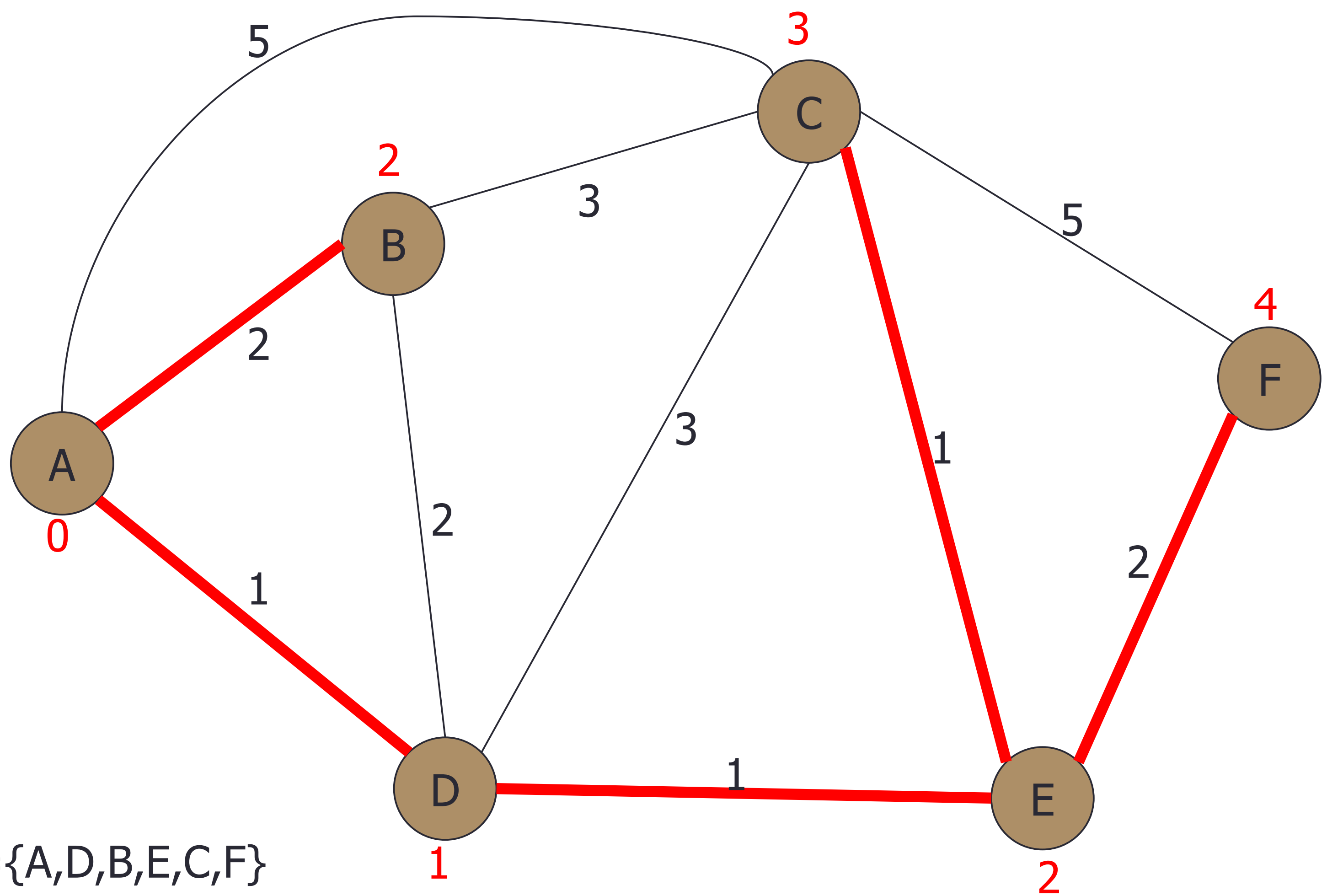
T={A,D,B,E}

Dijkstra's Algorithm



T={A,D,B,E,C}

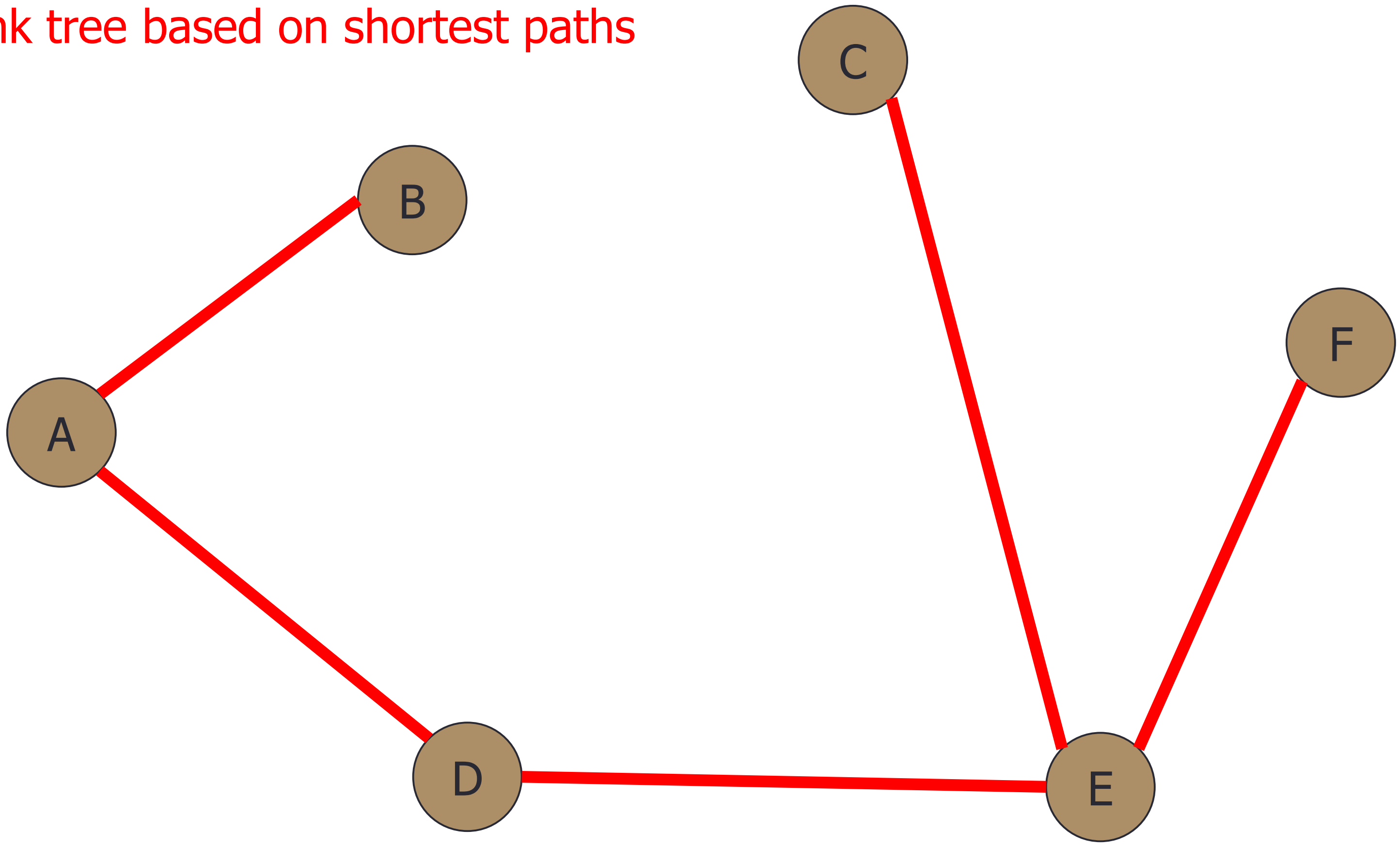
Dijkstra's Algorithm



T={A,D,B,E,C,F}

Dijkstra's Algorithm

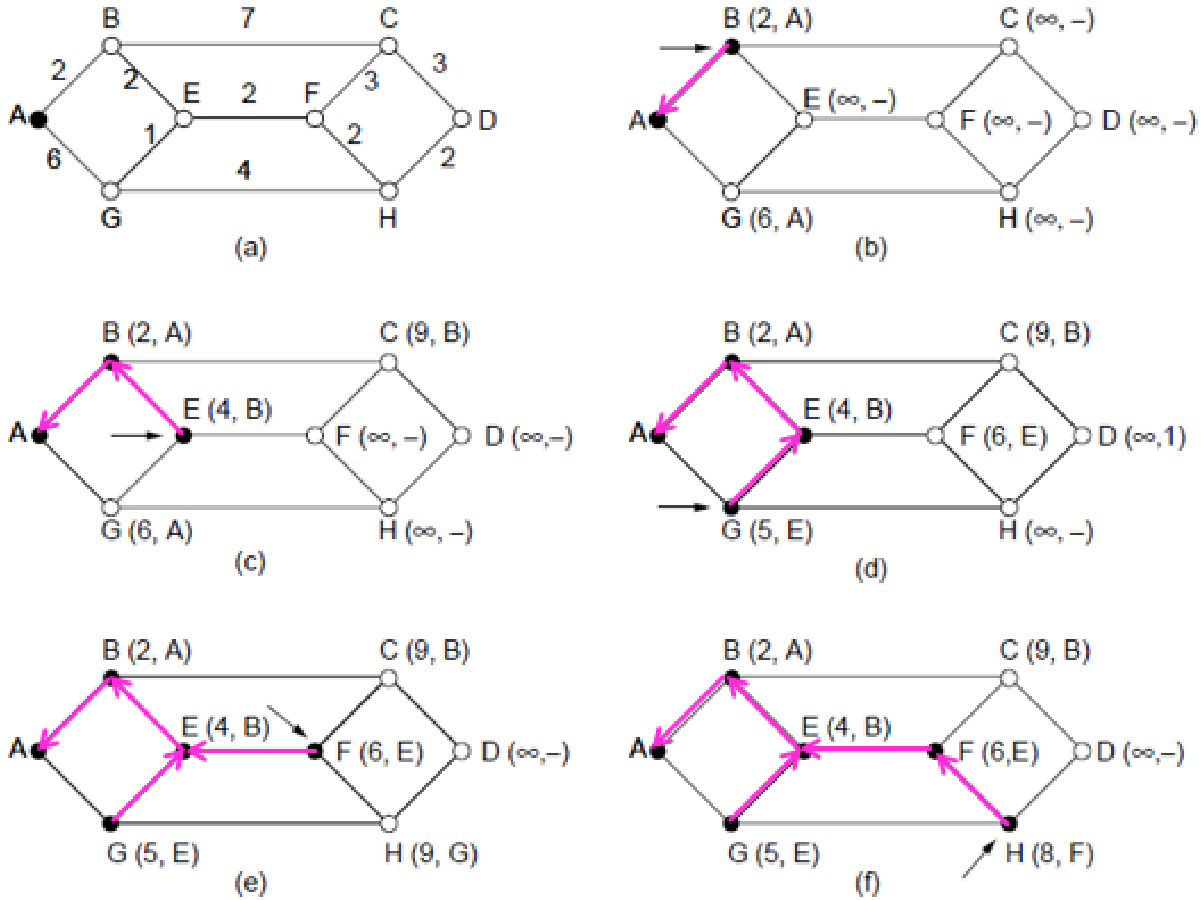
Sink tree based on shortest paths



Dijkstra's Algorithm

#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2	{A,**D**}	2, A	4, D	-	2, D	∞, -
3	{A,D,**B**}	-	4, D	-	2, D	∞, -
4	{A,D,B,**E**}	-	3, E	-	-	4, E
5	{A,D,B,E,**C**}	-	-	-	-	4, E
6	{A,D,B,E,C,**F**}	-	-	-	-	-

Dijkstra's Algorithm



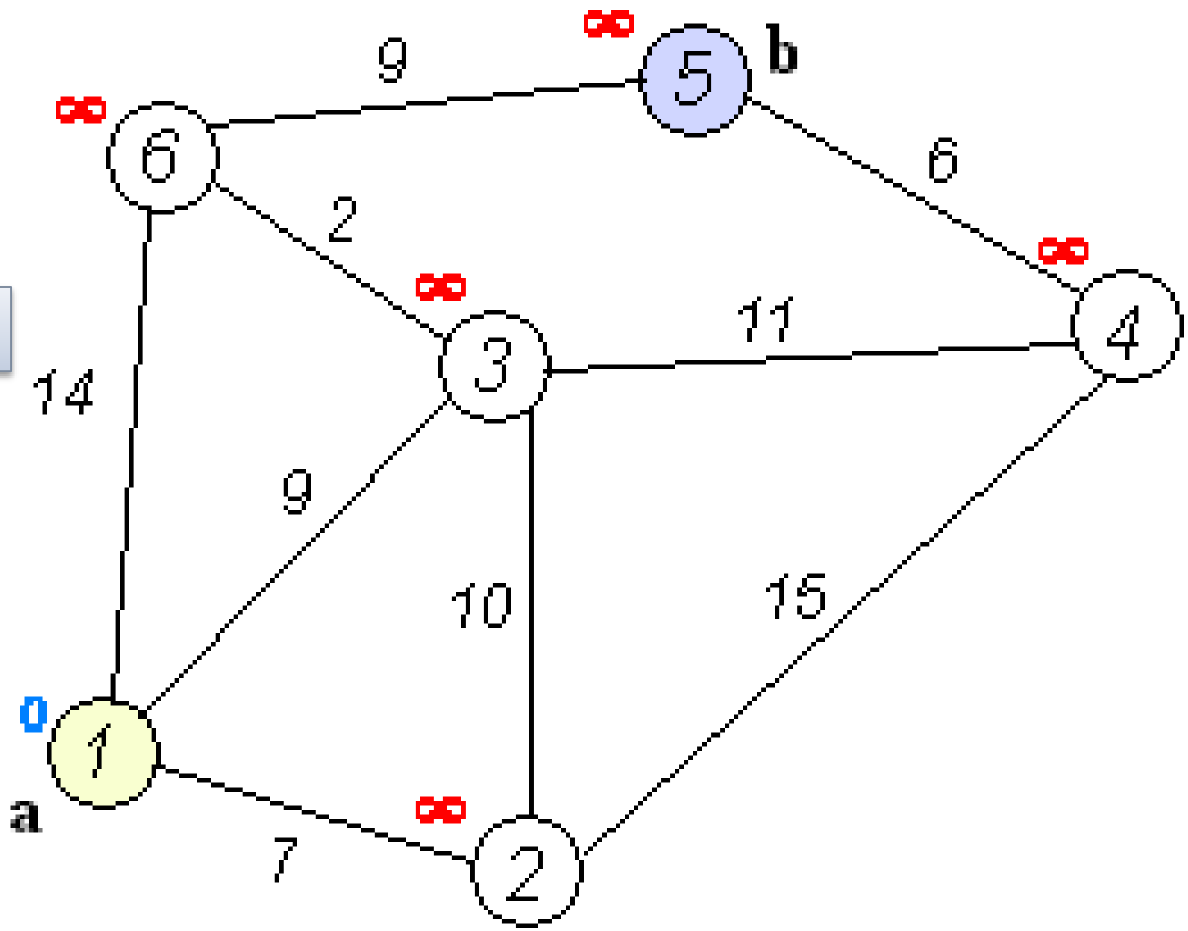
A network and first five steps in computing the shortest paths from A to D. Pink arrows show the sink tree so far.

Shortest Path Algorithm

To Round	A	B	C	D	E	F	G	H
1	0, -	∞, -	∞, -	∞, -	∞, -	∞, -	∞, -	∞, -
2		2, A	∞, -	∞, -	∞, -	∞, -	6, A	∞, -
3			9, B	∞, -	4, B	∞, -	6, A	∞, -
4			9, B	∞, -		6, E	5, E	∞, -
5			9, B	∞, -		6, E		9, G
6			9, B	∞, -				8, F
7			9, B	10, H				
8				10, H				

Dijkstra's Algorithm

Dijktra Animation



Source: Wikipedia
http://en.wikipedia.org/wiki/File:Dijkstra_Animation.gif

Flooding

- Send every packet to all lines except the one it arrived on
- Large number of duplicate packets
- Should use **counter** to prevent infinite duplicates
- Should use sequence numbers to identify duplicates
- Will always find shortest path

Flooding

- Military applications
- Distributed database systems
- Wireless stations use it by nature
- Metric for other algorithms (e.g. delay)

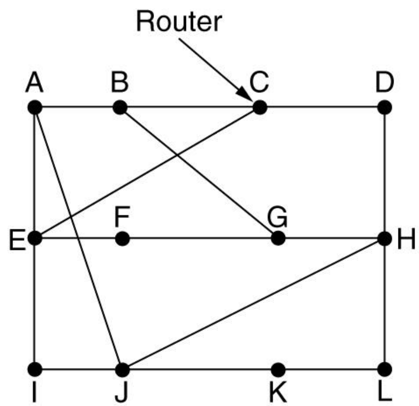
Distance Vector Routing

- Each router maintains a table containing
  - destination
  - best known distance to that destination
  - line to use to get there
- Uses Bellman-Ford algorithm
- Used in ARPANET and now used in RIP
- Distance can be any metric: delay, hop count, queue length, etc.

Distance Vector Routing

- Each router exchange with its neighbors list of delays to each destination
- Router X estimates delay to router Z
  - Router Y is a neighbor to router X
  - $D(X,Z) = D(X,Y) + D(Y,Z)$

Distance Vector Routing



(a)

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

(b)

(a) A subnet (b) Input from A, I, H, K, and the new routing table for J

The Count-to-Infinity Problem

Failures can cause DV to "count to infinity" while seeking a path to an unreachable node

A	B	C	D	E	
•	•	•	•	•	
	•	•	•	•	Initially
	1	•	•	•	After 1 exchange
	1	2	•	•	After 2 exchanges
	1	2	3	•	After 3 exchanges
	1	2	3	4	After 4 exchanges

Good news of a path to *A* spreads quickly

A		B	C	D	E	
•	X	•	•	•	•	
		1	2	3	4	Initially
		3	2	3	4	After 1 exchange
		3	4	3	4	After 2 exchanges
		5	4	5	4	After 3 exchanges
		5	6	5	6	After 4 exchanges
		7	6	7	6	After 5 exchanges
		7	8	7	8	After 6 exchanges
			⋮			
		•	•	•	•	

Bad news of no path to *A* is learned slowly

Link State Routing

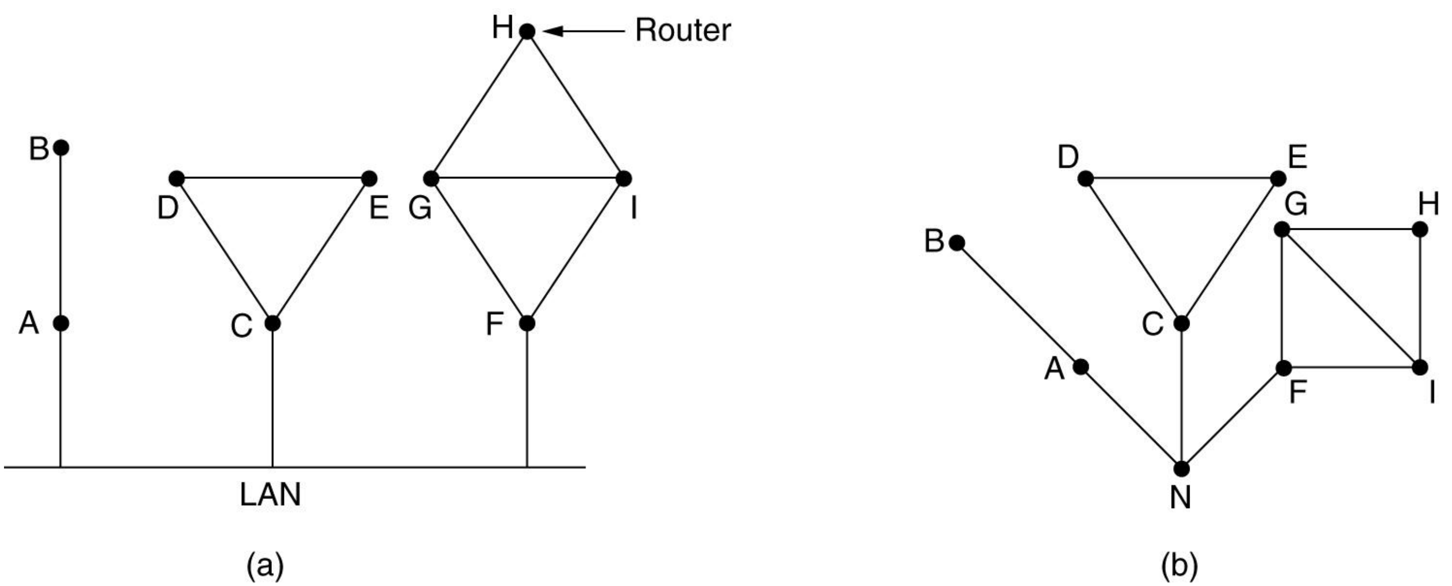
- Each router construct the topology of the entire configuration and calculates the shortest path to each destination network

Link State Routing

- Discover neighbors and learn their network addresses
- Measure delay or cost to each of the neighbors
- Construct a packet telling all what has just learned
- Send this packet to all other routers
- Compute shortest path to every other router (using Dijkstra's algorithm)

Learning about Neighbors

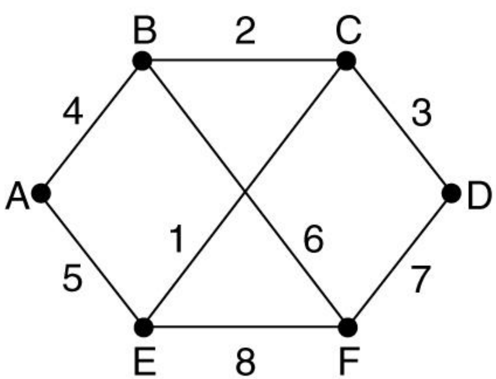
- Send HELLO packet on point-to-point lines
- If routers are connected to a LAN, the LAN can be represented as a node



Measuring Line Cost

- Send ECHO packet
- Wait for response
- Measure round-trip-time
- To take load into account: start timer when packet is queued
- To ignore the load: start timer when packet reaches the front of the queue

Building Link State Packets



(a)

Link State Packets

A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

(b)

Distributing Link State Packets

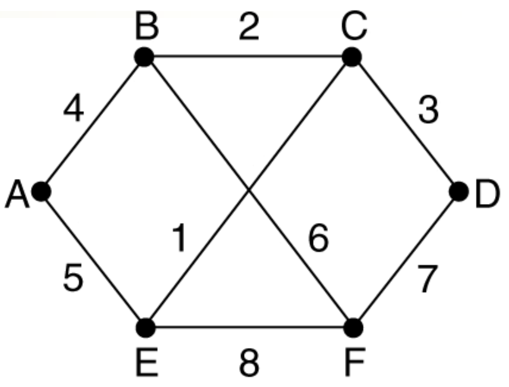
- Use flooding
- Packet contains sequence number
- When packet is received
  - If new, forward to all except coming from
  - If duplicate, discard
  - If old, rejected

Distributing Link State Packets

Problems

- Sequence number wrap around
  - Use 32-bit sequence numbers
- Router crashes, seq. no. starts over
- Seq. no. corrupted: 65540 instead of 4
  - Include age, decremented once per second

Distributing Link State Packets



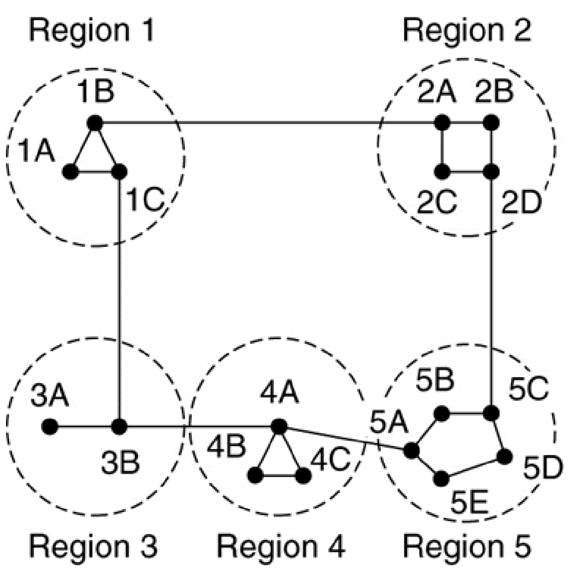
Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

The packet buffer for router B

Hierarchical Routing

- Routing tables grow with network size
- More router memory
- More CPU time to scan them
- More bandwidth to send updates
- For large networks, better to do routing hierarchically
- Hierarchy can be in multiple levels
  - regions
  - clusters
  - zones
  - groups …

Hierarchical Routing



Full routing table has 17 entries

Full table for 1A

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)

Hierarchical table for 1A

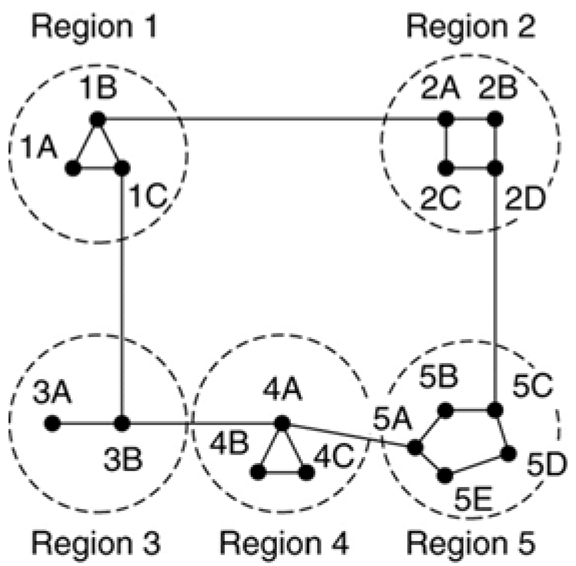
Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Hierarchical routing table has 7 entries

Hierarchical Routing

- The gain in space is not free
- Increased path length for some hosts
- Example
  - best route from 1A to 5C is via R2
  - with hierarchical routing all traffic to R5 is via R3
  - because it is better for most dests in R5



Broadcast Routing

- Send message to many or all other hosts
- e.g. distributing information
- Send one packet to each destination?
  - wasteful of bandwidth
  - require having complete list of destinations
- Flooding
  - generates too many packets, waste bandwidth

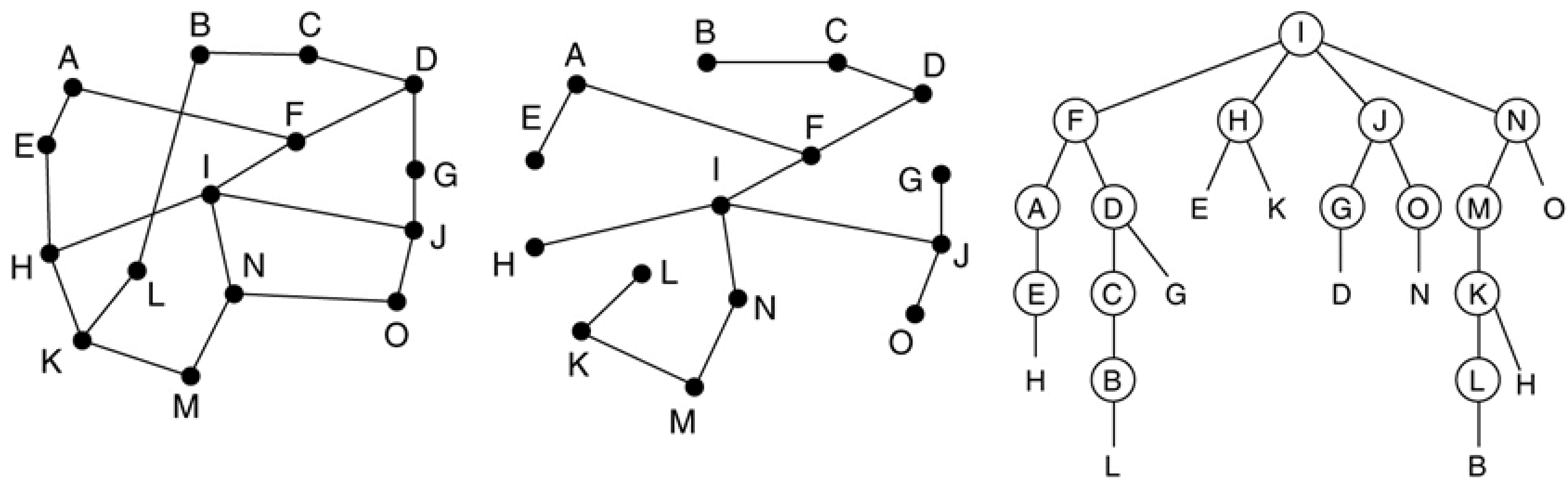
Broadcast Routing

- Multi-destination routing
  - packet contain list of destinations
  - or bit map indicating desired destinations
- Spanning tree
  - use sink tree for router initiating broadcast
  - includes all routers but contains no loops
  - copy packet to all spanning tree lines (-arrived)
  - routers need to know spanning tree of source
  - works with link state, not distance vector

Broadcast Routing

- Reverse path routing
  - approximate without knowing spanning tree
  - if packet arrive on link used to send *to* source
  - high chance it followed best path *from* source
  - thus, forward to all except incoming line
  - from other link? duplicate, discard

Example: Reverse Path Routing



Subnet (a) Sink tree for (I) (b) Tree built by reverse path forwarding (c)

Example: Reverse-Path Routing

- First hop: I sends 4 packets to F, H, J, and N
  - each of these arrives on the preferred path to I
  - so, indicated by circle; forwarded
- Second hop: 8 packets are generated
  - all 8 arrive at previously unvisited routers
  - 5 of these arrive along preferred line
- Third hop: 6 packets generated
  - only 3 arrive at preferred path (C, E, K)
  - others are duplicates (copies); discarded

Example: Reverse-Path Routing

- Fourth hop: 4 packets generated
  - 2 arrive at preferred lines (B, L); forwarded
  - 2 are duplicates; discarded
- Fifth hop: 2 packets generated
  - 2 are duplicates; discarded
  - no more forwarded packets
- Total: 24 packets, 5 hops
- If sink tree was followed exactly
  - only 14 packets, 4 hops required

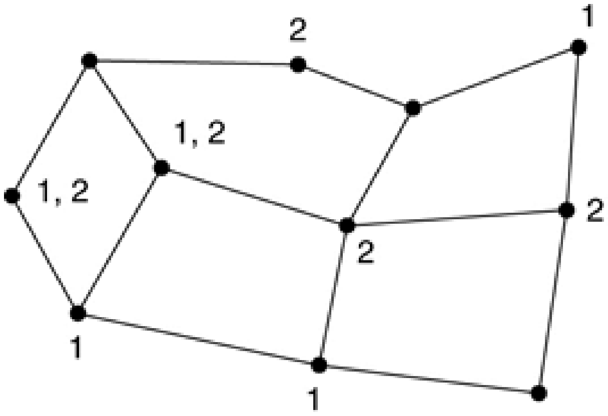
Advantages of Reverse-Path Forwarding

- Reasonably efficient and easy to implement
- Routers don’t need to know spanning trees
- No overhead of destination lists in packets
  - as in multi-destination addressing
- No special mechanism to stop the process
  - as in flooding

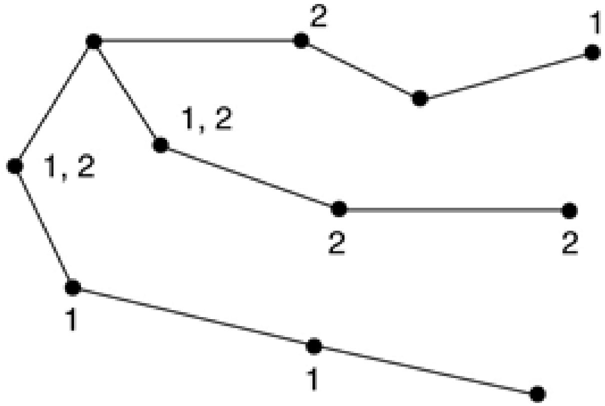
Multicast Routing

- Multicasting
  - sending message to a group of nodes
  - routing algorithm called multicast routing
- Why multicasting?
  - distributed processing
  - broadcasting is inefficient, sometimes insecure
- Require group management
  - create, destroy groups
  - processes to join, leave groups

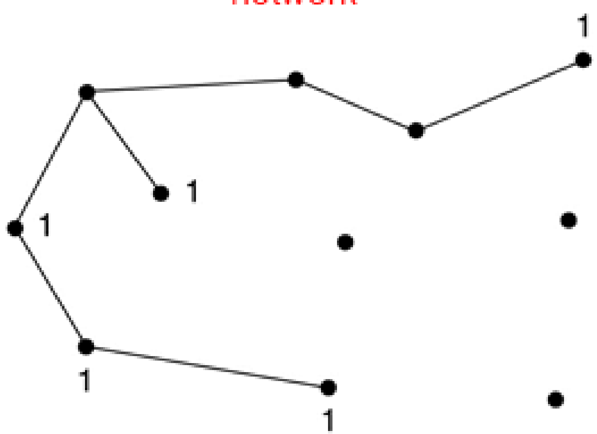
Multicast Routing



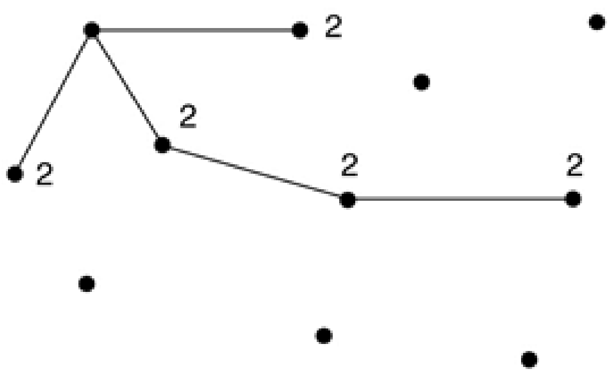
(a)
network



(b)
spanning tree for leftmost node



(c)
multicast tree for group 1



(d)
multicast tree for group 2

External References

- Data and Computer Communications, Stallings, 8/E
  - Dijkstra and Bellman-Ford algorithm descriptions and examples