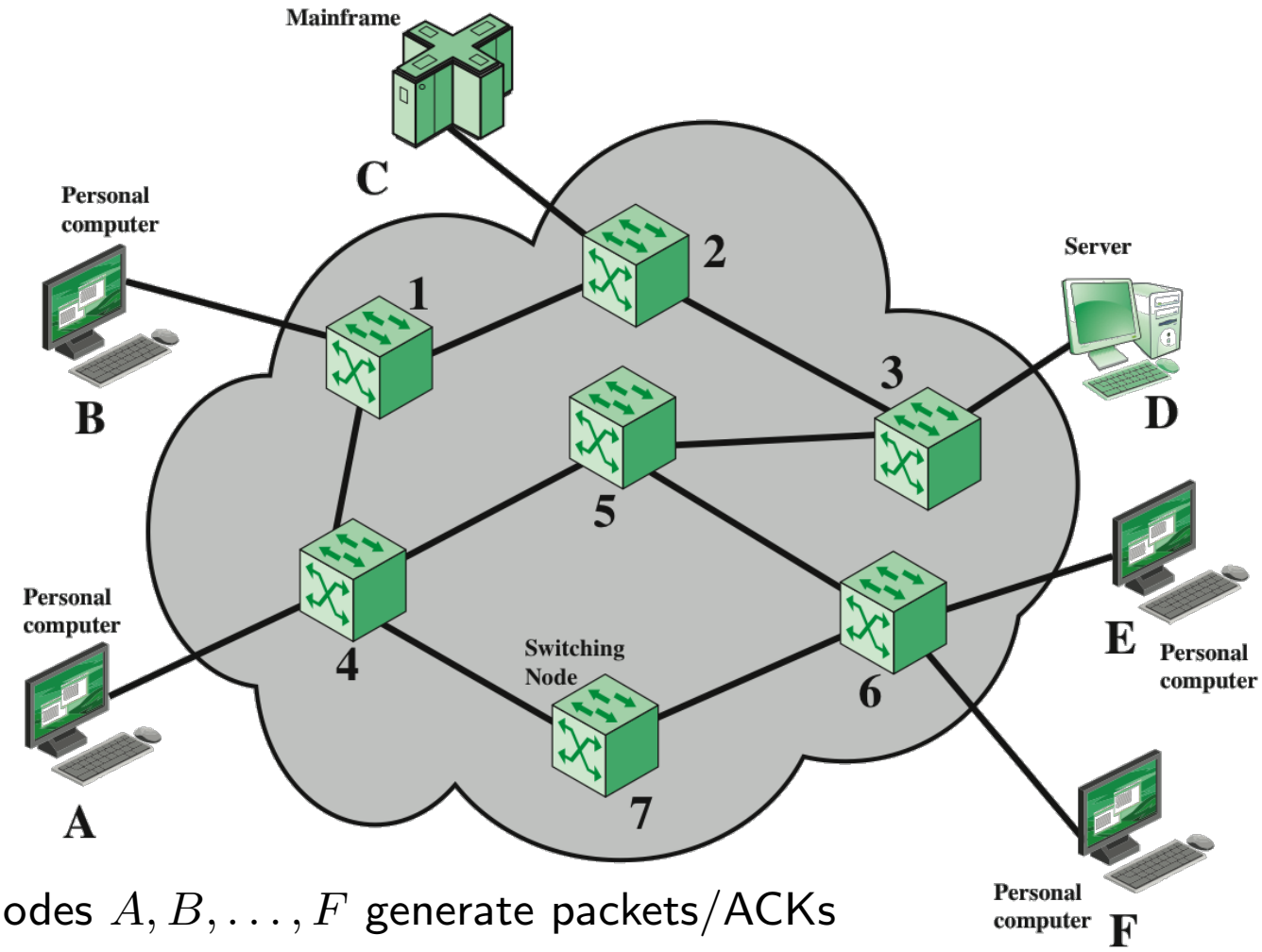
Communication and Networking
Routing Basics

D. Richard Brown III

(selected figures from Stallings Data and Computer Communications 10th edition)

Routing



- Edge nodes $A, B, \ldots, F$ generate packets/ACKs
- Network nodes $1, \ldots, 7$ route packets/ACKs
- Goal is to correctly/efficiently transfer packets between edge nodes

Terminology and Notation

- Edge nodes $A, B, \ldots, F$
- Network nodes $1, \ldots, 7$
- Hops: number of links between network nodes
  - Usually edge$\leftrightarrow$network links are not counted as hops
  - $A \leftrightarrow 4$ is not a hop
  - $4 \leftrightarrow 5$ is a hop
- N_i is the set of network nodes directly connected to network node i
  - $N_1 = \{2, 4\}$
  - $N_5 = \{3, 6, 4\}$
  - $\ldots$
- E_i the set of edge nodes directly connected to network node i
  - $E_1 = \{B\}$
  - $E_5 = \emptyset$
  - $E_6 = \{E, F\}$
  - $\ldots$

Elements of Routing for Packet Switched Networks

We want our routing strategy to be simple, robust, fair, efficient, ...
These criteria must be traded off, however.

Elements of a routing strategy:

Performance Criteria	**Network Information Source**
Number of hops	None
Cost	Local
Delay	Adjacent node
Throughput	Nodes along route
	All nodes
Decision Time	
Packet (datagram)	**Network Information Update Timing**
Session (virtual circuit)	Continuous
	Periodic
Decision Place	Major load change
Each node (distributed)	Topology change
Central node (centralized)	
Originating node (source)	

Routing Strategies

We will discuss:

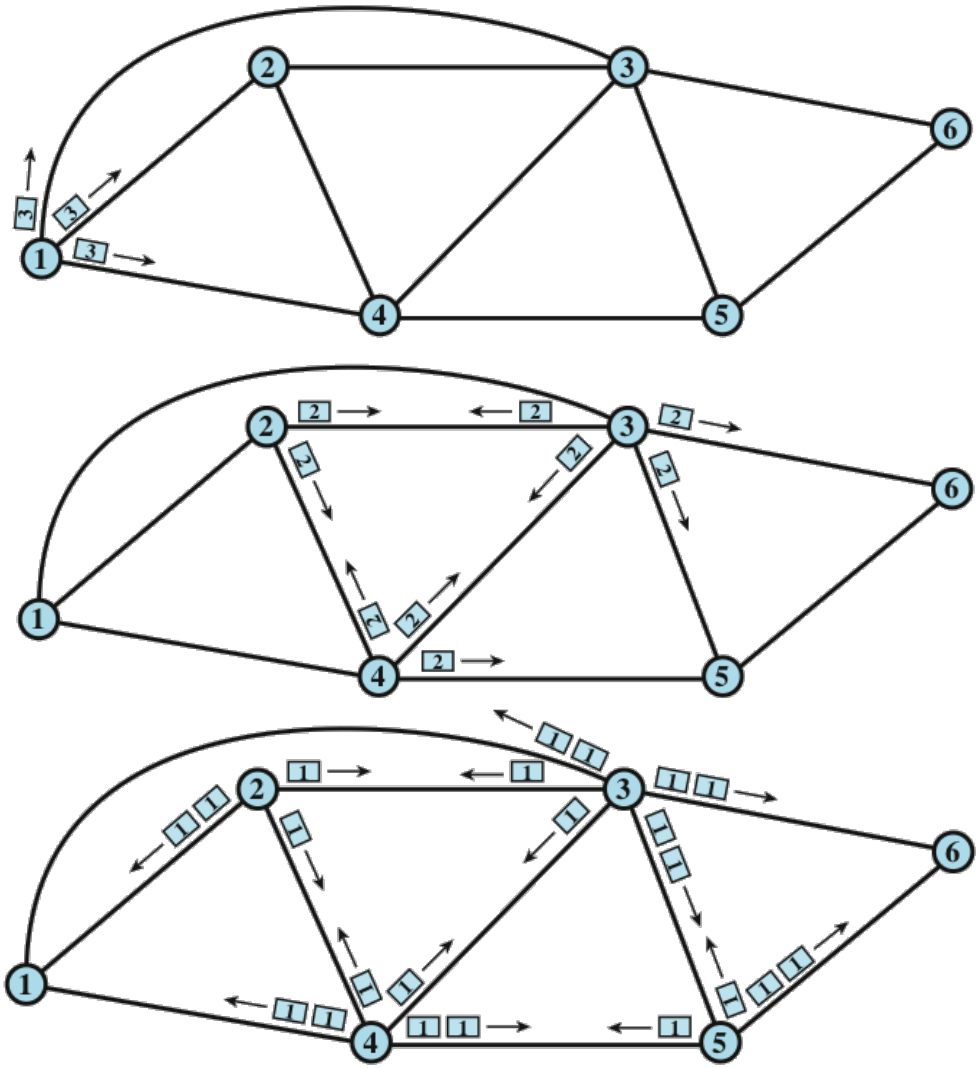
1. Flooding
2. Random routing
3. Fixed routing
4. Adaptive routing

Each has various tradeoffs among the performance criteria.

Flooding

Basic idea:

1. Network node i receives a packet from edge/network node j
2. Network node i checks if destination is in E_i
   - Yes? Packet is forwarded to destination node and done
   - No? Packet is forwarded to **all** nodes in N_i except node j (this is the node that sent the packet to node i)



Flooding Advantages and Disadvantages

Advantages:

- Guaranteed minimum delay delivery!
- Robust since packet travels all routes
- Very simple: network nodes don't need to know anything except N_i and E_i
- Could be used to set up a virtual circuit (call setup)
- Might be useful for broadcast messages

Problems/Disadvantages:

- Packets never die
  - Fix 1: Have nodes recognize and not forward duplicate packets
  - Fix 2: Add maximum hop count to each packet. Each network node decrements the maximum hop counter until it is zero (like TTL) and then the packet is dropped
- Destination will likely receive several copies of the packet
  - Fix: use a unique sequence number and discard duplicates
- Very inefficient due to large amounts of redundant traffic.

Random Forwarding

Basic idea:

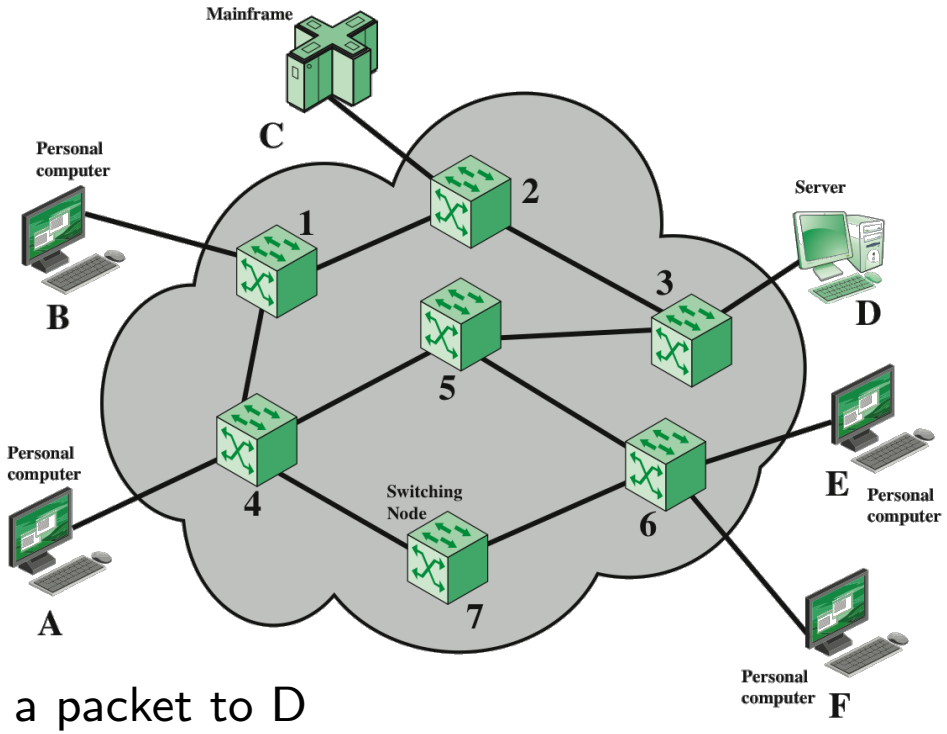
1. Network node i receives a packet from edge/network node j
2. Network node i checks if destination is in E_i
   - Yes? Packet is forwarded to destination node and done
   - No? Packet is forwarded to **one randomly chosen node** in N_i except node j (this is the node that sent the packet to node i)

Note that the forwarding probabilities can be non-uniform.

Advantages/Disadvantages/Problems:

1. Very simple: network nodes don't need to know anything except N_i and E_i
2. Potentially robust
3. Much less redundant traffic than flooding
4. Better fairness than flooding
5. Packet may take a long time to deliver
6. Packet may travel more hops than necessary

Random Forwarding Example

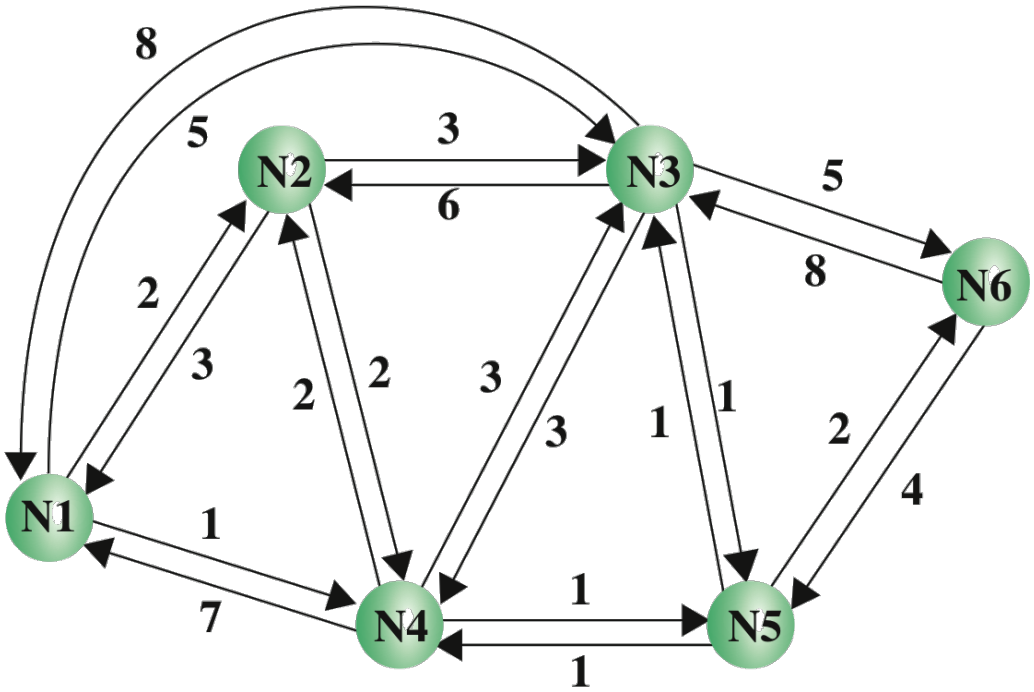


- A wants to send a packet to D
- $A \rightarrow 4$
- 4 randomly chooses between 1, 5, and 7: $4 \rightarrow 7$
- 7 can only forward to 6: $7 \rightarrow 6$
- 6 can only forward to 5: $6 \rightarrow 5$
- 5 randomly chooses between 3 and 4: $5 \rightarrow 4$
- ...

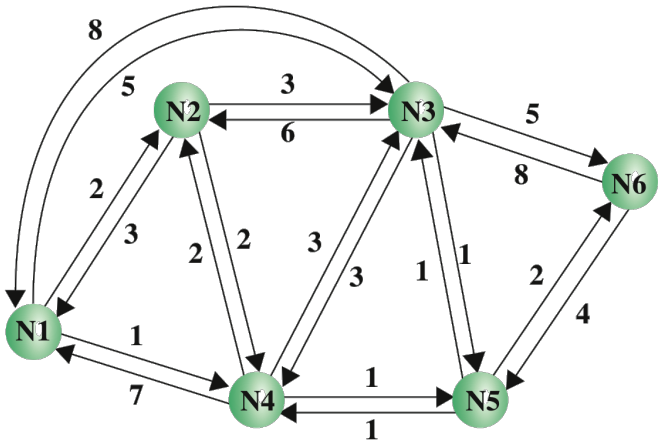
Fixed Routing

Basic idea:

- Each link is assigned a "cost", typically proportional to delay and inversely proportional to throughput
- Prior to forwarding any packets, the network builds a table of optimum (lowest cost) routes between each pair of network nodes
- The lowest cost route may not be the same as the minimum hop route
- These routes are permanent



Fixed Routing: Routing Table Example



CENTRAL ROUTING DIRECTORY

To Node \ From Node	1	2	3	4	5	6
1	—	1	5	2	4	5
2	2	—	5	2	4	5
3	4	3	—	5	3	5
4	4	4	5	—	4	5
5	4	4	5	5	—	5
6	4	4	5	5	6	—

Node 1 Directory

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Directory

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Directory

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Node 4 Directory

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Node 5 Directory

Destination	Next Node
1	4
2	4
3	3
4	4
6	6

Node 6 Directory

Destination	Next Node
1	5
2	5
3	5
4	5
5	5

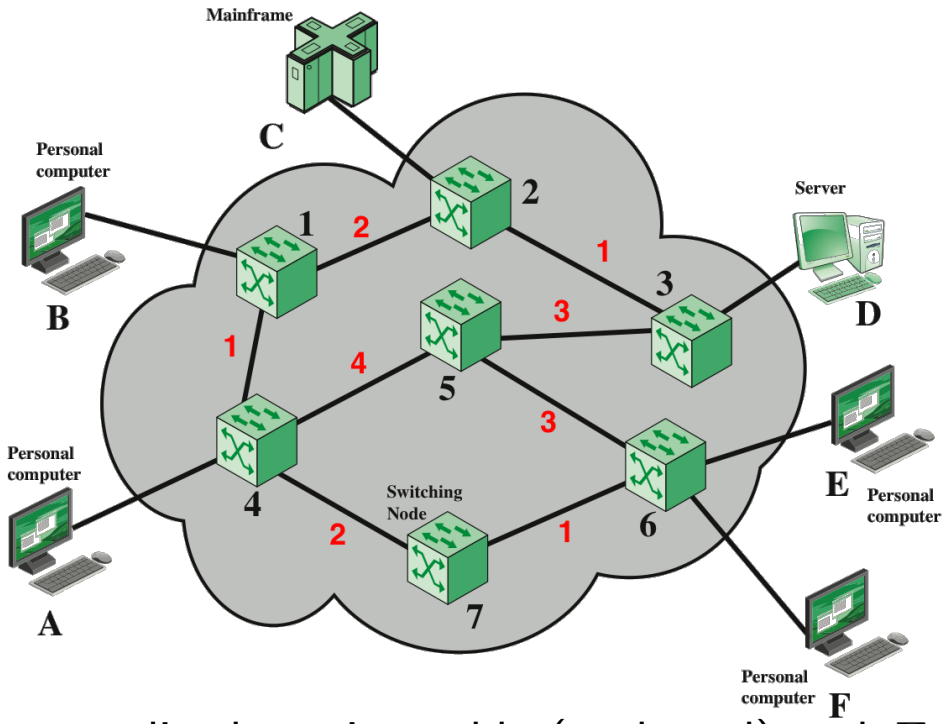
Fixed Routing: Key Concepts

- Each network node $k = 1, \ldots, N$ only keeps the relevant part of the central routing table in its memory. This can be thought of as a "next node" vector indexed by destination:

$$S_k = \begin{bmatrix} s_{k,1} \\ \vdots \\ s_{k,N} \end{bmatrix} = \begin{bmatrix} \text{next node if destination is } 1 \\ \vdots \\ \text{next node if destination is } N \end{bmatrix}$$

- Given a destination node, each network node knows the next node for the minimum cost route
- Network nodes may also need to know all E_i to determine which network node is the correct destination for a given edge node

Fixed Routing Example (Symmetric Link Costs)



- First build 7×7 centralized routing table (on board) and $E_1, \ldots, E_7$
- A wants to send a packet to D
- $A \rightarrow 4$ with destination address D
- 4 knows $D \in E_3$, looks at routing table, and routes $4 \rightarrow 1$
- 1 knows $D \in E_3$, looks at routing table, and routes $1 \rightarrow 2$
- 2 knows $D \in E_3$, looks at routing table, and routes $2 \rightarrow 3$
- 3 knows D is a connected edge node and delivers the packet

Fixed Routing: Advantages/Disadvantages

Advantages:

- If cost=delay, then messages are delivered with minimum delay
- No redundant traffic
- Fairly simple (not as simple as flooding or random forwarding)

Disadvantages/Problems:

- Establishes virtual circuits:
  - less robust to network problems
  - may not react to congestion
- Overhead from setting up routing tables
- How to find the minimum cost routes? (Section 19.4: Dijkstra's Algorithm, Bellman-Ford Algorithm)
- More knowledge needed by network nodes than flooding/random forwarding
- Lack of flexibility

Adaptive Routing

Basic idea:

- Routing decisions are dynamic and are based on the current network conditions
- Routes updated when network state changes, e.g.,
  - Node failure detected
  - New nodes added to the network
  - Congestion detected
- Used in almost all packet switching networks
- Can be very complex to implement

Three classes of adaptive routing schemes:

1. Use only local information (isolated adaptive routing)
2. Use information from adjacent nodes
   - Distributed or centralized
3. Use information from all other nodes
   - Distributed or centralized

Adaptive Routing Example: Distance Vector Routing

Used in first-generation ARPANET. Basic idea:

- N is the number of network nodes
- Like fixed routing, each node $k = 1, \ldots, N$ has a "next node" vector indexed by destination

$$S_k[n] = \begin{bmatrix} s_{k,1}[n] \\ \vdots \\ s_{k,N}[n] \end{bmatrix} = \begin{bmatrix} \text{next node if destination is } 1 \\ \vdots \\ \text{next node if destination is } N \end{bmatrix}$$

- Each network node $k = 1, \ldots, N$ also has a "distance vector"

$$D_k[n] = \begin{bmatrix} d_{k,1}[n] \\ \vdots \\ d_{k,N}[n] \end{bmatrix} = \begin{bmatrix} \text{least known delay to node } 1 \\ \vdots \\ \text{least known delay to node } N \end{bmatrix}.$$

- Periodically, nodes exchange distance vectors with neighbors
- Each node $k = 1, \ldots, N$ then updates its "next node" and distance vectors

Distance Vector Routing: How to Update S_k[n] and $D_k[n]$

Recall N_k is the set of neighbors for network node k.

After the distance vector exchange, node k has $D_i[n]$ from all $i \in N_k$.

Node k then updates all of its least known delays and next nodes. For $j = 1, \ldots N$ but $j \neq k$ we compute

$$d_{k,j}[n+1] = \min_{i \in N_k} \left(d_{k,i}[n] + d_{i,j}[n] \right)$$

$$s_{k,j}[n] = \arg\min_{i \in N_k} \left(d_{k,i}[n] + d_{i,j}[n] \right)$$

Intuition:

- $d_{k,i}[n]$ is the least known delay $k \rightarrow i$ where i is a neighbor of k.
- $d_{i,j}[n]$ is the least known delay $i \rightarrow j$.
- Node k searches through all of its neighbors ($i \in N_k$) to find the new least known delay to each destination node $j = 1, \ldots, N$.

Distance Vector Routing Example (part 1 of 2)

Suppose $N = 6$, $k = 1$, and $N_k = \{2, 3, 4\}$. The current state at node k is

$$S_k[n] = \begin{bmatrix} \text{—} \\ 2 \\ 3 \\ 4 \\ 3 \\ 3 \end{bmatrix} \text{ and } D_k[n] = \begin{bmatrix} 0 \\ 0.02 \\ 0.05 \\ 0.01 \\ 0.06 \\ 0.08 \end{bmatrix}$$

After the distance vector exchange, node k receives the distance vectors

$$D_2[n] = \begin{bmatrix} 0.03 \\ 0 \\ 0.03 \\ 0.02 \\ 0.03 \\ 0.05 \end{bmatrix} \text{ and } D_3[n] = \begin{bmatrix} 0.07 \\ 0.04 \\ 0 \\ 0.02 \\ 0.01 \\ 0.03 \end{bmatrix} \text{ and } D_4[n] = \begin{bmatrix} 0.05 \\ 0.02 \\ 0.02 \\ 0 \\ 0.01 \\ 0.03 \end{bmatrix}$$

Node k must now update its distance vector and next node vector...

Distance Vector Routing Example (part 2 of 2)

Updating elements of the distance and next node vectors at network node $k = 1$:

$$
\begin{aligned}
&j = 1: \text{ skip} \\
&j = 2: d_{1,2}[n+1] = \min_{i \in \{2,3,4\}} (d_{1,i}[n] + d_{i,2}[n]) = 0.02 \quad \Rightarrow s_{1,2}[n+1] = 2 \\
&j = 3: d_{1,3}[n+1] = \min_{i \in \{2,3,4\}} (d_{1,i}[n] + d_{i,3}[n]) = 0.03 \quad \Rightarrow s_{1,3}[n+1] = 4 \\
&j = 4: d_{1,4}[n+1] = \min_{i \in \{2,3,4\}} (d_{1,i}[n] + d_{i,4}[n]) = 0.01 \quad \Rightarrow s_{1,3}[n+1] = 4 \\
&j = 5: d_{1,5}[n+1] = \min_{i \in \{2,3,4\}} (d_{1,i}[n] + d_{i,5}[n]) = 0.02 \quad \Rightarrow s_{1,3}[n+1] = 4 \\
&j = 6: d_{1,6}[n+1] = \min_{i \in \{2,3,4\}} (d_{1,i}[n] + d_{i,6}[n]) = 0.04 \quad \Rightarrow s_{1,3}[n+1] = 4
\end{aligned}
$$

These values then are used to populate the vectors $D_1[n+1]$ and $S_1[n+1]$.

Note this procedure is performed at all network nodes $k = 1, \ldots, N$.

Final Remarks

- Four types of routing covered:
  1. Flooding
  2. Random forwarding
  3. Fixed routing
  4. Adaptive routing
- Adaptive routing is the most common method (but can be very complex)
  - Distance vector routing
  - Link-state routing
  - ...
- Exterior/interior routing protocols used in large-scale networks, e.g., the border gateway protocol (BGP) and open shortest path first (OSPF) (see section 19.3)
- Efficient algorithms for finding least-cost routes (see section 19.4):
  - Dijkstra's algorithm
  - Bellman-Ford algorithm