Topics for This Week

• Routing Algorithms/Protocols
  – Link state routing
  – Distance vector routing

• Readings
  – Sections 5.2.1-5.2.5
Routing Table: Issues

- How are routing tables determined?
- Who determines table entries?
- What info used in determining table entries?
- When do routing table entries change?
- Where is routing info stored?
- How to control table size?
- Answer these and we are done!
Routing Algorithms/Protocols: Issues

• Route selection may depend on different criteria
  – Performance: choose route with smallest delay
  – Policy: choose a route that doesn’t cross .gov network

• Adapt to changes in topology or traffic
  – Self-healing: little or no human intervention

• Scalability
  – Must be able to support large number of hosts, routers
Hop-by-Hop Routing vs Source Routing

• Hop-by-hop routing
  – Each packet contains destination address
  – Each router chooses next-hop to destination
  – Example: IP

• Source routing
  – Sender selects the path to destination precisely
  – Routers forward packet to next-hop as specified
  – Example: ATM, MPLS. IP’s loose/strict source route

• Pros and cons of these two approaches?
Distributed Routing Algorithms

• Routers cooperate using a distributed protocol
  – To create mutually consistent routing tables
• Two standard distributed routing algorithms
  – Link state routing
  – Distance vector routing
Link State vs Distance Vector

• Both assume that
  – The address of each neighbor is known
  – The cost of reaching each neighbor is known

• Both find global information
  – By exchanging routing info among neighbors

• Differ in info exchanged and route computation
  – LS: tells every other node its distance to neighbors
  – DV: tells neighbors its distance to every other node
Link State Algorithm

• Basic idea: Distribute to all routers
  – Topology of the network
    • Cost of each link in the network
• Each router independently computes optimal paths
  – From itself to every destination
  – Routes are guaranteed to be loop free if
    • Each router sees the same cost for each link
    • Uses the same algorithm to compute the best path
Topology Dissemination

- Each router creates a set of link state packets
  - Describing its links to neighbors
  - LSP contains
    - Router id, neighbor’s id, and cost to its neighbor
- Copies of LSPs are distributed to all routers
  - Using controlled flooding
- Each router maintains a topology database
  - Database containing all LSPs
Dijkstra’s Algorithm

• Given the network topology
  – How to compute shortest path to each destination?

• Some notation
  – X: source node
  – N: set of nodes to which shortest paths are known so far
    • N is initially empty
  – D(V): cost of known shortest path from source X
  – C(U,V): cost of link U to V
    • C(U,V) = ∞ if not neighbors
Algorithm (at Node X)

- **Initialization**
  - $N = \{X\}$
  - For all nodes $V$
    - If $V$ adjacent to $X$, $D(V) = C(X,V)$ else $D(V) = \infty$

- **Loop**
  - Find $U$ not in $N$ such that $D(U)$ is smallest
  - Add $U$ into set $N$
  - Update $D(V)$ for all $V$ not in $N$
    - $D(V) = \min\{D(V), D(U) + C(U,V)\}$
  - Until all nodes in $N$
# Link State Database

<table>
<thead>
<tr>
<th>(c(x,y))</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
<td>∞</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>∞</td>
<td>∞</td>
<td>5</td>
<td>∞</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
# Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>start N</th>
<th>D(B),p(B)</th>
<th>D(C),p(C)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2,A</td>
<td>5,A</td>
<td>1,A</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2,A</td>
<td>4,D</td>
<td>2,D</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2,A</td>
<td>3,E</td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3,E</td>
<td></td>
<td>4,E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td></td>
<td></td>
<td>4,E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Routing Table Computation

<table>
<thead>
<tr>
<th>dest</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>D</td>
</tr>
</tbody>
</table>
Distance Vector Routing

• A router tells neighbors its distance to every router
  – Communication between neighbors only
• Based on Bellman-Ford algorithm
  – Computes “shortest paths”
• Each router maintains a distance table
  – A row for each possible destination
  – A column for each neighbor
    • $D^X(Y,Z)$: distance from $X$ to $Y$ via $Z$
• Exchanges distance vector with neighbors
  – Distance vector: current least cost to each destination
Distance Table: Example

cost to destination via

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>
Distance Table to Routing Table

<table>
<thead>
<tr>
<th>Destination</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

- **Distance table**

- **Routing table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Outgoing link to use, cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A,1</td>
</tr>
<tr>
<td>B</td>
<td>D,5</td>
</tr>
<tr>
<td>C</td>
<td>D,4</td>
</tr>
<tr>
<td>D</td>
<td>D,2</td>
</tr>
</tbody>
</table>
Distance Vector Routing Algorithm

iterative:
• continues until no nodes exchange info.
• *self-terminating*: no “signal” to stop

asynchronous:
• nodes need *not* exchange info/iterate in lock step!

distributed:
• each node talks *only* with directly-attached neighbors

Distance Table data structure
• each node has its own
• row for each possible destination
• column for each directly-attached neighbor to node
• example: in node X, for dest. Y via neighbor Z:

\[
D_{X}^{Y,Z} = \text{distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop} \\
= c(X,Z) + \min_{W} \{D_{Z}^{Y,W}\}
\]
Distance Vector Routing: Overview

Iterative, asynchronous:
each iteration caused by:
• local link cost change
• message from neighbor: its least cost path change from neighbor

Distributed:
• each node notifies neighbors *only* when its least cost path to any destination changes
  – neighbors then notify their neighbors if necessary

Each node:

1. *wait* for (change in local link cost or msg from neighbor)
2. *recompute* distance table
3. if least cost path to any dest has changed, *notify* neighbors
### Distance Vector Algorithm: Example

<table>
<thead>
<tr>
<th>dest</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\} \]
\[ = 7 + 1 = 8 \]

<table>
<thead>
<tr>
<th>dest</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\} \]
\[ = 2 + 1 = 3 \]
Distance Vector Algorithm: Example
Convergence of DV Routing

- router detects local link cost change
- updates distance table
- if cost change in least cost path, notify neighbors

“good news travels fast”
Problems with DV Routing

Link cost changes:
- good news travels fast
- bad news travels slow
  - “count to infinity” problem!

algorithm continues on!
Count-to-Infinity Problem
Fixes to Count-to-Infinity Problem

• Split horizon
  – A router never advertises the cost of a destination to a neighbor
    • If this neighbor is the next hop to that destination

• Split horizon with poisonous reverse
  – If X routes traffic to Z via Y, then
    • X tells Y that its distance to Z is infinity
      – Instead of not telling anything at all
  – Accelerates convergence
Split Horizon with Poisoned Reverse

If Z routes through Y to get to X:

- Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z)
Count-to-Infinity Problem

X Y Z

W
Link State vs Distance Vector

- Tells everyone about neighbors
- Controlled flooding to exchange link state
- Dijkstra’s algorithm
- Each router computes its own table
- May have oscillations
- Open Shortest Path First (OSPF)

- Tells neighbors about everyone
- Exchanges distance vectors with neighbors
- Bellman-Ford algorithm
- Each router’s table is used by others
- May have routing loops
- Routing Information Protocol (RIP)