Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 VC and Datagram Networks
4.3 What's Inside a Router
4.4 The Internet (IP) Protocol - IPv4, IPv6
4.5 Routing Algorithms
   - Link state routing
   - Distance vector routing
   - Hierarchical Routing
4.6 Routing in the Internet
4.7 Broadcast and Multicast Routing

No class on 3/24/2005

Link State Routing: Dijkstra's algorithm, discussion

- Each router broadcasts the identities and costs of its attached links to all other routers
- All nodes have an identical and complete view of the network
- Each node runs the LS algorithm and computes the shortest paths
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.
  - self-terminating: no "signal" to stop.

- **Distributed:**
  - each node communicates 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:

\[
d_{X,Y,Z} = \text{distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop} = c(X,Z) + \min_{w} d_{Y,w}
\]

**Distance Table: example**

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

**Distance table gives routing table**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outgoing Link to use, cost:
- A, 1
- B, 5
- C, 4
- D, 4

Distance table → Routing table
Distance Vector Routing: overview

Iterative, asynchronous:
- each local 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:
- wait for (change in local link cost msg from neighbor)
- recompute distance table
- if least cost path to any dest has changed, notify neighbors

Distance Vector Algorithm
(Bellman-Ford):

At all nodes, X:
1. Initialization:
   2. for all adjacent nodes v:
   3. \( D^X(v) = \infty \) (the \(^*\) operator means "for all rows")
   4. \( D^X(x) = c(X,v) \)
   5. for all destinations, y:
   6. send \( \min_{w} D^X(y,w) \) to each neighbor \( w \) over all X's neighbors \(^*\)

Distance Vector Algorithm (cont.):

8 loop
9 wait (until I see a link cost change to neighbor V
10 or until I receive update from neighbor V)
11
12 if \( c(X,V) \) changes by \( d \)
13 (note: \( d \) could be positive or negative \(^*\))
14 for all destinations y: \( D^X(y,V) = D^X(y,V) + d \)
15
16 else if (update received from V wrt destination Y)
17 (shortest path from V to some Y has changed \(^*\))
18 V has sent a new value for its \( \min_{w} D^X(y,w) = \text{newval} \)
19
20 for the single destination y: \( D^X(y,V) = c(X,V) + \text{newval} \)
21
22 else if (update received from V wrt destination Y)
23 if we have a new \( \min_{w} D^X(y,w) \) for any destination Y
24 send new value of \( \min_{w} D^X(y,w) \) to all neighbors
25
26 forever
Distance Vector Algorithm: example

\[ D^X(Y,Z) = c(X,Z) + \min_{w} D^Z(Y,w) \]
\[ = 7 + 1 = 8 \]
\[ D^Z(Y,X) = c(Z,Y) + \min_{w} D^Y(Z,w) \]
\[ = 2 + 1 = 3 \]

Distance Vector: link cost changes

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23, 24)

"good news travels fast"
Distance Vector: link cost changes

Link cost changes:
- Good news travels fast
- Bad news travels slow — "count to infinity" problem!

Distance Vector: use poisoned reverse idea

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)
- Will this completely solve count to infinity problem?

RECALL
Link State Routing: Dijkstra’s algorithm, discussion

- Each router broadcasts the identities and costs of its attached links to all other routers
- All nodes have an identical and complete view of the network
- Each node runs the LS algorithm and computes the shortest paths
Comparison of LS and DV algorithms

Message complexity
- **LS:** with n nodes, E links, O(nE) msgs sent each
- **DV:** exchange between neighbors only
  - convergence time varies

Speed of Convergence
- **LS:** O(n²) algorithm requires O(nE) msgs
  - may have oscillations
- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?
- **LS:**
  - node can advertise incorrect link cost
  - each node computes only its own table
- **DV:**
  - DV node can advertise incorrect path cost
  - each node's table used by others
  - error propagate thru network

Network Layer