The Network Layer: Routing 2: Distance Vector

Recap of Routing So Far

- **Questions:**
  - What is the objective of routing?
  - Does routing occur between hosts or routers?
  - What are differences between centralized (global) and decentralized algorithms?
    - What are examples of each?
    - Amount of information initially
    - How information is shared/spread
    - Synchronous or asynchronous?
      - (see pathologies as well)

Algorithm 2: Distance Vector

Rather than using global information, a distance vector algorithm is:

- **distributed:**
  - each node communicates only with directly-attached neighbors

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

- **asynchronous:**
  - nodes need not exchange information or iterate in lock step!

Overview

- **Routing Algorithms**
  - Link-state – From last week
  - Distance-vector – TODAY
Distance Vector Algorithm

Bellman-Ford Equation, an important relationship among costs of least-cost paths

Define
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]

Then
\[ d_x(y) = \min \{c(x,v) + d_y(v)\} \]

where \( \min \) is taken over all neighbors \( v \) of \( x \)

Distance Vector Routing Algorithm

Distance Table data structure
- each node has its own
  - row for each possible destination
  - column for each directly-attached neighbor
- example: in node \( X \), for destination \( Y \) via neighbor \( Z \):

\[
D^X(Y,Z) = \text{distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop} = c(X,Z) + \min \{D^Z(Y,W)\}
\]

Bellman-Ford Equation

Clearly, \( d_z(z) = \), \( d_y(z) = \), \( d_w(z) = \)

B-F equation says:
\[
d_x(z) = \min \left\{ \begin{array}{c}
c(u,v) + d_y(v) \\
c(u,x) + d_y(z) \\
c(u,w) + d_y(z)
\end{array} \right\} = \min \{ \}
\]

The node that achieves the minimum is the next hop in shortest path \( \rightarrow \) forwarding table

Distance Table: example with complete information

<table>
<thead>
<tr>
<th></th>
<th>D(A,B) = c(E,B) + min {D^B(A,W)}</th>
<th>D(A,D) = c(E,D) + min {D^D(A,W)}</th>
<th>D(C,D) = c(E,D) + min {D^D(C,W)}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>= c(E,B) + min {D^B(A,W)}</td>
<td>= c(E,D) + min {D^D(A,W)}</td>
<td>= c(E,D) + min {D^D(C,W)}</td>
</tr>
</tbody>
</table>

\[ D^E() \]

cost to destination via

destination

distance

2
**Distance table to forwarding 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>7</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>14</td>
<td>8</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

**Distance vector algorithm**

**Basic idea:**
- Each node begins with $D_x(y)$
  - An estimate of the cost of the least-cost path from itself to node $y$, for all nodes in $N$
- Each node periodically sends its own distance vector estimate to neighbors
- When a node $x$ receives new DV estimate from neighbor, it updates its own DV using B-F equation, and sends any update to its neighbors

\[
D\{y\} \leftarrow \min \{c(x,y) + D\{y\} \} \quad \text{for each node } y \in N
\]

- Under normal conditions, the estimate $D\{y\}$ converge to the actual least cost $d\{y\}$

**Distance Vector Algorithm: example for obtaining complete information**

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

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

**Distance Vector Algorithm: obtaining info**
Distance Vector Routing Activity

Comparison of LS and DV algorithms

- Information requirements
- Message complexity
- Convergence time varies
- Robustness: what happens if router malfunctions?
- Oscillations possible?
- Loops possible?

Summary

Forwarding:
- Leads to questions of addressing
  - Assignment of IP addresses
  - NAT, IPv6 ...

Routing:
- Routing objectives
- Routing notation
- Routing classification
- Link state v. Distance Vector
- Hierarchical structure