The Network Layer: Routing & Addressing

Outline

- Network layer services
- Routing algorithms
  - Least cost path computation algorithms
- Hierarchical routing
  - Connecting networks of networks
- IP Internet Protocol
  - Addressing
  - IPv6
- Routing on the Internet
  - Intra-domain routing
  - Inter-domain routing
Routing Algorithms

Taxonomy

- Global or decentralized information?
  
  - Global — all routers maintain the complete graph of the network (topology, costs)
    - “Link state” algorithms
  
  - Decentralized — router knows link costs to physically connected adjacent nodes
    - Run iterative algorithm to exchange information with adjacent nodes
    - “Distance vector” algorithms

Decentralized Routing Algorithms

Distance Vector Routing

- Iterative:
  - Nodes exchange cost information until each node has the current route costs
  - The algorithm is self-terminating — there’s no explicit stopping point

- Asynchronous:
  - Nodes need not exchange information and iterate in lock step
  - Intermediate results may be inconsistent across nodes

- Distributed:
  - Each node communicates only with directly-attached adjacent nodes
  - (But there is no flooding of cost information)
Distance Vector Routing

Distance table data structure

- Each node has its own table with a...
  - Row for each possible destination
  - Column for each directly-attached adjacent node (neighbor)
- Each table entry gives cost to reach destination via that adjacent node
  - Distance = Cost

\[ D^X(Y,Z) = \text{distance from } X \text{ to } Y \text{ via } Z \text{ as first hop} \]
\[ = c(X,Z) + \min_{w} \{ D^X(Y,w) \} \]
\[ w = \{ \text{neighbors of } Z \} \]

Distance table example

\[
\begin{array}{c|ccc}
\text{Destination} & A & B & D \\
\hline
D & 4 & 11 & 2 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
\text{Destination} & A & B & D \\
\hline
A & | & 1 & 4 & 5 \\
B & 7 & 8 & 5 \\
C & 6 & 9 & 4 \\
D & 4 & 11 & 2 \\
\end{array}
\]

\[
\begin{align*}
D^E(C,D) &= c(E,D) + \min_{w} \{ D^E(C,w) \} \\
&= 2 + 2 = 4 \\
D^E(A,D) &= c(E,D) + \min_{w} \{ D^E(A,w) \} \\
&= 2 + 3 = 5 \\
D^E(A,B) &= c(E,B) + \min_{w} \{ D^E(A,w) \} \\
&= 8 + 6 = 14 \\
\end{align*}
\]

A loop?!  
Loop!
**Distance Vector Routing**

**Distance table example**

- The distance table gives the routing table
  - Just take the minimum cost per destination

<table>
<thead>
<tr>
<th>Destination</th>
<th>(D^k())</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>
<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>

**Routing Table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>(D^k())</th>
<th>(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>(A, 1)</td>
<td></td>
</tr>
<tr>
<td>(B)</td>
<td>(D, 5)</td>
<td></td>
</tr>
<tr>
<td>(C)</td>
<td>(D, 4)</td>
<td></td>
</tr>
<tr>
<td>(D)</td>
<td>(D, 2)</td>
<td></td>
</tr>
</tbody>
</table>

**Distance Vector Routing Algorithm**

- Iterative, asynchronous:
  - each local iteration caused by:
    » Local link cost change, or
    » Message from adjacent node that its least cost path to some destination has changed
- Distributed:
  » Each node notifies adjacent nodes only when its least cost path to some destination changes
  » Adjacent nodes then notify their adjacent nodes if this update changes a least cost path

Each node:

1. **wait for** change in local link cost or message from adjacent node
2. **recompute** distance table
3. **if** least cost path to any destination has changed, **notify** adjacent nodes
Distance Vector Routing

Algorithm

- Initialization phase: At all nodes X:

\[
\text{for all adjacent nodes } V \{ \\
D^X(\ast,V) = \infty \quad /\!\!\!/ \text{ the cost to reach all destinations through any neighbor is infinite } \\
D^X(V,V) = c(X,V) \quad /\!\!\!/ \text{ record the cost to reach each adjacent node (cost from X to each V) } \\
\}
\]

\[
\text{for all destinations } Y \text{ & adjacent nodes } V \{ \\
\text{send } \min_W D^X(Y,W) \text{ to } V \quad /\!\!\!/ \text{ send current minimum costs for all destinations to all neighbors } \\
\}
\]

Distance Vector Routing

Algorithm main loop (at node X)

loop
wait until (receive link cost change to adjacent node v or receive new_val \(= \min_W D^X(Y,W)\) from v)
if (c(X,v) changes by d) \{ /* d could be + or - */
  /* change cost to all destinations via v by d */
  for all destinations y \{ /* includes v */
    DX(y,v) = DX(y,v) + d
  }
else { if (received new_val for y from v)
  /* shortest path from v to some y has changed */
  /* change the distance to y through v */
  DX(y,v) = c(X,v) + new_val
  }
for all destinations y \{
  \text{find } \min_w DX(y,w) \quad /\!\!\!/ \text{ w is all X's neighbors } \\
  \text{if (new } \min_{\text{cost}}(y)) \{ /* new minimum cost to y found */
    \text{for all adjacent nodes v}
    \text{send } \text{new_val }= \text{min}\_\text{cost}(y) \text{ to } v
  \}
  \}
forever
Distance Vector Algorithm
Example

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

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

Example

Distance Vector Algorithm

Link cost changes

◆ When a node detects a local link cost change:
  » The node updates its distance table
  » If the least cost path changes, the node notifies its neighbors

“Good news travels fast”
Distance Vector Algorithm

Link cost changes

- Good news travels fast, but...
- "Bad news" travels slow!
  > The "count to infinity" problem

The Count to Infinity Problem

The "poisoned reverse" technique

- If $Z$ routes through $Y$ to get to $X$:
  > Then $Z$ tells $Y$ that $Z$'s distance to $X$ is infinite

Initialization...
The Count to Infinity Problem

The “poisoned reverse” technique

- If Z routes through Y to get to X:
  - Then Z tells Y that Z’s distance to X is infinite
- (Will this completely solve the problem?)

Least Cost Path Computations

Comparison of the link-state & distance vector algorithms

- Message complexity:
  - LS: With N nodes, E links, O(NxE) messages sent for flooding
  - DV: Exchange between neighbors only (may trigger further exchanges)

- Speed of Convergence:
  - LS: O(N^2) algorithm and O(NxE) messages
    - May have oscillations
  - DV: Convergence time varies
    - Routing loops possible
    - Count-to-infinity problem

- Robustness: what happens if there are failures?
  - LS: Node can advertise incorrect link cost
    - Each node computes only its own table
  - DV: Node can advertise incorrect path cost
    - Each node’s table used by others
      - Errors propagate through network