The Network Layer: Routing 2: Distance Vector

Overview

- Routing Algorithms
  - Link-state - From last week
  - Distance-vector - TODAY
Overview of Routing so far

- Routing algorithms
  - Find the 'best' path through a network
  - Create forwarding tables
- Routing occurs between routers (not hosts)
- Differences between centralized (global) and decentralized algorithms
  - What are examples of each
  - Amount of information known initially
  - How information is shared/spread
  - Synchronous or asynchronous?

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!
Distance Vector Algorithm

Bellman-Ford Equation

Define
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]
Then
\[ d_x(y) = \min_v \{ c(x,v) + d_v(y) \} \]
where \( \min \) is taken over all neighbors \( v \) of \( x \)

Bellman-Ford Equation

\[ \begin{array}{c}
\text{Clearly, } d_v(z) = \\
\text{, } d_x(z) = \\
\text{, } d_w(z) = \\
\end{array} \]

B-F equation says:
\[ d_u(z) = \min \{ \begin{array}{l}
c(u,v) + d_v(z), \\
c(u,x) + d_x(z), \\
c(u,w) + d_w(z) \\
\end{array} \} \]
\[ = \min \{ \begin{array}{l}
\end{array} \} = \]

The node that achieves the minimum, is the next hop in the shortest path \( \rightarrow \) forwarding table
Distance Vector Routing Algorithm

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

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

Distance Table: example with complete information

\[ D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\} \]
\[ = \]
\[ D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\} \]
\[ = \]
\[ D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\} \]
\[ = \]
Distance table to forwarding table

Distance vector algorithm

Asynchronous Iterations:

- 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
  - A vector of least costs from itself to all routers

- 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
  
  \[ D_x(y) \leftarrow \min_{v \in N} \{ c(x,v) + D_v(y) \} \quad \text{for each node } y \in N \]

- Under normal conditions, the estimate $D_x(y)$ converges to the actual least cost $d_x(y)$
Distance Vector Algorithm: example for obtaining complete information

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

\[ = 7 + 1 = 8 \]

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

\[ = 2 + 1 = 3 \]

Distance Vector Algorithm: obtaining info
Distance Vector Routing Activity

A  B

C  D

Distance Vector Routing Activity

A  B

C  D

- Review actual graph – does it match activity results?
- What happens if/when \( c(A, D) = 4 \) & \( c(C, D) = 1 \)?
• Review actual graph – does it match your results?
• What happens if/when \( c(\text{A,B}) = 2 \) and/or if \( c(\text{C,D}) = 5 \)?
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