TDTS43 Lab Lecture 2
Distance Vector Routing
Chapter 4: Network Layer

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
    Datagram format
    IPv4 addressing
    ICMP
    IPv6

4.5 Routing algorithms
    Link state
    Distance Vector
    Hierarchical routing

4.6 Routing in the Internet
    RIP
    OSPF
    BGP

4.7 Broadcast and multicast routing
Interplay between routing, forwarding

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

value in arriving packet’s header
Graph abstraction

Graph: $G = (N,E)$

$N = \text{set of routers} = \{ u, v, w, x, y, z \}$

$E = \text{set of links} = \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where $N$ is set of peers and $E$ is set of TCP connections
Graph abstraction: costs

- $c(x, x')$ = cost of link $(x, x')$
  - e.g., $c(w, z) = 5$
- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{p-1}, x_p)$

Question: What's the least-cost path between $u$ and $z$?

Routing algorithm: algorithm that finds least-cost path
Routing Algorithm classification

Global or decentralized information?

Global:
all routers have complete topology, link cost info
“link state” algorithms

Decentralized:
router knows physically-connected neighbors, link costs to neighbors
iterative process of computation, exchange of info with neighbors
“distance vector” algorithms

Static or dynamic?

Static:
routes change slowly over time

Dynamic:
routes change more quickly
periodic update in response to link cost changes
Chapter 4: Network Layer

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
   Datagram format
   IPv4 addressing
   ICMP
   IPv6

4.5 Routing algorithms
   Link state
   Distance Vector
   Hierarchical routing

4.6 Routing in the Internet
   RIP
   OSPF
   BGP

4.7 Broadcast and multicast routing
Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

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 example

Clearly, \( d_v(z) = 5 \), \( d_x(z) = 3 \), \( d_w(z) = 3 \)

B-F equation says:
\[
d_u(z) = \min \{ c(u,v) + d_v(z), \quad c(u,x) + d_x(z), \quad c(u,w) + d_w(z) \}
\]
\[
= \min \{2 + 5, \quad 1 + 3, \quad 5 + 3\} = 4
\]

Node that achieves minimum is next hop in shortest path forwarding table
Distance Vector Algorithm

\[ D_x(y) = \text{estimate of least cost from } x \text{ to } y \]

Distance vector: \[ D_x = [D_x(y) : y \in N] \]

Node \( x \) knows cost to each neighbor \( v \):
\[ c(x,v) \]

Node \( x \) maintains \[ D_x = [D_x(y) : y \in N] \]

Node \( x \) also maintains its neighbors’ distance vectors

For each neighbor \( v \), \( x \) maintains
\[ D_v = [D_v(y) : y \in N] \]
Distance vector algorithm (4)

Basic idea:
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:

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

Under minor, natural conditions, the estimate \( D_x(y) \) converge to the actual least cost \( d_x(y) \).
Distance Vector Algorithm (5)

Iterative, asynchronous:
  each local iteration caused by:
    local link cost change
    DV update message from neighbor

Distributed:
  each node notifies neighbors only when its DV changes
    neighbors then notify their neighbors if necessary

Each node:
  wait for (change in local link cost of msg from neighbor)
  recompute estimates
  if DV to any dest has changed, notify neighbors
\[ D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \]
\[ = \min\{2+0, 7+1\} = 2 \]

\[ D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} \]
\[ = \min\{2+1, 7+0\} = 3 \]
Distance Vector: link cost changes

Link cost changes:
- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors

At time $t_0$, $y$ detects the link-cost change, updates its DV, and informs its neighbors.

At time $t_1$, $z$ receives the update from $y$ and updates its table. It computes a new least cost to $x$ and sends its neighbors its DV.

At time $t_2$, $y$ receives $z$'s update and updates its distance table. $y$'s least costs do not change and hence $y$ does not send any message to $z$. 
Distance Vector: link cost changes

Link cost changes:
  good news travels fast
  bad news travels slow - “count to infinity” problem!
  44 iterations before algorithm stabilizes: see text

Poissoned 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)
  will this completely solve count to infinity problem?