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

A Distance Vector Routing Algorithm

Decentralized algorithm:
- Router knows its neighbors and link costs to neighbors
- Iterative computation, exchange of info with neighbors

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, \( d_v(z) = 5 \), \( d_x(z) = 3 \), \( d_u(z) = 3 \)

Bellman-Ford equation says:

\[
d_u(z) = \min \{ c(u,v) + d_v(z), \]
\[
c(u,x) + d_x(z), \]
\[
c(u,w) + d_w(z) \}
\]

\[
= \min \{ 2 + 5, \}
\]
\[
1 + 3, \]
\[
5 + 3 \} = 4
\]

Node that yields minimum is next hop in shortest path ➔ forwarding table

Distance Vector Algorithm

Iterative, asynchronous:

- Each local iteration caused by:
  - Local link cost change
  - DV update message from neighbor

Distributed:

- Each node notifies neighbors only when its Distance Vector changes
  - Neighbors then notify their neighbors if necessary

Each node:

- wait for (change in local link cost of msg from neighbor)
- recompute estimates
- if Distance Vector to any dest has changed, notify neighbors
Distance Vector (DV): Link Cost Changes

Link cost changes:
- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbors

“good news travels fast”
- At time $t_0$, $y$ detects link-cost change, updates its DV, and informs its neighbors.
- At time $t_1$, $z$ receives update from $y$ and updates its table, 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. As $y$'s least costs do not change $y$ does not send updates to $z$. 

\[ D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \]
\[ = \min\{2+0 , 7+1\} = 2 \]
\[ D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} \]
\[ = \min\{2+1 , 7+0\} = 3 \]
### Distance Vector: Link Cost Changes

**Link cost changes:**
- Good news travels fast
- Bad news travels slow
\[ D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} \]
\[ = \min\{60 + 0, 1 + 5\} = 6 \]

\[ D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} \]
\[ = \min\{60 + 0, 1 + 7\} = 8 \]

**Distance Vector: Link Cost Changes**

**Link cost changes:**
- Good news travels fast
- Bad news travels slow – “count to infinity” problem!
- E.g., 44 iterations before algorithm stabilizes

**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?
A Link-State Routing Algorithm

- Net topology, link costs known to all nodes
  - Accomplished via “link state broadcast”
  - All nodes have same info
- Computes least cost paths from one node ("source") to all other nodes
  - Gives routing table for that node
- Example:
  - Dijkstra’s algorithm
    - Iterative: after $k$ iterations, know least cost path to $k$ dst.’s

Dijkstra’s Algorithm: Example

<table>
<thead>
<tr>
<th>Step</th>
<th>start</th>
<th>$N$</th>
<th>$D(B),p(B)$</th>
<th>$D(C),p(C)$</th>
<th>$D(D),p(D)$</th>
<th>$D(E),p(E)$</th>
<th>$D(F),p(F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td></td>
<td>2,A</td>
<td>5,A</td>
<td>1,A</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td></td>
<td>2,A</td>
<td>4,D</td>
<td>2,D</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td></td>
<td>2,A</td>
<td>3,E</td>
<td>4,E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td></td>
<td></td>
<td>3,E</td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra’s Algorithm: Example (2)

Resulting shortest-path tree from A:

```
A  B  C  D  E  F
(B, A)
(D, A)
(D, A)
(D, A)
(D, A)
```

Resulting forwarding table at A:

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>(A, B)</td>
</tr>
<tr>
<td>D</td>
<td>(A, D)</td>
</tr>
<tr>
<td>E</td>
<td>(A, D)</td>
</tr>
<tr>
<td>C</td>
<td>(A, D)</td>
</tr>
<tr>
<td>F</td>
<td>(A, D)</td>
</tr>
</tbody>
</table>

Dijkstra’s Algorithm: Discussion

Oscillations possible:

- E.g., link cost = amount of carried traffic
Comparison of LS and DV Algorithms

Speed of Convergence
- **LS**: $O(n \log n)$ algorithm requires $O(n^2)$ 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