Link State Routing

Jasleen Kaur

Fall 2014

Intra-domain Routing: Formulation

- Intra-domain routing \( \approx \) 100 routers

- Given:
  - Graph: where nodes are routers and edges are links
  - Cost: associated with each link

- Find:
  - Lowest-cost path between any two nodes

- Requirements:
  - Self-healing, traffic-sensitive, scalable

Need dynamic, distributed algorithms!
Two classes: based on “distance-vector” and “link-state”
Link-state Routing: Basic Idea

- Speed of convergence is key advantage of link-state routing

- Approach: if each node has complete info about all links, it can:
  - Build complete map of network
  - And compute shortest path to any node

- Two key mechanisms:
  - Reliable dissemination (of complete link-state of the network)
  - Calculation of routes (from the sum of accumulated link-state)

Link State Routing: Reliable Flooding

- On link-cost changes, and periodically, each node creates a link-state packet (LSP) that contains:
  - For enabling route-computation
    - ID of node that created it
    - List of directly-connected neighbors + cost of link to each
  - For ensuring reliability of flooding
    - Sequence number
    - TTL for this packet

- Transmission of LSPs between adjacent routers is made reliable
  - Using ACKs and retransmissions

- When K receives an LSP originated at Y, it stores it if:
  - Has no previous state (or has only smaller sequence number) from Y
    - If it stores, it also forwards to all neighbors (except one who forwarded LSP)
Link State Flooding Algorithm: Example

1. Link State Flooding Algorithm: Example

2. Link State Flooding Algorithm: Example

3. Link State Flooding Algorithm: Example

4. Link State Flooding Algorithm: Example

5. Link State Flooding Algorithm: Example

6. Link State Flooding Algorithm: Example

Copyright © by Jasleen Kaur
Link State Routing: Dijkstra’s Algorithm

1 Initialization:
2 \( N = \{A\} \)
3 for all nodes \( v \)
4 if \( v \) adjacent to \( A \)
5 then \( D(v) = c(A,v) \)
6 else \( D(v) = \infty \)
7
8 Loop
9 find node \( w \) not in \( N \) such that \( D(w) \) is a minimum
10 add node \( w \) to \( N \)
11 update \( D(v) \) for all nodes \( v \) adjacent to \( w \) and not in \( N \):
12 \( D(v) = \min(D(v), D(w) + c(w,v)) \)
13 /* new cost to node \( v \) is either old cost to \( v \) or known
14 shortest path cost to \( w \) plus cost from \( w \) to \( v \) */
15 until all nodes in \( N \)

Dijkstra’s Algorithm: Example

<table>
<thead>
<tr>
<th>Step</th>
<th>start 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>2,A</td>
<td>5,A</td>
<td>1,A</td>
<td>\infty</td>
<td>\infty</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>1</th>
<th>B</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

\( N \) is the set of nodes to which we have computed the minimum cost path
\( D(x) \) is the current minimum cost path to \( x \)
\( c(n,m) \) is the cost of the link from \( n \) to \( m \)
\( p(x) \) is the predecessor of \( x \) on the current minimum cost path to \( x \)

Copyright © by Jasleen Kaur
**Link State Routing: Oscillating Routes**

- "Route oscillations" are possible in link state algorithms
- Let the link cost equal the amount of carried traffic
  - Assume the link cost is updated as traffic changes

![Diagram showing route oscillations](image)

**Least Cost Path Computations**

**Link-state vs. 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)
    - Due to reliable flooding, LS considered to generate less traffic

- **Speed of Convergence:**
  - LS: $O(N^2)$ algorithm and $O(NxE)$ messages
    - May have oscillations depending on choice of metric
  - 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