Lecture 15:
Distance-vector Routing
Lecture 15 Overview

• Link-state convergence

• Distance vector
  • Assume each router knows its own address and cost to reach each of its directly connected neighbors

• Bellman-Ford algorithm
  • Distributed route computation using only neighbor’s info
Need to remove failed/old links from topology
- LSPs carry sequence numbers to distinguish new from old
- Routers only accept (and forward) the “newest” LSP
- Send a new LSP with cost infinity to signal a link down

But also need to remove entire routers
- TTL in every LSP, decremented periodically by each router
- When TTL = 0, purge the LSP and flood the network with an LSP with TTL 0 to tell everyone else to do the same
When to Flood?

- Triggered by a topology change
  - Link or node failure/recovery or
  - Configuration change like updated link metric
  - Converges quickly, but can cause flood of updates

- Periodically
  - Typically (say) every 30 minutes
  - Corrects for possible corruption of the data
  - Limits the rate of updates, but also failure recovery
Convergence

● Getting consistent routing information to all nodes
  ♦ E.g., all nodes having the same link-state database
  ♦ Until routing protocol converges, strange things happen…

● Consistent forwarding after convergence
  ♦ All nodes have the same link-state database
  ♦ All nodes forward packets on shortest paths
  ♦ The next router on the path forwards to the next hop
Detection delay
- A node does not detect a failed link immediately
- … and forwards data packets into a **black hole**
- Depends on timeout for detecting lost hellos
Transient Disruptions

- Inconsistent link-state database
  - Some routers know about failure before others
  - The shortest paths are no longer consistent
  - Can cause transient forwarding loops
Convergence Delay

- Sources of convergence delay
  - Detection latency
  - Flooding of link-state information
  - Shortest-path computation
  - Creating the forwarding table

- Performance during convergence period
  - Lost packets due to blackholes and TTL expiry
  - Looping packets consuming resources
  - Out-of-order packets reaching the destination

- Very bad for VoIP, online gaming, and video
Reducing Delay

- Faster detection
  - Smaller hello timers
  - Link-layer technologies that can detect failures

- Faster flooding
  - Flooding immediately
  - Sending link-state packets with high-priority

- Faster computation
  - Faster processors on the routers
  - Incremental Dijkstra’s algorithm

- Faster forwarding-table update
  - Data structures supporting incremental updates
Real Link-state Protocols

- OSPF (Open Shortest Path First) and IS-IS
  - Most widely used intra-domain routing protocols
  - Run by almost all ISPs and many large organizations

- Basic link state algorithm plus many features:
  - Authentication of routing messages
  - Extra hierarchy: Partition into routing areas
    » “Border” router pretends to be directly connected to all routers in an area (answers for them)
  - Load balancing: Multiple equal cost routes
Link State evaluation

● Strengths
  ◆ Loop free as long as LS database’s are consistent
    » Can have transient routing loops – shouldn’t last long
  ◆ Messages are small
  ◆ Converges quickly
  ◆ Guaranteed to converge

● Weaknesses
  ◆ Must flood data across entire network (scalability?)
  ◆ Must maintain state for entire topology (database)
Distance vector algorithm

- **Base assumption**
  - Each router knows its *own address* and the cost to reach each of its *directly connected neighbors*.

- **Bellman-Ford algorithm**
  - Distributed route computation using *only neighbor’s info*.

- **Mitigating loops**
  - Split horizon and poison reverse.
Bellman-Ford Algorithm

- Define distances at each node \( X \)
  - \( d_x(y) = \text{cost of least-cost path from } X \text{ to } Y \)

- Update distances based on neighbors
  - \( d_x(y) = \min \{c(x,v) + d_v(y)\} \) over all neighbors \( V \)

\[
d_u(z) = \min\{c(u,v) + d_v(z), \ c(u,w) + d_w(z)\}
\]
Distance Vector Algorithm

Iterative, asynchronous: each local iteration caused by:
- Local link cost change
- Distance vector update message from neighbor

Distributed:
- Each node notifies neighbors when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:
- Wait for (change in local link cost or message from neighbor)
- Recompute estimates
- If distance to any destination has changed, notify neighbors
Step-by-Step

● \( c(x,v) \) = cost for direct link from \( x \) to \( v \)
  ◆ Node \( x \) maintains costs of direct links \( c(x,v) \)

● \( D_x(y) \) = estimate of least cost from \( x \) to \( y \)
  ◆ Node \( x \) maintains distance vector \( D_x = [D_x(y) : y \in N] \)

● Node \( x \) maintains its neighbors’ distance vectors
  ◆ For each neighbor \( v \), \( x \) maintains \( D_v = [D_v(y) : y \in N] \)

● Each node \( v \) periodically sends \( D_v \) to its neighbors
  ◆ And neighbors update their own distance vectors
  ◆ \( D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \) for each node \( y \in N \)
Example: Initial State

Info at node | Distance to Node
--- | --- | --- | --- | --- | --- | ---
A | B | C | D | E
--- | --- | --- | --- | ---
A | 0 | 7 | 1 | 1
B | 7 | 0 | 1 | 8
C | 1 | 0 | 2 | 2
D | 2 | 0 | 2 | 2
E | 1 | 8 | 2 | 0

Example: Initial State

CSE 123 – Lecture 15: Distance-vector Routing
**CSE 123 – Lecture 15: Distance-vector Routing**

**D sends vector to E**

- **Info at node**
  - **Distance to Node**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>E</strong></td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

- **I’m 2 from C, 0 from D and 2 from E**
- **D is 2 away, 2+2<**, so best path to C is 4
$B$ sends vector to $A$

I’m 7 from $A$, 0 from $B$, 1 from $C$ & 8 from $E$

B is 7 away, 1+7< so best path to C is 8

<table>
<thead>
<tr>
<th>Info at node</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>0</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$B$</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>$C$</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$D$</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$E$</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
E sends vector to A

E is 1 away, 4+1<8
so C is 5 away, 1+2<
so D is 3 away

I’m 1 from A, 8 from B, 4
from C, 2 from D & 0 from E

CSE 123 – Lecture 15: Distance-vector Routing
…until Convergence

CSE 123 – Lecture 15: Distance-vector Routing
Node $B$’s distance vectors

<table>
<thead>
<tr>
<th>Dest</th>
<th>A</th>
<th>E</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>
Handling Link Failure

- A marks distance to E as 8, and tells B
- E marks distance to A as 5, and tells B and D
- B and D recompute routes and tell C, E and E
- etc… until converge

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
</tr>
<tr>
<td>E</td>
<td>12</td>
</tr>
</tbody>
</table>
Problem: Counting to Infinity

Distance to C

CSE 123 – Lecture 15: Distance-vector Routing
Why so High?

- Updates don’t contain enough information
- Can’t totally order “bad news” (a link has gone down) above “good news” (a link is available)
- $B$ accepts $A$’s path to $C$ that is *implicitly* through $B$!
- Aside: this also causes delays in convergence even when it doesn’t count to infinity
Hold downs
- As metric increases, delay propagating information
- Limitation: Delays convergence

Loop avoidance
- Full path information in route advertisement
- Explicit queries for loops

Split horizon
- Never advertise a destination through its next hop
  - A doesn’t advertise C to B
- Poison reverse: Send negative information when advertising a destination through its next hop
  - A advertises C to B with a metric of
  - Limitation: Only works for “loop”s of size 2
For next time…

- Read Ch. 3.4 in P&D
- Get moving on Project 2