Lecture 16: Distance Vector Routing

CSE 123: Computer Networks
Aaron Schulman
Lecture 16 Overview

• Wrap up discussion of link state routing
  • Convergence delay and how to mitigate it

• Distance vector routing
Link State Convergence Delay

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

- Poor performance during convergence period
  - Lost packets due to black holes 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 LSP 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**
  - **No flooding** like in link-state routing

- Mitigating loops
  - Split horizon and poison reverse
Define distances at each node $X$
- $d_x(y) =$ cost of least-cost path from $X$ to $Y$

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

Bellman-Ford Algorithm

CSE 123 – Lecture 16: Distance vector routing
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:
- \textit{wait} for (change in local link cost or message from neighbor)
- \textit{recompute} estimates
- if distance to any destination has changed, \textit{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

CSE 123 – Lecture 16: Distance vector routing
$D$ sends vector to $E$

I’m 2 from C, 0 from D and 2 from E

D is 2 away, 2+2< $\infty$, so best path to C is 4

<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>$\infty$</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>
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>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>1</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

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 7 5 3 1</td>
</tr>
<tr>
<td>B</td>
<td>7 0 1 ∞ 8</td>
</tr>
<tr>
<td>C</td>
<td>∞ 1 0 2 ∞</td>
</tr>
<tr>
<td>D</td>
<td>∞ ∞ 2 0 2</td>
</tr>
<tr>
<td>E</td>
<td>1 8 4 2 0</td>
</tr>
</tbody>
</table>
…until Convergence

<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>6</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>
Node B’s distance vectors

Node B:
- Distance to A: 7
- Distance to C: 1
- Distance to D: 2
- Distance to E: 8

Next hop:

<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>

CSE 123 – Lecture 16: Distance vector routing
Handling Link Failure

- A marks distance to E as $\infty$, and tells B
- E marks distance to A as $\infty$, 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>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>10</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>12</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Problem: Counting to Infinity

Distance to C

CSE 123 – Lecture 15: Interdomain 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
Mitigation Strategies

- **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 $\infty$
    - Limitation: Only works for “loop”s of size 2
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)
Split Horizon Limitations

- **A** tells **B** & **C** that **D** is unreachable
- **B** computes new route through **C**
  - Tells **C** that **D** is unreachable (poison reverse)
  - Tells **A** it has path of cost 3 (split horizon doesn’t apply)
- **A** computes new route through **B**
  - **A** tells **C** that **D** is now reachable
- Etc…
In practice

- **RIP: Routing Information Protocol**
  - DV protocol with hop count as metric
    - Infinity value is 16 hops; limits network size
    - Includes split horizon with poison reverse
  - Routers send vectors every 30 seconds
    - With triggered updates for link failures
    - Time-out in 180 seconds to detect failures
  - Rarely used today

- **EIGRP: proprietary Cisco protocol**
  - Ensures loop-freedom (DUAL algorithm)
  - Only communicates changes (no regular broadcast)
  - Combine multiple metrics into a single metric (BW, delay, reliability, load)
Distance Vector shortest-path routing
- Each node sends list of its shortest distance to each destination to its neighbors
- Neighbors update their lists; iterate

Weak at adapting to changes out of the box
- Problems include loops and count to infinity
For next time...

- HW3 out tonight
- Project 3 out Friday
- Read Chapter 4.1 in P&D (internetwork routing)