Lecture 15:
Distance-vector Routing

Attention: Project 2 is assigned
Recall: Basic Approaches

- **Static**
  - Type in the right answers and hope they are always true
  - ...So far

- **Link state**
  - Tell everyone what you know about your neighbors
  - Last lecture

- **Distance vector**
  - Tell your neighbors when you know about everyone
  - Today
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 $X$
  - $d_x(y) =$ 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 $V$

$$d_u(z) = \min\{c(u,v) + d_v(z), c(u,w) + d_w(z)\}$$
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:

\[
\text{wait for (change in local link cost or message from neighbor)}
\]

\[
\text{recompute estimates}
\]

if distance to any destination has changed, \text{notify} neighbors
Step-by-Step

- $c(x,v) = \text{cost for direct link from } x \text{ to } v$
  - Node $x$ maintains costs of direct links $c(x,v)$

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

![Diagram with nodes A, B, C, D, E and distances between them]

<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>$\infty$</td>
<td>$\infty$</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>$\infty$</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>$\infty$</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>8</td>
<td>$\infty$</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
**D sends vector to E**

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

<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>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</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></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>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>
I’m 1 from A, 8 from B, 4 from C, 2 from D & 0 from E

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

E sends vector to A
...until Convergence

![Diagram of a network with nodes A, B, C, D, and E connected by arrows indicating distances.]

<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>6</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
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 $\infty$, 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
Problem: Counting to Infinity

Distance to C

Update 3

Update 4

Etc…
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
    » Limitation: Only works for “loop”s of size 2

  - Poison reverse: Send negative information when advertising a destination through its next hop
    » A advertises C to B with a metric of $\infty$
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
Link-state vs. Distance-vector

Message complexity
- **LS**: with $n$ nodes, $E$ links, $O(nE)$ messages sent
- **DV**: exchange between neighbors only

Speed of Convergence
- **LS**: relatively fast
- **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**:
  - Node can advertise incorrect *path* cost
  - Each node’s table used by others (error propagates)
Routing so far…

- Shortest-path routing
  - Metric-based, using link weights
  - Routers share a common view of path “goodness”

- As such, commonly used inside an organization
  - EIGRP and OSPF are mostly used as intradomain protocols

- But the Internet is a “network of networks”
  - How to stitch the many networks together?
  - When networks may not have common goals
  - … and may not want to share information
For next time…

- Read Ch. 4.1 in P&D
- Get moving on Project 2