Lecture 8: Routing I
Distance-vector Algorithms
This class

- New topic: routing

How do I get there from here?
Overview

- Routing overview
- Intra vs. Inter-domain routing
- Distance-vector routing protocols
Router Tasks

- **Forwarding**
  - Move packet from input link to the appropriate output link
  - Purely local computation
  - Must go be very fast (executed for every packet)

- **Routing**
  - Make sure that the next hop actually leads to the destination
  - Global decisions; distributed computation and communication
  - Can go slower (only important when topology changes)
Forwarding Options

- **Source routing**
  - Complete path listed in packet

- **Virtual circuits**
  - Set up path out-of-band and store path identifier in routers
  - Local path identifier in packet

- **Destination-based forwarding**
  - Router looks up address in forwarding table
  - Forwarding table contains (address, next-hop) tuples
Source Routing

- **Routing**
  - Host computes path
    - Must know global topology and detect failures
  - Packet contains complete ordered path information
    - I.e. node A then D then X then J…
  - Requires variable length path header

- **Forwarding**
  - Router looks up next hop in packet header, strips it off and forwards remaining packet
    - Very quick forwarding, no lookup required

- **In practice**
  - ad hoc networks (DSR), some HPC networks (Myrinet), and for debugging on the Internet (LSR, SSR)
Virtual Circuits

- **Routing**
  - Hosts sets up path out-of-band, requires *connection setup*
  - Write (input id, output id, next hop) into each router on path
  - Flexible (one path per flow)

- **Forwarding**
  - Send packet with path id
  - Router looks up input, swaps for output, forwards on next hop
  - Repeat until reach destination
  - Table lookup for forwarding (faster than IP lookup?)

- **In practice**
  - ATM: fixed VC identifiers and separate signaling code
  - MPLS: ATM meets the IP world (why? *traffic engineering*)
Destination-based Forwarding

- **Routing**
  - All addresses are globally known
    - No connection setup
  - Host sends packet with destination address in header
    - No path state; only routers need to worry about failure
  - Distributed routing protocol used to routing tables

- **Forwarding**
  - Router looks up destination in table
    - Must keep state proportional to destinations rather than connections
  - Lookup address, send packet to next-hop link
    - All packets follow same path to destination

- **In Practice: IP routing**
Routing Tables

- The routing table at A, lists *at a minimum* – the next hops for the different destinations

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
</tr>
</tbody>
</table>
Routing on a Graph

- Essentially a graph theory problem
  - Network is a directed graph; routers are vertices
- Find “best” path between every pair of vertices
  - In the simplest case, best path is the shortest path

Diagram:

- X = router
- \( \frac{X}{\text{link}} \)
- 1 = cost
Routing Challenges

- How to choose best path?
  - Defining “best” can be slippery

- How to scale to millions of users?
  - Minimize control messages and routing table size

- How to adapt to failures or changes?
  - Node and link failures, plus message loss
Intra-domain Routing

- Routing within a network/organization
  - A single administrative domain
  - The administrator can set edge costs

- Overall goals
  - Provide intra-network connectivity
  - Adapt quickly to failures or topology changes
  - Optimize use of network resources

- Non-goals
  - Extreme scalability
  - Lying, and/or disagreements about edge costs
  - We’ll deal with these when we talk about inter-domain routing
Basic Approaches

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

- **Distance vector**
  - Tell your neighbors when you know about everyone
  - Today’s lecture!

- **Link state**
  - Tell everyone what you know about your neighbors
  - Next time…
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 position 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 only 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

```
<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>
```
$D$ 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>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>
Info at 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>A</td>
<td>0</td>
<td>7</td>
<td>8</td>
<td>∞</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>∞</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</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>

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

B sends vector to A
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

- **Info at node**
  - A
  - B
  - C
  - D
  - E

- **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>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

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 7 8 10 12</td>
</tr>
<tr>
<td>B</td>
<td>7 0 1 3 5</td>
</tr>
<tr>
<td>C</td>
<td>8 1 0 2 4</td>
</tr>
<tr>
<td>D</td>
<td>10 3 2 0 2</td>
</tr>
<tr>
<td>E</td>
<td>12 5 4 2 0</td>
</tr>
</tbody>
</table>
Counting to Infinity

Distance to C

Etc…
Why so High?

- Updates don’t contain enough information
- Can’t totally order bad news above good news
- $B$ 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
Poison Reverse Example

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)
Summary

• Routing is a distributed algorithm
  ◆ React to changes in the topology
  ◆ Compute the paths through the network

• 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
Next time

- Link state routing
- Turn in homework…