Internet Routing: Overview and Distance Vector Algorithms
This class

- New topic: routing

How do I get there from here?
Overview

- Intro & Design choices
- Intra-domain routing
  - Distance vector
  - Link state
- Inter-domain routing
  - Policy
Intra-domain routing

- Routing *within* a network/organization
- A *single* administrative domain

Overall goals
- Provide intra-network connectivity
- Adapt quickly to failures or topology changes
- **Optimize** use of network resources
Intra-domain routing: Problem statement

- Routing is essentially a problem in graph theory

- Network is a directed graph $G=(V,E)$
- Routers are vertices, links are edges labeled with some metric
  - For simplicity ignore hosts, they are part of each $V$
- For each $V$, find the shortest path to every other $V$
Network as a Graph

- Find “best” path between every pair of vertices
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
Recap: What does a router do?

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

- **Routing**
  - Doing work so you’re sure that the “next hop” actually leads to the destination
  - Global decisions; distributed computation and communication
  - Can go slower (only important when topology changes)
Kinds of forwarding

- **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
  - Very flexible

- 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 (why 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
  - \((\text{Address}, \text{next-hop})\) tuple
  - Lookup address, send packet to next-hop link
    » All packets follow same path to destination

- **In Practice: IP routing**
What’s in a forwarding table?

- 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>
Three approaches to IP routing

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

- **Distance vector**
  - Tell your neighbors when you know about everyone

- **Link state**
  - Tell everyone what you know about your neighbors
Distance Vector routing

- **Assume**
  - Each router knows own address & cost to reach neighbors

- **Goal**
  - Calculate routing table containing next-hop information for every destination at each router

- **Distributed Bellman-Ford algorithm**
  - Each router maintains a vector of costs to all destinations
    - Initialize neighbors with known cost, others with infinity
  - Periodically send copy of distance vector to neighbors
  - On reception of a vector
    - If neighbor’s path to a destination is shorter, switch to it
Initial conditions

```
<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>
```
E receives D’s’s vector

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>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>
A receives B's vector

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>
A receives E’s vector

E is 1 away, 4+1<8
so C is 5 away, 1+2<
\(\infty\) 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>B</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>
Final state

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

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D, E</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0 6 5 3 1</td>
</tr>
<tr>
<td>B</td>
<td>6 0 1 3 5</td>
</tr>
<tr>
<td>C</td>
<td>5 1 0 2 4</td>
</tr>
<tr>
<td>D</td>
<td>3 3 2 0 2</td>
</tr>
<tr>
<td>E</td>
<td>1 5 4 2 0</td>
</tr>
</tbody>
</table>
View from a node (B)

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7 9 6</td>
</tr>
<tr>
<td>C</td>
<td>12 12 1</td>
</tr>
<tr>
<td>D</td>
<td>10 10 3</td>
</tr>
<tr>
<td>E</td>
<td>8 8 5</td>
</tr>
</tbody>
</table>
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>3 3 2 0 2</td>
</tr>
<tr>
<td>E</td>
<td>12 5 4 10 0</td>
</tr>
</tbody>
</table>
Problems: *Count to Infinity*

Distance to C

Etc…
Why?

- Updates don’t contain enough information.
- Can’t totally order bad news above good news.
- B’s 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.
Solutions

- **Hold downs**
  - As metric increases, delay propagating information
  - Limitation: Delays convergence

- **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

- **Loop avoidance**
  - Full path information in route advertisement
  - Explicit queries for loops (e.g. DUAL)
How split horizon/pv fails

- A tells B & C that D is unreachable
- B tells C that D is unreachable
- B tells A that D is reachable with cost=3 (since route is through C, split horizon doesn’t apply)
- A tells C that D is reachable through A (cost=4)
- Etc…
Other issues

- When to send route updates?
  - Periodically
    - Limits granularity of failure recovery
    - Global synchronization can cause packet loss
  - Jittered
    - Random offset from periodic deals with synchronization problem
  - Triggered
    - Send updates immediately when metric changes
    - Converges more quickly, but causes flood of packets
Routing Information Protocol (RIP)

- 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

- RIPv1 specified in RFC1058
  - www.ietf.org/rfc/rfc1058.txt
- RIPv2 (adds authentication etc.) in RFC1388
  - www.ietf.org/rfc/rfc1388.txt
Key Concepts

- Routing is a global process, forwarding is local one
- The Distance Vector algorithm and RIP
  - Simple and distributed exchange of shortest paths.
  - Weak at adapting to changes (loops, count to infinity)
For next time...

- Review the section about link state routing since that’s what we’ll be talking about