CSE 123b
Communications Software
Spring 2004

Lecture 6: Routing: Overview and Distance Vector Algorithms

Stefan Savage

Some slides courtesy David Wetherall
Administrativa

- Homework – up front
  - I’ll give out solutions next class

- Computer accounts
  - You should have them (if you don’t, or you enrolled via extension, come see me for an add slip after class)
  - You’re set up on ieng9 and can use the uAPE lab (AP&M B402), the Suns in EBU1-3327 and EBU1-3329
  - There is a web board for this class (go to http://webboard.ucsd.edu)
Last class

- Goals of congestion control
  - Use allocated bandwidth efficiently
  - Avoid sending so quickly that the network has to drop packets
  - Avoid sending so slowly that the network is underutilized

- Approach taken by TCP
  - Congestion window limits outstanding packets
  - Adjust congestion window in response to packet losses (AIMD)
  - Slow start
  - Fast retransmit/ fast recovery
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

- **Problem statement**
  - 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

- Routing is essentially a problem in graph theory
- Find “best” path between every pair of vertices

Network as a Graph

Routing is essentially a problem in graph theory
Find “best” path between every pair of vertices
Routing Questions

- 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
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 ever 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 routing/forwarding

- Source routing
  - Complete path in packet

- Virtual circuits
  - Routers communicate to set up *per-path* forwarding tables in routers
  - Forwarding table contains (inputid, next-hop, outputid) tuples

- Destination-based routing
  - Distributed algorithm sets up *per-destination* forwarding table among routers
  - 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), SANs (Myrinet), and for debugging on the Internet (LSR, SSR)
Virtual circuits

- **Routing**
  - Hosts set 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 and forwards on next hop with new path id
  - Repeat until reach destination
  - Table lookup for forwarding (very fast)

- **In practice**
  - ATM: fixed identifiers and separate signaling code
  - MPLS: ATM meets the IP world
Destination-based routing

- 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
  - (Address, 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 Routing Table?

- The routing table at A, for example, 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 destination-based 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

![Diagram showing network nodes A, B, C, D, and E with distances]

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0  7  ∞  ∞  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  ∞  2  0</td>
</tr>
</tbody>
</table>
E receives D’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>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 B’s vector

I’m 7 from A, 0 from B, 1 from X & 8 from D

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>
A receives E’s vector

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></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>5</td>
<td>3</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>
Final state

The diagram shows a network with nodes A, B, C, D, and E, connected by directed edges with weights 1, 2, 7, and 8. The table lists the distance to each node from each node:

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<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)

```
+---+---+---+---+---+
| A | B | C | D | E |
+---+---+---+---+---+
|    | 1 | 8 | 2 |    |
+---+---+---+---+---+
| 7  | 1 |    |    | 1  |
+---+---+---+---+---+
|    |    | 2 |    |    |
+---+---+---+---+---+
|    | 8 | 2 |    |    |
+---+---+---+---+---+
```

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>
**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
Solutions

- Limit infinity
- 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](http://www.ietf.org/rfc/rfc1058.txt)
- RIPv2 (adds authentication etc.) in RFC1388
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...

- No new reading… although review the section about link state routing

- Make sure your accounts work…