Announcements

• HW6 due this week
• HW7 due 3/21
HW7 Preview

• RIP (Routing Information Protocol)
• Components
  – Forwarding
  – Routing
• Driver, libraries, etc. provided so the focus is on networking code
• Port assignments
Today’s Topics

• Distance Vector Routing
• Link State Routing
• Inter-AS Routing
Network as a graph

• Nodes are routers
• Assign *cost* to each edge
  – Can be based on latency, b/w, queue length, ...
• Problem: find lowest-cost path between nodes
  – Each node individually computes routes
Basic Algorithms

• Two classes of intra-domain routing algorithms
• Distance Vector
  – Requires only local state
  – Harder to debug
  – Can suffer from loops
• Link State
  – Each node has global view of the network
  – Simpler to debug
  – Requires global state
Distance Vector

• Local routing algorithm
• Each node maintains a set of triples
  – \( <\text{Destination}, \text{Cost}, \text{NextHop}> \)
• Exchange updates with neighbors
  – Periodically (seconds to minutes)
  – Whenever table changes (triggered update)
• Each update is a list of pairs
  – \( <\text{Destination}, \text{Cost}> \)
• Update local table if receive a “better” route
  – Smaller cost
• Refresh existing routes, delete if time out
Shortest Path Example

B’s table

<table>
<thead>
<tr>
<th>ID</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

E’s table

<table>
<thead>
<tr>
<th>ID</th>
<th>Nexthop</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
</tr>
</tbody>
</table>

Graph:
- A connected to B with cost 4
- A connected to C with cost 2
- B connected to D with cost 8
- C connected to D with cost 6
- C connected to E with cost 12
- D connected to F with cost 1
- E connected to F with cost 1
Adapting to Failures

- F-G fails
- F sets distance to G to infinity, propagates
- A sets distance to G to infinity
- A receives periodic update from C with 2-hop path to G
- A sets distance to G to 3 and propagates
- F sets distance to G to 4, through A
Count-to-Infinity

- Link from A to E fails
- A advertises distance of infinity to E
- B and C advertise a distance of 2 to E
- B decides it can reach E in 3 hops through C
- A decides it can reach E in 4 hops through B
- C decides it can reach E in 5 hops through A, ...
- When does this stop?
Good news travels fast

- A decrease in link cost has to be fresh information
- Network converges at most in $O(\text{diameter})$ steps
Bad news travels slowly

- An increase in cost may cause confusion with old information
- May form loops
How to avoid loops

• IP TTL field prevents a packet from living forever
  – Does not *repair* a loop

• Simple approach: consider a small cost $n$ (e.g., 16) to be infinity
  – After $n$ rounds decide node is unavailable
  – But rounds can be long, this takes time
Better loop avoidance

• Split Horizon
  – When sending updates to node A, don’t include routes you learned from A
  – Prevents B and C from sending cost 2 to A

• Split Horizon with Poison Reverse
  – Rather than not advertising routes learned from A, explicitly include cost of $\infty$.
  – Faster to break out of loops, but increases advertisement sizes
Warning

• Split horizon/split horizon with poison reverse only help between two nodes
  – Can still get loop with three nodes involved
  – Might need to delay advertising routes after changes, but affects convergence time
Link State Routing

• Strategy
  – send to all nodes information about directly connected neighbors

• Link State Packet (LSP)
  – ID of the node that created the LSP
  – Cost of link to each directly connected neighbor
  – Sequence number (SEQNO)
  – TTL
Reliable Flooding

• Store most recent LSP from each node
  – Ignore earlier versions of the same LSP
• Forward LSP to all nodes but the one that sent it
• Generate new LSP periodically
  – Increment SEQNO
• Start at SEQNO=0 when reboot
  – If you hear your own packet with SEQNO=n, set your next SEQNO to n+1
• Decrement TTL of each stored LSP
  – Discard when TTL=0
Calculating best path

- Djikstra’s single-source shortest path algorithm
  - Each node computes shortest paths from itself

- Let:
  - $N$ denote set of nodes in the graph
  - $l(i,j)$ denote the non-negative link between $i,j$
    - $\infty$ if there is no direct link between $i$ and $j$
  - $C(n)$ denote the cost of path from $s$ to $n$
  - $s$ denotes yourself (node computing paths)

- Initialize variables
  - $M = \{s\}$ (set of nodes incorporated thus far)
  - For each $n$ in $N-\{s\}$, $C(n) = l(s,n)$
  - $R(n) = n$ if $l(s,n) < \infty$, otherwise
Dijkstra’s Algorithm

• While N≠M
  – Let w ∈ (N-M) be the node with lowest C(w)
  – M = M ∪ {w}
  – Foreach n ∈ (N-M), if C(w) + l(w,n) < C(n)
  – then C(n) = C(w) + l(w,n), R(n) = R(w)

• Example: D: (D,0,-) (C,2,C) (B,5,C) (A,10,C)
Distance Vector vs. Link State

• # of messages (per node)
  – DV: $O(d)$, where $d$ is degree of node
  – LS: $O(nd)$ for $n$ nodes in system

• Computation
  – DV: convergence time varies (e.g., count-to-infinity)
  – LS: $O(n^2)$ with $O(nd)$ messages

• Robustness: what happens with malfunctioning router?
  – DV: Nodes can advertise incorrect path cost
  – DV: Others can use the cost, propagates through network
  – LS: Nodes can advertise incorrect link cost
Examples

• RIPv2
  – Fairly simple implementation of DV
  – RFC 2453 (38 pages)

• OSPF (Open Shortest Path First)
  – More complex link-state protocol
  – Adds notion of *areas* for scalability
  – RFC 2328 (244 pages)
RIPv2

- Runs on UDP port 520
- Link cost = 1
- Periodic updates every 30s, plus triggered updates
- Relies on count-to-infinity to resolve loops
  - Maximum diameter 15 (∞ = 16)
  - Supports split horizon, poison reverse
Packet format

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>command (1)</th>
<th>version (1)</th>
<th>must be zero (2)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RIP Entry (20)</th>
</tr>
</thead>
</table>

~ RIP Entry (20) ~
# RIPv2 Entry

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|   +------------------------------------------------------------------+
|   | address family identifier (2) | Route Tag (2) |
|   +------------------------------------------------------------------+
|   | IP address (4) |
|   +------------------------------------------------------------------+
|   | Subnet Mask (4) |
|   +------------------------------------------------------------------+
|   | Next Hop (4) |
|   +------------------------------------------------------------------+
|   | Metric (4) |
|   +------------------------------------------------------------------+
Next Hop field

• Allows one router to advertise routes for multiple routers on the same subnet

• Suppose only XR1 talks RIPv2:

```
|IR1| |IR2| |IR3| |XR1| |XR2| |XR3|
|---|---|---|---|---|---|---|---|
```

<-----------------------------RIP-2-------------------------->
OSPFv2

• Link state protocol
• Runs directly over IP (protocol 89)
  – Has to provide its own reliability
• All exchanges are authenticated
• Adds notion of areas for scalability
OSPF Areas

- Backbone
- Boundary router
- Backbone router
- Area border routers
- Internal routers

Area 1
Area 2
Area 3
Inter-domain Routing
Why Inter vs. Intra

• Why not just use OSPF everywhere?
  – E.g., hierarchies of OSPF areas?
  – Hint: scaling is not the only limitation

• BGP is a policy control and information hiding protocol
  – intra == trusted, inter == untrusted
  – Different policies by different ASs
  – Different costs by different ASs