Lab 4
Dynamic Routing

CS144 Review Session
Based on slides by: Samir Selman
Administrivia

- Lab 4 – Due on Nov 11th
- Lab 3 – Due yesterday
- Can take extension upto midnight, Sunday – grade capped at 90%
- Grading - max(Lab3,Lab5)
Routing - Overview

Exterior Routing Protocols:
- BGP
- EGP

Interior Routing Protocols:
- RIP
- IGRP
Routing Information Protocol (RIP)

- Routing protocol
  - Based on Bellman-Ford, Distance Vector algorithm
  - Intra-domain protocol or Interior Gateway protocol
  - 'cost' or 'metric' is usually hop count

- Limitations:
  - Limited to networks whose longest path is 15 hops
  - Relies on “counting to infinity” to resolve situations like routing loops
  - Employs fixed metrics to compare alternative routes
Implementation – Lab 4

- Subset of RIP Version 2.
- You will implement the basic RIP procedure, split horizon with poisoned reverse, triggered updates and timer mechanism.
- You will need to focus only on the RIP part of the router; we have supplied you with the 'dr' binary.
- Binary relies on API implemented in a shared library to handle dynamic routing. This library will be implemented by you.
- You will write code in dr_api.h and dr_api.c.
RIP Basic Procedure

- Every router has a forwarding table with an entry for every possible destination network.

- Routing table entry - (destination-address, metric, next-hop, time-stamp)
  - Destination-address: Given by a network address and a network mask (X.X.X.X/Y).
  - Metric: Sum of costs of links traversed to reach destination.
  - Next-hop: IPv4 address of next router on the path to destination.
  - Time-stamp: Used for route time-outs.
RIP Packet Format

<table>
<thead>
<tr>
<th>1-octet command field</th>
<th>1-octet version number field</th>
<th>2-octet unused field</th>
<th>2-octet AFI field</th>
<th>2-octet route tag field</th>
<th>4-octet network address field</th>
<th>4-octet subnet mask field</th>
<th>4-octet next hop field</th>
<th>4-octet metric field</th>
</tr>
</thead>
</table>

- **Command field** – Indicates whether packet is a 'request' or 'response'
  - Request – Ask a router to sent part of/entire routing table.
    
    *(Do NOT worry about RIP requests in this lab)*
  - Response – Can be a periodic routing update or a reply to a request.
- **Version** – V2 of RIP
- **AFI** – Address Family Identifier (AF_INET)
- **Route tag field** – *not used in the lab, replaced by pad*
Packet format contd.

- Network address – IP address for the entry
- Subnet mask – mask for the entry
- Next hop – IP address of next hop to which packet is forwarded
- Metric – 'hops' denoting number of routers that have been traversed in the path to destination
  - Maximum 15 hops; 16 - unreachable
Routing Updates

• Send routing updates:
  • At periodic intervals
  • Triggered updates – when a router changes a metric to a route
• Send RIP reply message containing whole routing table to all neighbors every 10s.
• Triggered updates – when an interface goes down, or cost of a local interface changes.
Route timeout

- Dynamic Route :
  - Route learned from a neighboring router.
  - Have a time-stamp (ttl) field associated with every entry.
  - Set time-stamp value to current time when you receive an update.
  - Check periodically for expired entries (current-time – updatetime>20s)
  - Set cost to infinity if entry expires.

- Local Route :
  - Directly-connected networks
  - Next-hop IP = 0
  - Have an invalid TTL = -1
Routing loops

- Split horizon with poisoned reverse:
  - Prevent routing loops involving two nodes
  - Advertise information back to source with cost INFINITY.
  - Loops with more than 2 nodes? e.g. a->b, b->c, c->a?
  - Loop will be resolved only when metric reaches infinity and then the network involved is declared unreachable.
  - Speed up convergence using triggered updates.
Counting to Infinity with a Split Horizon

Distance from D

1. Stable point
2. Link CD fails
3. C believes D unreachable (SH)
4. C sends update to A and B
5. C’s update reaches B
6. A sends periodic update
7. C’s update reaches A
8. B sends periodic update

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>∞</td>
<td>3</td>
<td>∞</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Lab 4: Code

- Touch only methods starting with safe_(threading handled already)
- `safe_dr_get_next_hop(ip)` : called when router needs to know how to route a packet; returns the outgoing interface and ip address of next hop
- `safe_dr_handle_packet` : called when router receives a dynamic routing packet.
- `safe_dr_handle_periodic()` : called periodically by router.
- `safe_dr_interface_changed` : called when an i/f goes up or down, or link cost changes
- `dr_init` : Use to initialize any data structures.
Running the lab

- Start lvns server
  - ./lvns -t simple.topo

- If the server is running on the same machine as the dr instance, you can start a router which takes control of a node named 'dr1' like this:
  - ./dr -v dr1

- If you are running dr from a separate location, specify server's address and port explicitly.
  - ./dr -v dr1 -s myth5.stanford.edu -p 9999
Start Early!
Good luck!