Spring 2014
Computer Networks
CMPE323

Laboratory Experiment No. 9: Introduction to Dynamic Routing Protocols

Aims and Objectives:

- Introducing dynamic routing protocols by studying basics of Routing Information Protocol (RIP) version 2.

Materials Required:

- IP routers,
- PCs with Ethernet adapters,
- and straight-through/crossover/rollover cables.

Change Log:

- 15-5-2014: original document – mkhonji.
- 16-5-2014: more coverage of the RIPv2 timers, replaced network merge by a single readily merged network – mkhonji.

1 Introduction

Consider a scenario where routers $R_1, R_2, \ldots R_n$ are interconnected using some partial mesh topology. If full end-to-end reachability is needed, then configuring static routing tables can have an high maintenance overhead. E.g. you may have to update the routing table configuration of all of the n routers should you desire to add a new network that is reachable by all.

While static routes are highly successful in limited scenarios, they can introduce maintenance challenges in larger networks as changes occur.

Dynamic routing protocols address that problem: instead of using a human administrator to manually configure the routers $R_1 \ldots R_n$ every time a new network is added, dynamic routing protocols allow us to configure the routers once such that they can dynamically communicate with themselves and learn the routing table automatically. Additionally, if there is a failure, dynamic routing protocols can allow for automatic self-healing by automatically choosing the next best path.

This lab covers a simple but widely spread routing protocol, namely Routing Information Protocol v2. The main changes in the 2^{nd} version is the addition of CIDR support (by adding subnet masks), authentication, multicasting communication instead of broadcasting, addition of requests, route tags.

While there are more scalable routing protocols, such as IS-IS and OSPF, RIPv2 remains fairly attractive as it has a very small overhead in *small* networks.

The work flow of RIPv2 is generally as follows (a lossy summary from RFC2453 for simplicity):

- RIPv2 is enabled on router interfaces, which allows their corresponding networks to be included in the RIPv2 advertisements.

- Every interface that has RIPv2 enabled, sends RIPv2 messages that advertise its routing table every 30 seconds by default.

- Neighboring routers that hear such advertised RIP updates generally install them into their own routing table if the advertised networks have a better *metric* than the one that is in the routing table (if there is any).

- A better metric by RIPv2 is simply the total number of layer 3 hops that lead to a given network.

  E.g. if R_1 is directly connected to network N_1, it shall advertise it to is neighboring routers while associating such network advertisement with a metric of 1 (as there are 1 layer-3 hops).

  Then, the receiving neighboring router (if any) will increment the hop count by 1 and then advertise it to its own neighbors (except the network from which the advertisement came from).

  This way, as the advertised network crosses more layer 3 hops, the metric will increase. By default, the maximum reachable metric is just 15 hops.

  A hop of 16 is possible but means “infinity” and is used to advertise lack of network reachability.

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  command (1)  |  version (1)  |       must be zero (2)        |
+---------------+---------------+-------------------------------+
|                                                               |
~                         RIP Entry (20)                        ~
|                                                               |
+---------------+---------------+---------------+---------------+
```

Figure 1: RIPv2 message format. Note the possibility adding multiple routing advertisement entries — Source RFC2453.

```
 0                   1                   2                   3 3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Address Family Identifier (2) |        Route Tag (2)          |
+-------------------------------+-------------------------------+
|                         IP Address (4)                        |
+---------------------------------------------------------------+
|                         Subnet Mask (4)                       |
+---------------------------------------------------------------+
|                         Next Hop (4)                          |
+---------------------------------------------------------------+
|                         Metric (4)                            |
+---------------------------------------------------------------+
```

Figure 2: RIPv2 routing entry format — Source RFC2453.

The RIPv2 protocol is structured as follows in Figures 1 and 2:
Fields are:

- Command — type of action (request(1) or response(2)).
- Version — the version of RIP (1 for v1, and 2 for v2).
- Address Family ID — the ID of the address family (2 for IPv4).
- Route tag — this is helpful to group RIP advertisements for different treatments later one. However this is beyond the scope of this lab.
- IP address — the IPv4 address of the advertised network.
- Subnet mask — the subnet mask of the advertised network.
- Next hop — the IP address of the gateway that packets to the network should be forwarded to. A next-hop of `0.0.0.0` is simply a self-reference of the interface of the advertising router.
- Metric — the total number of hops that are to be crossed in order to reach the advertised network.

RIPv2 has the following timers as implemented by Cisco's IOS:

- Time interval between the updates t_u — by default, this timer is set to 30 seconds which causes the router to advertise routes from its routing table to its neighbors once every 30 seconds.
- Invalidation timer t_i — if a router does not receive a RIPv2 update about a given network for t_i seconds, it will consider the network as unreachable, remove it from its routing table, and send RIPv2 messages that indicate the lack of reachability for such network. Lack of reachability in RIPv2 is advertised similar to normal network advertisements except for settting metric to 16. By default, $t_i = t_u \times 6 = 180$.
- Holddown timer t_h – this is a Cisco-specific extension that forbids the router from updating its routing table with better routes with regards to some given network for t_h seconds should the network get advertised as *unreachable*. This is intended to minimize number of routing table updates in cases where too-frequent updates occur (e.g. fluctuating links)[1]. By default, $t_h = 180$ seconds.
- Flush timer t_f — if a router does not receive a RIPv2 update about a given network for t_f seconds, it will completely remove it from its routing table. By default, $t_i = 240$.

2 Lab Preparation

1. Connect a PC to the routers console port using a rollover cable[2].
2. Erase the configuration of the routers[3].
3. Connect a PC to the switches console ports using rollover cables.
4. Erase the configuration of the switch[4]. **Note:** the switches are optional at this stage and can be omitted.
5. Physically connect the lab as depicted in Figure 3.
6. Logically assign[5] the following IP addresses and subnet masks:
    - R1's Gi0/1: 10.0.1.1/24

[1] `http://www.cisco.com/web/techdoc/dc/reference/cli/nxos/commands/rip/timers_basic.html`

[2] If using Linux: `screen /dev/ttySx` were `x` is the serial interfaces ID that is connected to the console cable. If using Windows: Use Hyperterminal or Putty to connect to `COMx` ports.

[3] `enable`, `erase startup-config`, `reload`, and make sure to answer `no` to all yes/no questions while hitting *enter* for all `confirm` prompts.

[4] `enable`, `erase startup-config`, `delete vlan.dat`, `reload`, and answer `no` to all yes/no questions while hitting *enter* for all `confirm` prompts.

[5] For Linux `ifconfig eth0 <ip_addr>/<netmask_bits>`, and for Cisco IOS `enable, config terminal, interface <interface_ID>, ip address <ip_addr> <netmask>, no shutdown`.

- PC1's eth0: 10.0.1.2/24
- R1's Gi0/0: 10.0.12.1/24
- R2's Gi0/0: 10.0.12.2/24
- R2's Gi0/1: 10.0.2.1/24
- PC2's eth0: 10.0.2.2/24

7. Run Wireshark on all involved lab PCs as depicted in Figure 3.

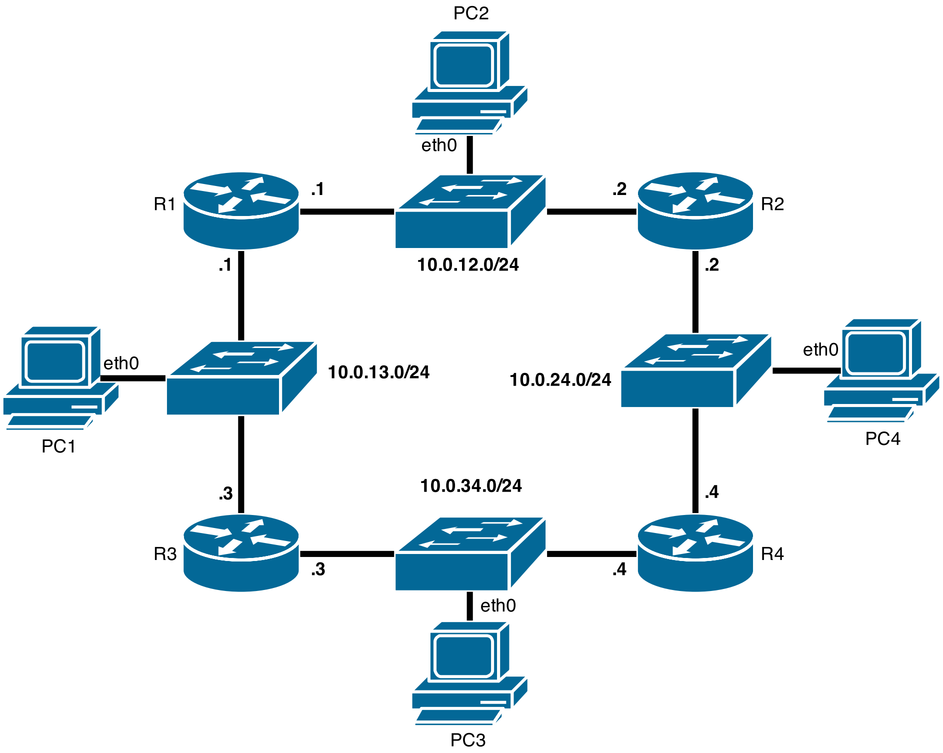

Figure 3: Physical lab topology.

3 Lab experiments

3.1 Configuring RIPv2

Enable RIPv2 on all of the routers' interfaces and networks as follows:

1. `enable`.
2. `config terminal`.
3. `router rip`.
4. `version 2`.
5. `network 0.0.0.0`[6]

[6] This will activate RIPv2 on all interfaces whose IP addresses fall under any network as well as including their network IP addresses in the RIPv2 advertisements.

3.2 Verifying RIPv2

[20 points]

Note: when done, show your work to the lab engineer for grading purposes.

Verify the correctness of the configuration:

- `show ip protocols`.
- `show ip route`.
- Use the `ping` command to test the reachability.

3.3 Analysis of converged RIPv2

[30 points]

Note: when done, show your work to the lab engineer for grading purposes.

While monitoring Wireshark instances on relevant PCs, answer the following questions:

1. What is the inter-RIPv2 update gap in seconds in average? How does this time compare against the output from `show ip protocols`?
2. What are the networks and their corresponding metrics as advertised by RIPv2:
   - From R1 to R2?
   - From R1 to R3?
   - From R2 to R1?
   - From R2 to R2?
   - From R3 to R1?
   - From R3 to R4?
   - From R4 to R2?
   - From R4 to R3?
3. What are the networks that are *not* advertised:
   - From R1 to R2?
   - From R1 to R3?
   - From R2 to R1?
   - From R2 to R2?
   - From R3 to R1?
   - From R3 to R4?
   - From R4 to R2?
   - From R4 to R3?
4. What is the effect of advertising next-hop address of `0.0.0.0` in the RIPv2 updates? You can use the Linux command `traceroute` and the Cisco IOS command `show ip route` as tools to answer this question.
5. How do your previous answers correlate with the output of Cisco IOS command `show ip route`.

3.4 Analysis of converging RIPv2

[50 points]

Note: when done, show your work to the lab engineer for grading purposes.

1. Shutdown the router R1, change PC1's default gateway to point to R3, and while keeping an eye on Wireshark instances, answer the following:
   - Find which PCs cannot successfully ping which other PCs?
   - For those PCs that cannot successfully ping some other PCs, how long did it take the routers (using RIPv2) to converge on the alternative routing setup? Compare this time against those timers in Cisco's IOS command `show ip protocols`.
   - Analyze the captured RIPv2 packets on the Wireshark instances and describe the change that occurred in the content of the RIPv2 messages and how that facilitated the convergence of the network to accommodate the fallen router.

3.5 Extra questions (not graded)

- Inject a spoofed RIPv2 packet such that you either cause a DoS or a MitM attack. Using `macframesender-v2.c` is easy as it calculates the IP checksum for you automatically and the UDP checksum can be disabled by setting its value to `0x0000`.

  The basic idea is using your PC to advertise reachability to a network with a much lower metric than the other routers.

  If a router receives such advertisement while setting yourself as the next-hop, and if your metric is lower than any other advertisement that the router has heard to so far, the router will update its routing table and start forwarding traffic of that network to your PC instead! It's up to you to make a DoS (by dropping them) or a MitM (by re-routing them while observing them or modifying them).
- Logically thinking, list some solutions that, if applied, the above MitM attack would become much harder to happen.

A Relevant protocols

```
 0                   1                   2                   3
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Internet Protocol (IP) version 4 header format — Source RFC791.

```
0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
| Source Port     | Destination Port|
+--------+--------+--------+--------+
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|          data octets ...
+---------------- ...
```

Figure 5: UDP Header Format — Source: RFC768.