Routerlab

Summer semester 2018

Worksheet 7
Group 08

Valentin Franck, 364066
Nikhil Singh, 387694

Pages: 13

Submission Date: June 21, 2018

Question 1

1a

In static routing we manually and permanently configure the routers to send traffic to the particular directions. For instance what we had been doing in all the worksheets since the beginning. However, in dynamic routing we use routing protocols such as RIP, OSPF, BGP, ISIS, EIGRP etc to discover the paths traffic should take.

1b

The main advantages of dynamic routing over static routing are adaptability and scalability. In dynamic routing the network can grow very quickly and whenever the changes occur in the network topology it adapts to them efficiently.

1c

RIP for example uses a request to broadcast (multicast in RIPv2) on all RIP enabled interfaces. All routers with RIP enabled will reply to that request with their routing table.

1d

In RIP when a link fails the adjacent router will notice (after a 180 second timeout) and will not only no longer announce the routes it learned from that link, but it will use Route Poisoning to tell routers on other interfaces to delete the respective routes.

Another event is when RIP is configured on a router, it will start by sending a request message to all interfaces configured to use RIP. The routers receiving such messages will then reply with their full routing table.

In RIP when a router learns a new route it can announce this single route immediately to other routers (triggered updates).

Question 2

2a

All the interfaces which have RIP enabled exchange routing information at regular intervals typically every 30 seconds. Typically these are entities that forward messages (routers).

2b

15 Hops

2c

As per RFC 2453. It keeps the following information about each destination apart from flags and other internal information. Address - It is the IP address of the host or the network. Router - It is the first router along the route to the network. Interface - It is the physical network used to reach the first router. Metric - It is the number that indicates the distance to the destination. Timer - It is the time since the last entry was updated.

2d

RIP message travels up to 15 Hops and within this RIP routing update contains all routes.

2e

Every 30 seconds.

2f

RIP uses an Invalid Timer (default 180 seconds). So once it does not receive an update from a neighbour for 180 seconds it considers the link down.

It propagates this via route poisoning. In this RIP advertises the failed destination to all of its interfaces with metric of 16 which tells all of its neighbours that it no longer has a route to reach this specific network.

2g

1. Invalid Timer and Flush Timer are not updated for 5 minutes.
2. Route poisoning tells the route is not available anymore.

2h

Count to infinity is also known as routing loops. It occurs when an interface goes down or when two routers send updates to each other at the same time. For example, we have three nodes A, B and C. Node A thinks it can reach C via B and the link between B and C goes down and it doesn't send any update about the failure. In this case Node A will transmit data to Node B thinking C is reachable via B but when B receives it. It will send it back to A because it knows the interface is down between B and C. A then check its routing table and send the data back to B which forms an infinite loop.

2i

The two common mechanisms RIP uses to solve count to infinity problem are Route Poisoning and Split Horizon. Split Horizon prohibits the router to advertise the same route on to the interface from which it was learned. And Route poisoning prevents the router from transmitting packets via invalid route which has an infinite route metric.

2j

The advantages that triggered updates offer are it restrains the chance of count to infinity problem, it speeds up convergence and saves link bandwidth because updates include only affected networks and it sends partial updates when a metric change occurs.

2k

Triggered updates reduce the chance of count to infinity problem but do not solve it. While triggered updates are being sent regular updates may also be happening which makes routers which haven't received triggered updates yet to send the wrong information about the routes which do not exist anymore. And even after receiving triggered updates routers may receive the regular update which was being sent to it before the triggered update which causes the formation of loops and leads to count to infinity problem.

2l

The disadvantages of Distance Vector protocols apart from count to infinity are it is slower to converge. It produces more traffic because of regular updates which leads to unnecessary bandwidth exhaustion which in case of larger network causes congestion in the network and of course these protocols only scale for small networks as the number of hops is limited (15 for RIP).

2m

The advantages of distance vector are it is simple to configure and easy to maintain, because they have less message overhead and computational complexity than link state protocols.

Destination	Next-Hop	Cost
10.1.1.2	link	3
10.1.1.3	link	5
10.1.1.4	10.1.1.2	4
10.1.1.5	10.1.1.3	6
10.1.1.6	10.1.1.2	10

Table 1: Routing table for 10.1.1.1

Destination	Next-Hop	Cost
10.1.1.2	link	3
10.1.1.3	link	5
10.1.1.4	10.1.1.2	4
10.1.1.5	10.1.1.3	6
10.1.1.6	10.1.1.2	10
10.1.1.2	link	3
10.1.1.3	link	5
10.1.1.4	10.1.1.2 -> 10.1.1.3	4 -> 9
10.1.1.5	10.1.1.3	6
10.1.1.6	10.1.1.2 -> 10.1.1.3	10 -> 15

Table 2: Evolution of the routing table for 10.1.1.1 (randomly selecting one link when distance is equal).

Question 3

3a

For our solution see Table 1.

3b

For our solutions see Tables 2 and 3. Note that we included the complete tables, but it should be easy enough to see the evolution of the route to 10.1.1.6..

3c

Yes 10.1.1.6 is reachable from 10.1.1.1. The path is 10.1.1.1 > 10.1.1.3 > 10.1.1.5 > 10.1.1.4 > 10.1.1.6

Destination	Next-Hop	Cost
10.1.1.1	link	3
10.1.1.3	link	3
10.1.1.4	link -> 10.1.1.3	1 -> 7
10.1.1.5	10.1.1.3	4
10.1.1.6	10.1.1.4 -> 10.1.1.3	7 -> 13
10.1.1.1	link	3
10.1.1.3	link	3
10.1.1.4	10.1.1.3	7
10.1.1.5	10.1.1.3	4
10.1.1.6	10.1.1.3	13

Table 3: Evolution of the routing table for 10.1.1.2 (randomly selecting one link when distance is equal).

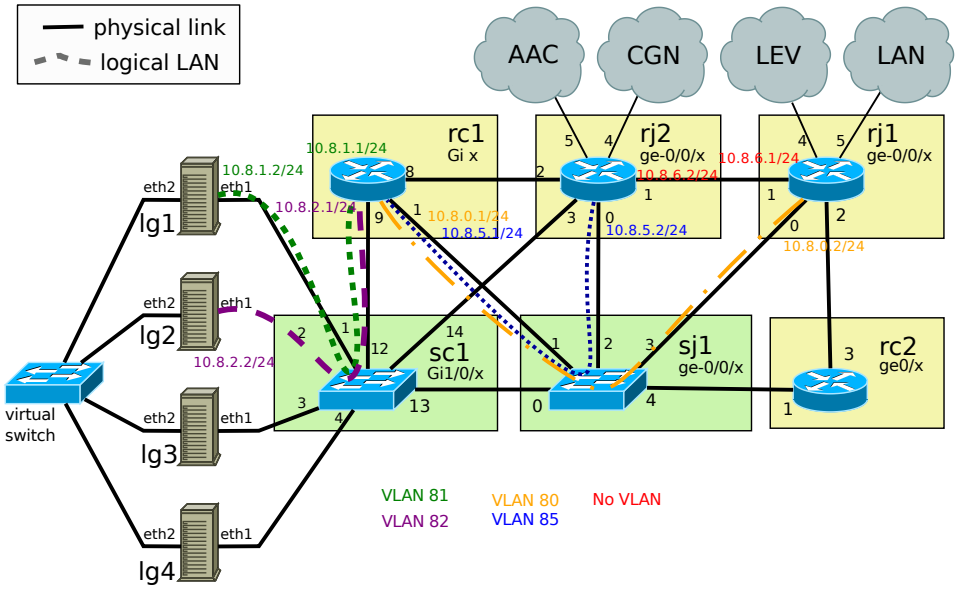

Figure 1: The topology we use for Question 4.

Question 4

We are using the topology as shown in Figure 1.

```
root@group08-lg1:~# ip a a 10.8.1.2/24 dev eth1
root@group08-lg1:~# ip link set up dev eth1
root@group08-lg1:~# ip route add 0.0.0.0/0 via 10.8.1.1 dev eth1
```

We configured lg2 as indicated:

```
root@group08-lg2:~# ip addr add 10.8.2.2/24 dev eth1
root@group08-lg2:~# ip link set up dev eth1
root@group08-lg2:~# ip link add name dummy0 type dummy
RTNETLINK answers: File exists
root@group08-lg2:~# ip addr add 10.8.99.99/24 dev dummy0
root@group08-lg2:~# ip link set dev dummy0 up
root@group08-lg2:~# apt-get install quagga-core quagga-ripd
...
root@group08-lg2:~# cat >/etc/quagga/zebra.conf << EOF
> !
> ! Zebra configuration file
> !
> hostname $(hostname)
> password zebra
> enable password zebra
> !
> EOF
root@group08-lg2:~# cat >/etc/quagga/ripd.conf <<EOF
```

```
!
hostname $(hostname)
password rip
enable password rip
!
router rip
network 10.8.2.0/24
> network 10.8.99.99/24
> !
> EOF
root@group08-lg2:~# systemctl start zebra
root@group08-lg2:~# systemctl start ripd
```

4a

```
lev-rc1#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B -
    BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
           level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
           static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l -
           LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 10 subnets, 2 masks
C        10.8.0.0/24 is directly connected, Vlan80
L        10.8.0.1/32 is directly connected, Vlan80
C        10.8.1.0/24 is directly connected, GigabitEthernet9.81
L        10.8.1.1/32 is directly connected, GigabitEthernet9.81
C        10.8.2.0/24 is directly connected, GigabitEthernet9.82
L        10.8.2.1/32 is directly connected, GigabitEthernet9.82
C        10.8.5.0/24 is directly connected, Vlan85
L        10.8.5.1/32 is directly connected, Vlan85
R        10.8.6.0/24 [120/1] via 10.8.5.2, 00:00:01, Vlan85
                     [120/1] via 10.8.0.2, 00:00:15, Vlan80
R        10.8.99.0/24 [120/1] via 10.8.2.2, 00:00:03, GigabitEthernet9
    .82
```

The last shown route is selected as traceroute proves:

```
lev-rc1#traceroute 10.8.99.99
Type escape sequence to abort.
Tracing the route to 10.8.99.99
VRF info: (vrf in name/id, vrf out name/id)
  1 10.8.99.99 0 msec 0 msec 0 msec
```

4b

```
root@lev-rj2# run show route
```

```
inet.0: 9 destinations, 10 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.8.0.0/24        *[RIP/100] 00:00:52, metric 2, tag 0
                      to 10.8.5.1 via ge-0/0/0.0
                    > to 10.8.6.1 via ge-0/0/1.0
10.8.1.0/24        *[RIP/100] 00:00:52, metric 2, tag 0
                    > to 10.8.5.1 via ge-0/0/0.0
10.8.2.0/24        *[RIP/100] 00:00:52, metric 2, tag 0
                    > to 10.8.5.1 via ge-0/0/0.0
10.8.5.0/24        *[Direct/0] 01:01:38
                    > via ge-0/0/0.0
10.8.5.2/32        *[Local/0] 01:02:56
                      Local via ge-0/0/0.0
10.8.6.0/24        *[Direct/0] 01:02:42
                    > via ge-0/0/1.0
                     [RIP/100] 00:00:52, metric 3, tag 0
                    > to 10.8.5.1 via ge-0/0/0.0
10.8.6.2/32        *[Local/0] 01:02:56
                      Local via ge-0/0/1.0
10.8.99.0/24       *[RIP/100] 00:00:52, metric 3, tag 0
                    > to 10.8.5.1 via ge-0/0/0.0
224.0.0.9/32       *[RIP/100] 00:00:52, metric 1
                      MultiRecv
```

[edit]

The penultimate route is selected as the output from traceroute shows:

```
root@lev-rj2> traceroute 10.8.99.99
traceroute to 10.8.99.99 (10.8.99.99), 30 hops max, 40 byte packets
 1  10.8.5.1 (10.8.5.1)  2.371 ms  1.913 ms  1.684 ms
 2  10.8.99.99 (10.8.99.99)  2.716 ms  2.693 ms  2.264 ms
```

4c

From rc1 output

```
R        10.8.6.0/24 [120/1] via 10.8.0.2, 00:00:15, Vlan80
```

the metric is 1 as [120/1] here indicates 120 as an administrative distance and 1 as the metric.

The metric for this route is 3 from rj2 as mentioned in the output

```
10.8.99.0/24       *[RIP/100] 00:00:52, metric 3, tag 0
                    > to 10.8.5.1 via ge-0/0/0.0
```

Metric is the number that indicates the distance to the destination.

4d

```
root@group08-lg1:~# traceroute 10.8.1.1
traceroute to 10.8.1.1 (10.8.1.1), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  1.105 ms * *
root@group08-lg1:~# traceroute 10.8.2.1
traceroute to 10.8.2.1 (10.8.2.1), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  1.010 ms * *
root@group08-lg1:~# traceroute 10.8.2.2
traceroute to 10.8.2.2 (10.8.2.2), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  0.903 ms  0.954 ms  0.916 ms
 2  10.8.2.2 (10.8.2.2)  1.128 ms  1.092 ms  1.006 ms
```

```
root@group08-lg1:~# traceroute 10.8.99.99
traceroute to 10.8.99.99 (10.8.99.99), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  0.749 ms  0.728 ms  0.743 ms
 2  10.8.99.99 (10.8.99.99)  0.916 ms  0.883 ms  0.838 ms
root@group08-lg1:~# traceroute 10.8.0.1
traceroute to 10.8.0.1 (10.8.0.1), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  1.141 ms * *
root@group08-lg1:~# traceroute 10.8.0.2
traceroute to 10.8.0.2 (10.8.0.2), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  0.977 ms  0.933 ms  0.907 ms
 2  10.8.0.2 (10.8.0.2)  3.802 ms  3.771 ms  3.737 ms
root@group08-lg1:~# traceroute 10.8.5.2
traceroute to 10.8.5.2 (10.8.5.2), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  0.826 ms  0.830 ms  0.818 ms
 2  10.8.5.2 (10.8.5.2)  4.497 ms  4.473 ms  4.434 ms
root@group08-lg1:~# traceroute 10.8.5.1
traceroute to 10.8.5.1 (10.8.5.1), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  1.189 ms * *
root@group08-lg1:~# traceroute 10.8.6.1
traceroute to 10.8.6.1 (10.8.6.1), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  0.867 ms  0.834 ms  0.856 ms
 2  10.8.6.1 (10.8.6.1)  2.922 ms  2.893 ms  2.861 ms
root@group08-lg1:~# traceroute 10.8.6.2
traceroute to 10.8.6.2 (10.8.6.2), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  0.957 ms  0.917 ms  0.881 ms
 2  10.8.6.2 (10.8.6.2)  3.489 ms  3.450 ms  3.411 ms

root@group08-lg2:~# traceroute 10.8.2.1
traceroute to 10.8.2.1 (10.8.2.1), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  1.113 ms * *
root@group08-lg2:~# traceroute 10.8.99.99
traceroute to 10.8.99.99 (10.8.99.99), 30 hops max, 60 byte packets
 1  10.8.99.99 (10.8.99.99)  0.052 ms  0.026 ms  0.014 ms
root@group08-lg2:~# traceroute 10.8.1.1
traceroute to 10.8.1.1 (10.8.1.1), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  1.062 ms * *
root@group08-lg2:~# traceroute 10.8.1.2
traceroute to 10.8.1.2 (10.8.1.2), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  0.762 ms  0.652 ms  0.721 ms
 2  10.8.1.2 (10.8.1.2)  1.030 ms  1.007 ms  0.964 ms
root@group08-lg2:~# traceroute 10.8.0.2
traceroute to 10.8.0.2 (10.8.0.2), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  0.855 ms  0.791 ms  0.804 ms
 2  10.8.0.2 (10.8.0.2)  3.449 ms  3.381 ms  3.359 ms
root@group08-lg2:~# traceroute 10.8.0.1
traceroute to 10.8.0.1 (10.8.0.1), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  1.037 ms * *
root@group08-lg2:~# traceroute 10.8.5.1
traceroute to 10.8.5.1 (10.8.5.1), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  1.058 ms * *
root@group08-lg2:~# traceroute 10.8.5.2
traceroute to 10.8.5.2 (10.8.5.2), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  0.814 ms  0.822 ms  0.780 ms
 2  10.8.5.2 (10.8.5.2)  3.436 ms  3.381 ms  3.356 ms
root@group08-lg2:~# traceroute 10.8.6.2
traceroute to 10.8.6.2 (10.8.6.2), 30 hops max, 60 byte packets
```

```
 1  10.8.2.1 (10.8.2.1)  0.845 ms  0.791 ms  0.789 ms
 2  10.8.0.2 (10.8.0.2)  1.069 ms  1.029 ms  0.991 ms
 3  10.8.6.2 (10.8.6.2)  3.525 ms  3.468 ms  3.436 ms
root@group08-lg2:~# traceroute 10.8.6.1
traceroute to 10.8.6.1 (10.8.6.1), 30 hops max, 60 byte packets
 1  10.8.2.1 (10.8.2.1)  0.825 ms  0.777 ms  0.789 ms
 2  10.8.5.2 (10.8.5.2)  1.018 ms  0.983 ms  0.934 ms
 3  10.8.6.1 (10.8.6.1)  3.432 ms  3.378 ms  3.346 ms
```

4e

Yes the count-to-infinity problem occurs if the link betwen rc1 and sc1 fails. In that case it will happen at rj1 and rj2. Because they announced the routes to each other, after rc1 announces, that it cannot provide the route anymore, they will use each others routes (which are longer) and therefore will count to infinity.

4f

We prepare both rj1 and lg2 as explained:

```
root@lev-rj1# set protocols rip traceoptions file mytrace

[edit]
root@lev-rj1# set protocols rip traceoptions flag route

[edit]
root@lev-rj1# commit
commit complete

root@group08-lg2:~# iptables -A OUTPUT -o eth1 -j DROP
```

We started to drop packets after the log message with the timestamp 22:12:47.780821 appeared in the log file.

It took about three minutes until even rc1 noticed that the link was down, but once rc1 considered the route to be down it immediately announced that to rj1, which deleted the route:

```
Jun 19 22:15:46.566187  10.8.99.0/24: metric-in: 16, change: 3 -> 16; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:46.567066 CHANGE   10.8.99.0/24         nhid 560 gw
    10.8.0.1         RIP       pref 100/0 metric 3/0 ge-0/0/0.0 <Delete
    Int>
Jun 19 22:15:46.567156 Best route to 10.8.99.0/24 got deleted. Doing
    route calculation on the stored rte-info
```

We checked rc1 manually every few seconds and when it said "Route is possibly down", we immediately noticed the Release message in rj1. So the time was 3 minutes to delete the route, because the Count to infinity problem did not occur. (rj1 and rj2 probably do not announce the route, they learned from rc1 back to rc1.)

However we can see that rj1 also is announced the route to 10.8.99.0/24 by rj2. Still it does not use this route after rc1 sets the route to 16=infinity.

Here we provide the monitor log from learning the route until releasing it. Note that we started dropping the output from lg2 a few seconds after rj1 learned the route (at about 22:12:48):

```
Jun 19 22:12:13.527806  Adding 10.8.99.0/24: nh 10.8.0.1; met 3; tag 0
Jun 19 22:12:13.528656 CHANGE   10.8.99.0/24         nhid 560 gw
    10.8.0.1         RIP       pref 100/0 metric 3/0 ge-0/0/0.0 <Active
    Int>
Jun 19 22:12:13.528777 ADD      10.8.99.0/24         nhid 560 gw
    10.8.0.1         RIP       pref 100/0 metric 3/0 ge-0/0/0.0 <Active
    Int>
```

```
Jun 19 22:12:13.528886 rt_close: 1 route proto RIPv2
Jun 19 22:12:13.528886
Jun 19 22:12:13.529182 rt_flash_update_callback: flash RIPv2 (inet.0)
    start
Jun 19 22:12:13.529294 Setting RIPv2 rtbit on route 10.8.99.0/24, tsi =
     0x1550ce8
Jun 19 22:12:13.529383 Rip is triggering for route 10.8.99.0.
Jun 19 22:12:13.529491 rt_flash_update_callback: flash RIPv2 (inet.0)
    done
Jun 19 22:12:16.592323  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:12:21.075861  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:12:21.076612  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:12:21.076733  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:12:21.076853  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:12:41.247823  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:12:41.248311  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:12:41.248433  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:12:41.248546  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:12:47.780033  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:12:47.780508  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:12:47.780701  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:12:47.780821  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:09.179489  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:13:09.180217  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:13:09.180331  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:13:09.180444  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:13:13.304402  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:13.304883  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:13.305007  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:13.305126  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:37.514679  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:13:37.515174  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:13:37.515287  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
```

```
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:13:37.515399  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:13:41.824712  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:41.825173  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:41.825555  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:13:41.825681  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:04.682394  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:14:04.682881  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:14:04.683004  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:14:04.683115  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:14:08.301518  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:08.302006  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:08.302124  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:08.302242  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:33.872206  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:14:33.872698  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:14:33.872809  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:14:33.872920  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:14:37.297333  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:37.298170  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:37.298286  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:14:37.298479  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:03.099178  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:15:03.099677  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:15:03.099790  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:15:03.099902  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:15:03.269636  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:03.269791  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:03.270392  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
```

```
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:03.270514  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:28.813999  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:28.814499  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:28.814618  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:28.814736  10.8.99.0/24: metric-in: 3, change: 3 -> 3; #
    gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:32.244241  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:15:32.244724  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:15:32.244835  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:15:32.244994  10.8.99.0/24: metric-in: 4, change: 3 -> 4; #
    gw: 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:15:46.566187  10.8.99.0/24: metric-in: 16, change: 3 -> 16; #
     gw: 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:46.567066 CHANGE   10.8.99.0/24         nhid 560 gw
    10.8.0.1        RIP       pref 100/0 metric 3/0 ge-0/0/0.0 <Delete
    Int>
Jun 19 22:15:46.567156 Best route to 10.8.99.0/24 got deleted. Doing
    route calculation on the stored rte-info
Jun 19 22:15:46.567249 rt_close: 1 route proto RIPv2
Jun 19 22:15:46.567249
Jun 19 22:15:46.567522 rt_flash_update_callback: flash RIPv2 (inet.0)
    start
Jun 19 22:15:46.567647 rt_flash_update_callback: flash RIPv2 (inet.0)
    done
Jun 19 22:15:57.874219  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:57.874710  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:57.874830  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:15:57.874940       Accept worse metric for route from same
    subnet.
Jun 19 22:15:57.874992       Resurrected or better route.
Jun 19 22:16:00.443086  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:16:00.443576  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:16:00.443688  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:16:27.703593  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:16:27.704090  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:16:27.704206  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:16:27.704354       Accept worse metric for route from same
    subnet.
Jun 19 22:16:27.704423       Resurrected or better route.
```

```
Jun 19 22:16:27.742023  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:16:27.742176  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:16:27.742288  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:16:55.678805  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:16:55.679303  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:16:55.679423  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:16:56.872016  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:16:56.872496  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:16:56.872618  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:17:21.595089  10.8.1.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:17:21.595583  10.8.2.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 1, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:17:21.595700  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.0.1, inx: 0, rte_upd_src 10.8.0.1
Jun 19 22:17:25.092942  10.8.5.0/24: metric-in: 2, change: 2 -> 2; # gw
    : 2, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 10.8.6.2
Jun 19 22:17:25.093419  10.8.1.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:17:25.093542  10.8.2.0/24: metric-in: 3, change: 2 -> 3; # gw
    : 1, pkt_upd_src 10.8.6.2, inx: 1, rte_upd_src 0.0.0.0
Jun 19 22:17:46.598660 RELEASE  10.8.99.0/24          nhid 560 gw
    10.8.0.1
```

4g

We can see that Route poisoning is working, because rc1 tells rj1 that the route to 10.8.99.0/24 is not available any longer. We can also see how split horizon works, because rj1 does not announce the route back to rc1 (from which it learned it). But still the route is announced by rj2 on interface 1. Since we only keep the best routes in our routing table and the one announced by rj2 is one hop more we do not save it. Therefore the count to infinity does not occur. (There is a minimal chance it occurs, if rj1 deletes the route before rj2 does so (for whatever reason) and then rj2 announces the route, so rj1 adds a new route. Then rj2 would probably delete the route via rc1, but rj1 would announce the route learned from rj2 to rc1 and so on. This is highly unlikely and will probably not count all the way to infinity, because Route poisoning will be used again.)

Included Files

q04-config-rc1.txt, q04-config-rj2.txt, q04-config-sj1.txt, q04-config-rj1.txt, q04-config-sc1.txt