Отчёт по лабораторной работе «Динамическая IP-маршрутизация»

Федоров П.В.

12 октября 2018 г.

Содержание

1. Настройка сети **1**
1.1. Топология сети 1
1.2. Назначение IP-адресов 3
1.3. Настройка протокола RIP 5

2. Проверка настройки протокола RIP **7**
2.1. Вывод сообщения RIP 7

3. Расщепленный горизонт и испорченные обратные обновления **8**
3.1. On: Split-horizon, Poisoned-reverse 9
3.2. Off: split-horizon, poisoned-reverse 9

4. Имитация устранимой поломки в сети **10**

5. Имитация неустранимой поломки в сети **11**

1. Настройка сети

1.1. Топология сети

Топология сети и используемые IP-адреса показаны на рисунке 1.
Перечень узлов, на которых используется динамическая IP-маршрутизация:

1) маршрутизатор **r1**
2) маршрутизатор **r2**
3) маршрутизатор **r3**
4) маршрутизатор **r4**
5) маршрутизатор **r5**

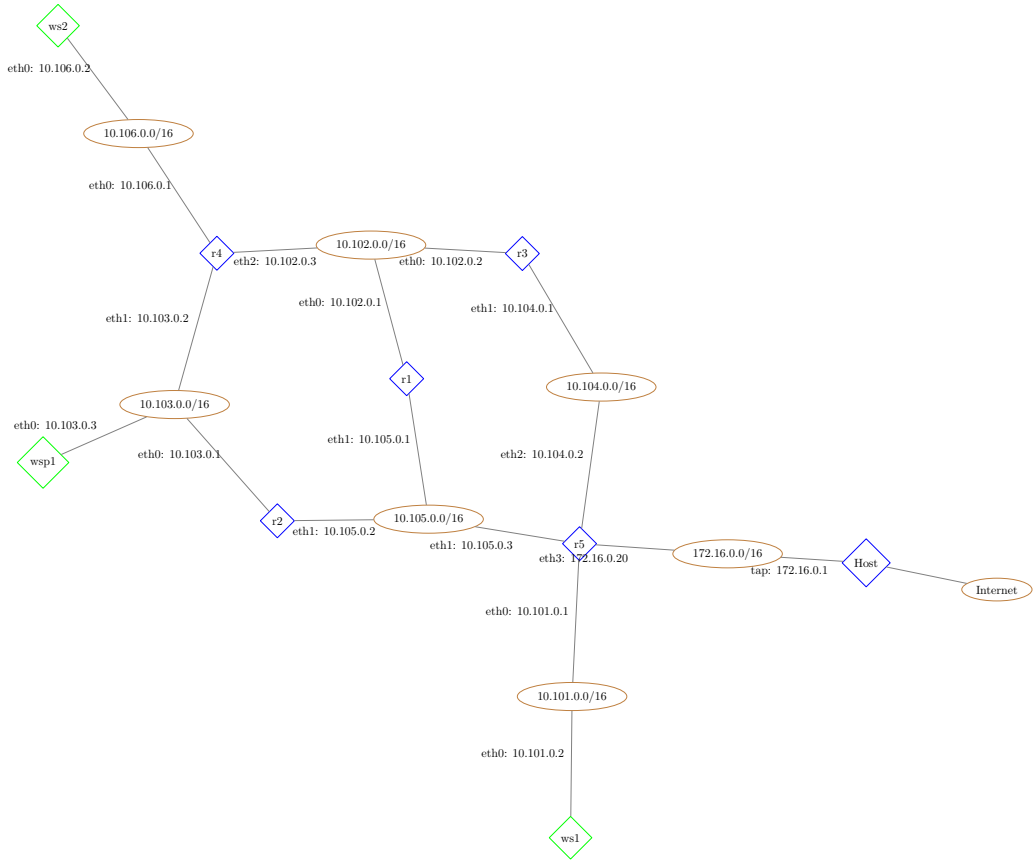

Рис. 1. Топология сети

1.2. Назначение IP-адресов

Сетевые настройки маршрутизатора **r1**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.102.0.1
netmask 255.255.0.0

auto eth1
iface eth1 inet static
address 10.105.0.1
netmask 255.255.0.0
```

Сетевые настройки маршрутизатора **r2**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.103.0.1
netmask 255.255.0.0

auto eth1
iface eth1 inet static
address 10.105.0.2
netmask 255.255.0.0
```

Сетевые настройки маршрутизатора **r3**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.102.0.2
netmask 255.255.0.0

auto eth1
iface eth1 inet static
address 10.104.0.1
netmask 255.255.0.0
```

Сетевые настройки маршрутизатора **r4**.

```
auto lo
iface lo inet loopback
```

```
auto eth0
iface eth0 inet static
address 10.106.0.1
netmask 255.255.0.0

auto eth1
iface eth1 inet static
address 10.103.0.2
netmask 255.255.0.0

auto eth2
iface eth2 inet static
address 10.102.0.3
netmask 255.255.0.0
```

Сетевые настройки маршрутизатора **r5**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.101.0.1
netmask 255.255.0.0

auto eth1
iface eth1 inet static
address 10.105.0.3
netmask 255.255.0.0

auto eth2
iface eth2 inet static
address 10.104.0.2
netmask 255.255.0.0
```

Сетевые настройки рабочей станции **ws1**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.101.0.2
netmask 255.255.0.0
gateway 10.101.0.1
```

Сетевые настройки рабочей станции **ws2**.

```
auto lo
iface lo inet loopback
```

```
auto eth0
iface eth0 inet static
address 10.106.0.2
netmask 255.255.0.0
gateway 10.106.0.1
```

Сетевые настройки рабочей станции **wsp1**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.103.0.3
netmask 255.255.0.0
```

1.3. Настройка протокола RIP

Ниже приведен файл `/etc/quagga/ripd.conf` маршрутизатора **r1**.

```
router rip

network eth0
network eth1

timers basic 10 60 120

redistribute kernel
redistribute connected

log file /var/log/quagga/ripd.log
```

Файл `/etc/quagga/ripd.conf` маршрутизатора **r2**.

```
router rip

network eth0
network eth1

timers basic 10 60 120

redistribute kernel
redistribute connected

log file /var/log/quagga/ripd.log
```

Файл `/etc/quagga/ripd.conf` маршрутизатора **r3**.

```
router rip

network eth0
network eth1

timers basic 10 60 120

redistribute kernel
redistribute connected

log file /var/log/quagga/ripd.log
```

Файл `/etc/quagga/ripd.conf` маршрутизатора **r4**.

```
router rip

network eth0
network eth1
network eth2

timers basic 10 60 120

redistribute kernel
redistribute connected

log file /var/log/quagga/ripd.log
```

Файл `/etc/quagga/ripd.conf` маршрутизатора **r5** связанный с интернетом.

```
router rip

network eth0
network eth1
network eth2

timers basic 10 60 120

redistribute kernel
! redistribute connected

log file /var/log/quagga/ripd.log
```

Ниже приведен файл `/etc/quagga/ripd.conf` рабочий станции, связанной с несколькими маршрутизаторами **wsp1**.

```
router rip

network eth0

timers basic 10 60 120
```

```
redistribute kernel
redistribute connected

log file /var/log/quagga/ripd.log
```

2. Проверка настройки протокола RIP

Вывод **traceroute** от узла **ws1** до **ws2** при нормальной работе сети.

```
traceroute -n 10.106.0.2
traceroute to 10.106.0.2 (10.106.0.2), 64 hops max, 40 byte packets
    1  10.101.0.1  7 ms  0 ms  0 ms
    2  10.105.0.2  0 ms  0 ms  0 ms
    3  10.103.0.2  0 ms  0 ms  0 ms
    4  10.106.0.2  0 ms  1 ms  0 ms
```

Вывод **traceroute** от узла **ws2** до **wsp1** при нормальной работе сети.

```
traceroute -n 10.103.0.3
traceroute to 10.103.0.3 (10.103.0.3), 64 hops max, 40 byte packets
    1  10.106.0.1  0 ms  0 ms  0 ms
    2  10.103.0.3  2 ms  0 ms  0 ms
```

Вывод **traceroute** от узла **ws2** до внешнего IP **10.0.2.2**.

```
traceroute to 10.0.2.2 (10.0.2.2), 64 hops max, 40 byte packets
1  10.106.0.1  8 ms  0 ms  0 ms
2  10.102.0.2  11 ms  0 ms  0 ms
3  10.105.0.3  11 ms  1 ms  1 ms
4  172.16.0.1  8 ms  1 ms  6 ms
5  10.0.2.2  2 ms  2 ms  1 ms
```

2.1. Вывод сообщения RIP

Вывод осуществлялся с помощью команды

```
tcpdump -tnv -i ethNUM -s 1518 udp
```

Вывод сообщений RIP на маршрутизаторе **r2** с сетевого интерфейса **eth1**

```
    IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 72) 10.105.0.
RIPv2, Response, length: 44, routes: 2
  AFI: IPv4:      10.103.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 2, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 112) 10.105.0.3.5
RIPv2, Response, length: 84, routes: 4
  AFI: IPv4:         0.0.0.0/0 , tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.101.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.102.0.0/16, tag 0x0000, metric: 2, next-hop: self
```

```
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 1, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 92) 10.105.0.1.52
RIPv2, Response, length: 64, routes: 3
  AFI: IPv4:      10.102.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 2, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 72) 10.105.0.2.52
RIPv2, Response, length: 44, routes: 2
  AFI: IPv4:      10.103.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 2, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 112) 10.105.0.3.5
RIPv2, Response, length: 84, routes: 4
  AFI: IPv4:         0.0.0.0/0 , tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.101.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.102.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 1, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 92) 10.105.0.1.52
RIPv2, Response, length: 64, routes: 3
  AFI: IPv4:      10.102.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 2, next-hop: self
```

Вывод таблицы RIP на **r2**.

```
       Network         Next Hop         Metric From            Tag Time
R(n) 0.0.0.0/0         10.105.0.3            2 10.105.0.3       0 00:56
R(n) 10.101.0.0/16     10.105.0.3            2 10.105.0.3       0 00:56
R(n) 10.102.0.0/16     10.105.0.1            2 10.105.0.1       0 00:55
C(i) 10.103.0.0/16     0.0.0.0               1 self             0
R(n) 10.104.0.0/16     10.105.0.3            2 10.105.0.3       0 00:56
C(i) 10.105.0.0/16     0.0.0.0               1 self             0
R(n) 10.106.0.0/16     10.103.0.2            2 10.103.0.2       0 00:51
```

Вывод таблицы маршрутизации.

```
10.101.0.0/16 via 10.105.0.3 dev eth1  proto zebra  metric 2
10.103.0.0/16 dev eth0  proto kernel  scope link  src 10.103.0.1
10.102.0.0/16 via 10.105.0.1 dev eth1  proto zebra  metric 2
10.105.0.0/16 dev eth1  proto kernel  scope link  src 10.105.0.2
10.104.0.0/16 via 10.105.0.3 dev eth1  proto zebra  metric 2
10.106.0.0/16 via 10.103.0.2 dev eth0  proto zebra  metric 2
default via 10.105.0.3 dev eth1  proto zebra  metric 2
```

3. Расщепленный горизонт и испорченные обратные обновления

Рассмотрим несколько выводов на одном маршрутизаторе с различной конфигурацией поддержки протокола **RIP** В качестве маршрутизатора для эксперимента выступит маршрутизатор **r2**

3.1. On: Split-horizon, Poisoned-reverse

Для включения правил изменим **ripd.conf** на маршрутизаторе **r2**

```
interface eth0
ip rip split-horizon poisoned-reverse
```

Вывод **tcpdump**

```
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 132) 10.103.0.2.5
RIPv2, Response, length: 104, routes: 5
  AFI: IPv4:         0.0.0.0/0 , tag 0x0000, metric: 3, next-hop: self
  AFI: IPv4:      10.101.0.0/16, tag 0x0000, metric: 3, next-hop: self
  AFI: IPv4:      10.102.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 1, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 172) 10.103.0.1.5
RIPv2, Response, length: 144, routes: 7
  AFI: IPv4:         0.0.0.0/0 , tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.101.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.102.0.0/16, tag 0x0000, metric: 16, next-hop: 10.103.0.2
  AFI: IPv4:      10.103.0.0/16, tag 0x0000, metric: 16, next-hop: self
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.105.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 16, next-hop: 10.103.0.2
```

Отметим, что маршрутизатор **r2** отправляет пакеты с указанием сетей полученных из данной сети (10.103.0.0/16) метрикой 16

3.2. Off: split-horizon, poisoned-reverse

Для включения правил изменим **ripd.conf** на маршрутизаторе **r2**

```
interface eth0
no ip rip split-horizon poisoned-reverse
```

Вывод **tcpdump**

```
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 112) 10.103.0.1.5
RIPv2, Response, length: 84, routes: 4
  AFI: IPv4:         0.0.0.0/0 , tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.101.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.105.0.0/16, tag 0x0000, metric: 1, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 132) 10.103.0.2.5
RIPv2, Response, length: 104, routes: 5
  AFI: IPv4:         0.0.0.0/0 , tag 0x0000, metric: 3, next-hop: self
  AFI: IPv4:      10.101.0.0/16, tag 0x0000, metric: 3, next-hop: self
  AFI: IPv4:      10.102.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.104.0.0/16, tag 0x0000, metric: 2, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 1, next-hop: self
```

Отметим, что маршрутизатор **r2** отправляет пакеты с указанием сетей полученных из данной сети (10.103.0.0/16), но уже без указания пометки, что данная сеть достижима с помощью маршрутизаторов сети, в которую происходит групповая передача данных

4. Имитация устранимой поломки в сети

В данном эксперименте будем задействовать 3 маршрутизатора

1) маршрутизатор **r2** - отключаемый

2) маршрутизатор **r3** - отключаемый

3) маршрутизатор **r5** - для мониторинга устранимой поломки сети

Проверим текущий путь от **ws1** до **ws2** не задействует маршрутизатор **r5** (пакет не проходит **10.105.0.0/16**)

```
traceroute to 10.106.0.2 (10.106.0.2), 64 hops max, 40 byte packets
1  10.101.0.1  2 ms  0 ms  0 ms
2  10.104.0.1  11 ms  0 ms  0 ms
3  10.102.0.3  17 ms  1 ms  1 ms
4  10.106.0.2  17 ms  1 ms  1 ms
```

Для чистоты эксперимента и направлении трафика через единственный возможный путь выключим маршрутизаторы **r2** и **r3** Таким образом останется единственный путь от **ws1** до **ws2** через **r5**

Перед отключением указанных маршрутизаторов рассмторим таблицу RIP на **r5**

```
K(r) 0.0.0.0/0          172.16.0.1          1 self              0
C(i) 10.101.0.0/16      0.0.0.0             1 self              0
R(n) 10.102.0.0/16      10.105.0.1          2 10.105.0.1        0 00:57
R(n) 10.103.0.0/16      10.105.0.2          2 10.105.0.2        0 00:52
C(i) 10.104.0.0/16      0.0.0.0             1 self              0
C(i) 10.105.0.0/16      0.0.0.0             1 self              0
R(n) 10.106.0.0/16      10.104.0.1          3 10.104.0.1        0 00:53
```

Отключим сетевые интерфейсы на маршрутизаторах **r2 r3** (в случае отключения начинаются баги)

Мониторим таблицу

Перед исчезновением

```
     Network            Next Hop         Metric From            Tag Time
K(r) 0.0.0.0/0          172.16.0.1            1 self              0
C(i) 10.101.0.0/16      0.0.0.0               1 self              0
R(n) 10.102.0.0/16      10.105.0.1            2 10.105.0.1        0 00:51
R(n) 10.103.0.0/16      10.105.0.2            2 10.105.0.2        0 00:03
C(i) 10.104.0.0/16      0.0.0.0               1 self              0
C(i) 10.105.0.0/16      0.0.0.0               1 self              0
R(n) 10.106.0.0/16      10.104.0.1            3 10.104.0.1        0 00:00
```

Один из маршрутов invalid

```
     Network            Next Hop         Metric From            Tag Time
K(r) 0.0.0.0/0          172.16.0.1            1 self              0
C(i) 10.101.0.0/16      0.0.0.0               1 self              0
R(n) 10.102.0.0/16      10.105.0.1            2 10.105.0.1        0 00:51
R(n) 10.103.0.0/16      10.105.0.2            2 10.105.0.2        0 00:03
C(i) 10.104.0.0/16      0.0.0.0               1 self              0
C(i) 10.105.0.0/16      0.0.0.0               1 self              0
R(n) 10.106.0.0/16      10.104.0.1           16 10.104.0.1        0 02:00
```

Найден новый путь

```
     Network            Next Hop         Metric From            Tag Time
K(r) 0.0.0.0/0          172.16.0.1            1 self              0
C(i) 10.101.0.0/16      0.0.0.0               1 self              0
R(n) 10.102.0.0/16      10.105.0.1            2 10.105.0.1        0 00:58
R(n) 10.103.0.0/16      10.105.0.2           16 10.105.0.2        0 02:00
C(i) 10.104.0.0/16      0.0.0.0               1 self              0
C(i) 10.105.0.0/16      0.0.0.0               1 self              0
R(n) 10.106.0.0/16      10.105.0.1            3 10.105.0.1        0 00:58
```

Вывод **traceroute** от узла **ws1** до **ws2** после того, как служба RIP перестроила таблицы маршрутизации.

```
traceroute to 10.106.0.2 (10.106.0.2), 64 hops max, 40 byte packets
1  10.101.0.1  0 ms  0 ms  0 ms
2  10.105.0.1  0 ms  0 ms  0 ms
3  10.102.0.3  0 ms  0 ms  0 ms
4  10.106.0.2  0 ms  1 ms  1 ms
```

5. Имитация неустранимой поломки в сети

Для данного эксперимента будем прерывать соединения на маршрутизаторе **r4** тем самым сегмент сети с адресом **10.106.0.0/16** станет недостижимым.

Рассмотрим таблицу RIP на маршрутизаторе **r3** до отключения **r4**

```
     Network            Next Hop         Metric From            Tag Time
R(n) 0.0.0.0/0          10.104.0.2            2 10.104.0.2        0 00:53
R(n) 10.101.0.0/16      10.104.0.2            2 10.104.0.2        0 00:53
C(i) 10.102.0.0/16      0.0.0.0               1 self              0
R(n) 10.103.0.0/16      10.102.0.3            2 10.102.0.3        0 00:52
C(i) 10.104.0.0/16      0.0.0.0               1 self              0
R(n) 10.105.0.0/16      10.104.0.2            2 10.104.0.2        0 00:53
R(n) 10.106.0.0/16      10.102.0.3            2 10.102.0.3        0 00:52
```

запустим на **r1** tcpdump для отслеживания сообщений RIP протокола и отключим **r4**

После отключения наблюдаем следующую RIP таблицу на **r3**

```
     Network            Next Hop         Metric From            Tag Time
R(n) 0.0.0.0/0          10.104.0.2            2 10.104.0.2        0 00:52
R(n) 10.101.0.0/16      10.104.0.2            2 10.104.0.2        0 00:52
C(i) 10.102.0.0/16      0.0.0.0               1 self              0
R(n) 10.103.0.0/16      10.104.0.2            3 10.104.0.2        0 00:52
C(i) 10.104.0.0/16      0.0.0.0               1 self              0
R(n) 10.105.0.0/16      10.104.0.2            2 10.104.0.2        0 00:52
R(n) 10.106.0.0/16      10.102.0.3           16 10.102.0.3        0 01:47
```

Так же отметим что происходило на **r1**

```
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 52) 10.102.0.3.52
RIPv2, Response, length: 24, routes: 1
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 16, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 72) 10.102.0.3.52
RIPv2, Response, length: 44, routes: 2
  AFI: IPv4:      10.103.0.0/16, tag 0x0000, metric: 1, next-hop: self
  AFI: IPv4:      10.106.0.0/16, tag 0x0000, metric: 16, next-hop: self
IP (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length 52) 10.102.0.3.52
RIPv2, Response, length: 24, routes: 1
  AFI: IPv4:      10.103.0.0/16, tag 0x0000, metric: 16, next-hop: self
```

Отметим что сеть **10.106.0.0/16** недостижима о чем свидетельствует метрика 16 в 1 RIP ответе

Ниже приведена таблица RIP на **r1**

```
     Network            Next Hop         Metric From            Tag Time
R(n) 0.0.0.0/0          10.105.0.3            2 10.105.0.3        0 00:57
R(n) 10.101.0.0/16      10.105.0.3            2 10.105.0.3        0 00:57
C(i) 10.102.0.0/16      0.0.0.0               1 self              0
R(n) 10.103.0.0/16      10.105.0.2            2 10.105.0.2        0 00:51
R(n) 10.104.0.0/16      10.105.0.3            2 10.105.0.3        0 00:57
C(i) 10.105.0.0/16      0.0.0.0               1 self              0
```

И таблица маршрутизации

```
10.101.0.0/16 via 10.105.0.3 dev eth1  proto zebra  metric 2
10.103.0.0/16 via 10.105.0.2 dev eth1  proto zebra  metric 2
10.102.0.0/16 dev eth0  proto kernel  scope link  src 10.102.0.1
10.105.0.0/16 dev eth1  proto kernel  scope link  src 10.105.0.1
10.104.0.0/16 via 10.105.0.3 dev eth1  proto zebra  metric 2
default via 10.105.0.3 dev eth1  proto zebra  metric 2
```