Отчёт по лабораторной работе «IP-маршрутизация»

Доржсурэн Одсэлмаа

28 сентября 2017 г.

Содержание

1. Топология сети

Топология сети и используемые IP-адреса показаны на рис. 1.

2. Назначение IP-адресов

Ниже приведён файл настройки протокола IP маршрутизатора **r1**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.0.40.1
netmask 255.255.255.0
```

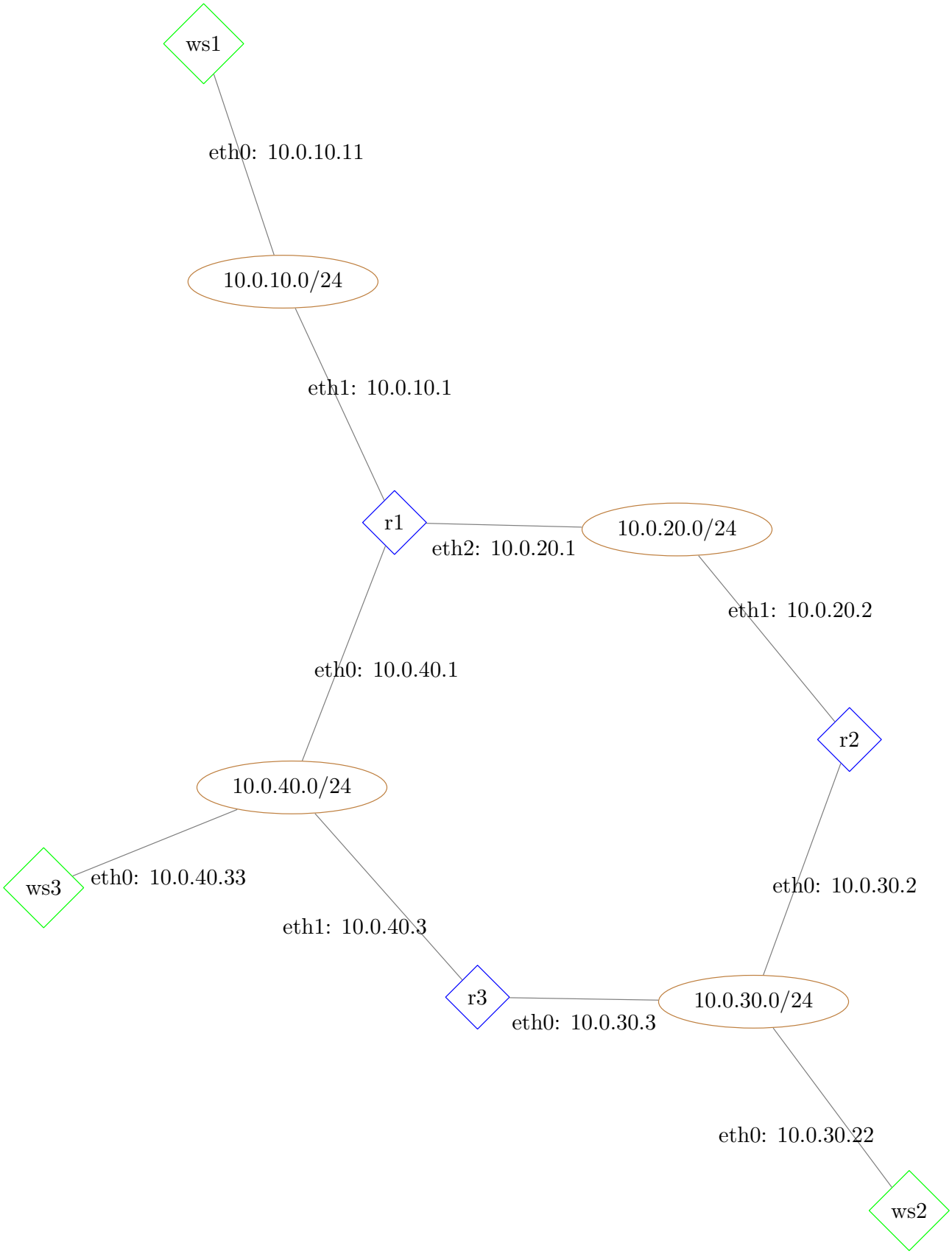

Рис. 1. Топология сети

```
auto eth1
iface eth1 inet static
address 10.0.10.1
netmask 255.255.255.0

auto eth2
iface eth2 inet static
address 10.0.20.1
netmask 255.255.255.0
up ip r add 10.0.30.0/24 via 10.0.20.2 dev eth2
down ip r del 10.0.30.0/24
```

Ниже приведён файл настройки протокола IP рабочей станции **ws1**.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.0.10.11
netmask 255.255.255.0
gateway 10.0.10.1
```

3. Таблица маршрутизации

Вывести (командой ip r) таблицу маршрутизации для **r1**.

```
10.0.20.0/24 dev eth2  proto kernel  scope link  src 10.0.20.1
10.0.30.0/24 via 10.0.20.2 dev eth2
10.0.40.0/24 dev eth0  proto kernel  scope link  src 10.0.40.1
10.0.10.0/24 dev eth1  proto kernel  scope link  src 10.0.10.1
```

Вывести (командой ip r) таблицу маршрутизации для **r2**.

```
10.0.20.0/24 dev eth1  proto kernel  scope link  src 10.0.20.2
10.0.30.0/24 dev eth0  proto kernel  scope link  src 10.0.30.2
10.0.40.0/24 via 10.0.30.3 dev eth0
10.0.10.0/24 via 10.0.20.1 dev eth1
```

Вывести (командой ip r) таблицу маршрутизации для **r3**.

```
10.0.20.0/24 via 10.0.30.2 dev eth0
10.0.30.0/24 dev eth0  proto kernel  scope link  src 10.0.30.3
10.0.40.0/24 dev eth1  proto kernel  scope link  src 10.0.40.3
10.0.10.0/24 via 10.0.40.1 dev eth1
```

Вывести (командой ip r) таблицу маршрутизации для **ws1**.

```
10.0.10.0/24 dev eth0  proto kernel  scope link  src 10.0.10.11
default via 10.0.10.1 dev eth0
```

Вывести (командой ip r) таблицу маршрутизации для **ws2**.

```
10.0.30.0/24 dev eth0  proto kernel  scope link  src 10.0.30.22
default via 10.0.30.3 dev eth0
```

Вывести (командой ip r) таблицу маршрутизации для **ws3**.

```
10.0.40.0/24 dev eth0  proto kernel  scope link  src 10.0.40.33
default via 10.0.40.1 dev eth0
```

4. Проверка настройки сети

Вывод **traceroute** от узла **ws1** до **ws3** при нормальной работе сети.

```
 1  10.0.10.1  9 ms  0 ms  0 ms
 2  10.0.40.33  11 ms  0 ms  0 ms
```

Вывод **traceroute** от узла **r1** до **ws2** при нормальной работе сети.

```
 1  10.0.20.2  2 ms  0 ms  0 ms
 2  10.0.30.22  26 ms  0 ms  0 ms
```

Вывод **traceroute** от узла **w1** до **r2** при нормальной работе сети.

```
 1  10.0.10.1  5 ms  0 ms  0 ms
 2  10.0.20.2  0 ms  0 ms  0 ms
```

5. Маршрутизация

Вначале стоит написать, какие MAC-адреса интерфейсов в опыте были у каких машин.

```
ws1 eth0:  a6:f9:52:b6:1e:69
r1 eth1: fa:de:dc:30:96:57
r1 eth2: 46:9d:1d:7c:12:b2
r2 eth1: 12:3e:e2:7d:e3:87
```

Затем вывести маршрутную таблицу маршрутизатора **r1** (вывод команды ip r!)

```
10.0.20.0/24 dev eth1  proto kernel  scope link  src 10.0.20.2
10.0.30.0/24 dev eth0  proto kernel  scope link  src 10.0.30.2
10.0.40.0/24 via 10.0.30.3 dev eth0
10.0.10.0/24 via 10.0.20.1 dev eth1
```

Показаны опыты после стирания кеша ARP. Далее показана отправка пакета на маршрутизатор **r2** (косвенная маршрутизация).

```
ws1:~# ping 10.0.30.2 -c 1
```

```
r1:~# tcpdump -tne -i eth1
a6:f9:52:b6:1e:69 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42:
        arp who-has 10.0.10.1 tell 10.0.10.11
fa:de:dc:30:96:57 > a6:f9:52:b6:1e:69, ethertype ARP (0x0806), length 42:
        arp reply 10.0.10.1 is-at fa:de:dc:30:96:57
a6:f9:52:b6:1e:69 > fa:de:dc:30:96:57, ethertype IPv4 (0x0800), length 98:
        10.0.10.11 > 10.0.20.2: ICMP echo request, id 12546, seq 1, length 64
fa:de:dc:30:96:57 > a6:f9:52:b6:1e:69, ethertype IPv4 (0x0800), length 98:
        10.0.20.2 > 10.0.10.11: ICMP echo reply, id 12546, seq 1, length 64
fa:de:dc:30:96:57 > a6:f9:52:b6:1e:69, ethertype ARP (0x0806), length 42:
        arp who-has 10.0.10.11 tell 10.0.10.1
a6:f9:52:b6:1e:69 > fa:de:dc:30:96:57, ethertype ARP (0x0806), length 42:
        arp reply 10.0.10.11 is-at a6:f9:52:b6:1e:69
```

Затем маршрутизатор отправил его далее.

```
r2:~# tcpdump -tne -i eth1
46:9d:1d:7c:12:b2 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42:
        arp who-has 10.0.20.2 tell 10.0.20.1
12:3e:e2:7d:e3:87 > 46:9d:1d:7c:12:b2, ethertype ARP (0x0806), length 42:
        arp reply 10.0.20.2 is-at 12:3e:e2:7d:e3:87
46:9d:1d:7c:12:b2 > 12:3e:e2:7d:e3:87, ethertype IPv4 (0x0800), length 98:
        10.0.10.11 > 10.0.20.2: ICMP echo request, id 12546, seq 1, length 64
12:3e:e2:7d:e3:87 > 46:9d:1d:7c:12:b2, ethertype IPv4 (0x0800), length 98:
        10.0.20.2 > 10.0.10.11: ICMP echo reply, id 12546, seq 1, length 64
12:3e:e2:7d:e3:87 > 46:9d:1d:7c:12:b2, ethertype ARP (0x0806), length 42:
        arp who-has 10.0.20.1 tell 10.0.20.2
46:9d:1d:7c:12:b2 > 12:3e:e2:7d:e3:87, ethertype ARP (0x0806), length 42:
        arp reply 10.0.20.1 is-at 46:9d:1d:7c:12:b2
```

6. Продолжительность жизни пакета

Сначала написать как и на чём ломали.

```
r2:~# ip link set eth0 down
r2:~# ip route add 10.0.30.0/24 via 10.0.20.1 dev eth1
```

Потом какая-то таблица вышла.

```
r2:~# ip r
10.0.20.0/24 dev eth1  proto kernel  scope link  src 10.0.20.2
10.0.30.0/24 via 10.0.20.1 dev eth1
10.0.10.0/24 via 10.0.20.1 dev eth1
```

Потом что слали.

```
ws1:~# ping 10.0.30.22 -c 1
```

И что в итоге получилось.

```
r2:~# tcpdump -tnve -i eth1
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
46:9d:1d:7c:12:b2 > 12:3e:e2:7d:e3:87, ethertype IPv4 (0x0800), length 98:
        (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
        10.0.10.11 > 10.0.30.22: ICMP echo request, id 14850, seq 1, length 64
12:3e:e2:7d:e3:87 > 46:9d:1d:7c:12:b2, ethertype IPv4 (0x0800), length 98:
        (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
        10.0.10.11 > 10.0.30.22: ICMP echo request, id 14850, seq 1, length 64
...
12:3e:e2:7d:e3:87 > 46:9d:1d:7c:12:b2, ethertype IPv4 (0x0800), length 98:
        (tos 0x0, ttl 2, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
        10.0.10.11 > 10.0.30.22: ICMP echo request, id 14850, seq 1, length 64
46:9d:1d:7c:12:b2 > 12:3e:e2:7d:e3:87, ethertype IPv4 (0x0800), length 98:
        (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
        10.0.10.11 > 10.0.30.22: ICMP echo request, id 14850, seq 1, length 64
12:3e:e2:7d:e3:87 > 46:9d:1d:7c:12:b2, ethertype IPv4 (0x0800), length 126:
        (tos 0xc0, ttl 64, id 49216, offset 0, flags [none], proto ICMP (1), length 112)
        10.0.20.2 > 10.0.10.11: ICMP time exceeded in-transit, length 92
        (tos 0x0, ttl 1, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
        10.0.10.11 > 10.0.30.22: ICMP echo request, id 14850, seq 1, length 64
```

И кто в итоге отравил сообщение о завершении жизни.

```
ws1:~# ping 10.0.30.22 -c 1
PING 10.0.30.22 (10.0.30.22) 56(84) bytes of data.
From 10.0.20.2 icmp_seq=1 Time to live exceeded

--- 10.0.30.22 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

7. Изучение IP-фрагментации

Написать, на каких узлах и как изменяли MTU.

```
r1:~# ip link set dev eth2 mtu 576
```

```
r2:~# ip link set dev eth1 mtu 576
```

Какие команды давали для тестирования и где.

```
ws1:~# ping -c 1 -s 1000 10.0.30.22
```

Вывод **tcpdump** на маршрутизаторе перед сетью с уменьшенным MTU.

```
r1:~# tcpdump -tnv -i eth1 icmp
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
IP (tos 0x0, ttl 64, id 18648, offset 0, flags [none], proto ICMP (1), length 1028) 10.0.10.11
IP (tos 0x0, ttl 62, id 18125, offset 0, flags [none], proto ICMP (1), length 1028) 10.0.30.22
```

Вывод **tcpdump** на маршрутизаторе после сети с уменьшенным MTU.

```
r2:~# tcpdump -tnv -i eth1 icmp
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
IP (tos 0x0, ttl 63, id 18648, offset 0, flags [+], proto ICMP (1), length 572) 10.0.10.11 > 10
IP (tos 0x0, ttl 63, id 18648, offset 552, flags [none], proto ICMP (1), length 476) 10.0.10.11
```

Вывод **tcpdump** на узле получателя.

```
ws2:~# tcpdump -tnv -i eth0 icmp
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
IP (tos 0x0, ttl 62, id 18648, offset 0, flags [none], proto ICMP (1), length 1028) 10.0.10.11
IP (tos 0x0, ttl 64, id 18125, offset 0, flags [none], proto ICMP (1), length 1028) 10.0.30.22
```

8. Отсутствие сети

Аналогично опишите опыт, когда маршрутизатор отсылает сообщение об отсутствии с сети. С командами и выводом, мак адреса не нужны.

```
ws1:~# ping -c 1 10.1.30.22
PING 10.1.30.22 (10.1.30.22) 56(84) bytes of data.
From 10.0.10.1 icmp_seq=1 Destination Net Unreachable
```

```
r1:~# tcpdump -n -i eth1 icmp
13:25:22.808880 IP 10.0.10.11 > 10.1.30.22: ICMP echo request, id 13058, seq 1, length 64
13:25:22.808897 IP 10.0.10.1 > 10.0.10.11: ICMP net 10.1.30.22 unreachable, length 92
```

9. Отсутствие IP-адреса в сети

Аналогично опишите опыт, когда маршрутизатор отсылает сообщение об отстутствии требуемого IP-адреса в сети. С командами и выводом, мак адреса не нужны.

```
ws1:~# ping -c 1 10.0.20.254
PING 10.0.20.254 (10.0.20.254) 56(84) bytes of data.
From 10.0.10.1 icmp_seq=1 Destination Host Unreachables
```

```
r1:~# tcpdump -n -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
13:32:13.144890 IP 10.0.10.11 > 10.0.20.254: ICMP echo request, id 14594, seq 1, length 64
13:32:16.149622 IP 10.0.10.1 > 10.0.10.11: ICMP host 10.0.20.254 unreachable, length 92
13:32:18.142925 arp who-has 10.0.10.1 tell 10.0.10.11
13:32:18.142939 arp reply 10.0.10.1 is-at fa:de:dc:30:96:57
```