Lab 5 : Endereçamento IP e Roteamento Estático

Rodrigo Seiji Piubeli Hirao (186837)

16 de dezembro de 2021

Conteúdo

1 Introdução

Nesse laboratório serão estudados o funcionamento de subredes e tabelas de roteamento, assim sendo visto o impacto de roteadores em uma rede, bem como uma configuração manual dos mesmos.

2 Metodologia

2.1 Atividade 1

Foi emulado o sistema da figura 01 e estudado seu comportamento

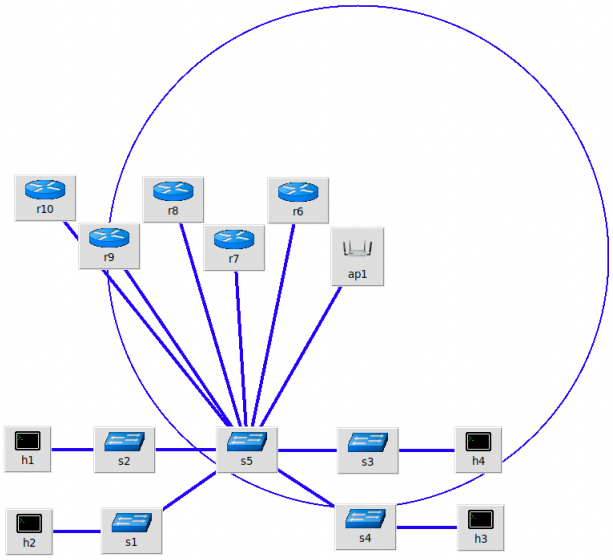

Figura 1: Topologia para configuração livre nos experimentos.

2.2 Atividade 2

Agora será analisado a rede a seguir com 1 domínio de broadcast que liga a subrede 10.0.1.0 com a 10.0.2.0 (cada uma com suporte de todos os ips 10.0.X.1 até 10.0.X.255)

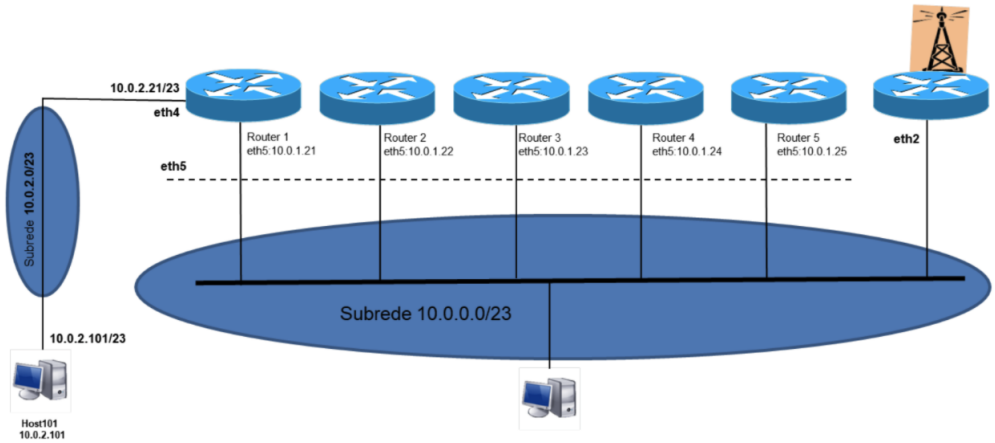

Figura 2: Topologia para atividade 2.

Foi então adicionado os códigos a segui, além de alterado todos os netmasks para 24.

```
46      h21 = net.addHost('h21', cls=Host, ip='10.0.2.101/24', defaultRoute='via 10.0.2.21')
```

```
70      net.addLink(r6, h21, intfName1='r6-eth1', intfName2='h21-eth0')
71      r6.setIP('10.0.2.21/23', intf='r6-eth1')
```

E testado as rotas, porém as rotas de 10.0.0.X até 10.0.2.X não iriam funcionar sem adicionar a roda padrão nos hosts em 10.0.0.0.

```
40      h11 = net.addHost('h1', cls=Host, ip='10.0.1.1/23', defaultRoute='via 10.0.1.21')
41      h12 = net.addHost('h2', cls=Host, ip='10.0.1.2/23', defaultRoute='via 10.0.1.21')
42      h13 = net.addHost('h3', cls=Host, ip='10.0.1.3/23', defaultRoute='via 10.0.1.21')
43      h14 = net.addHost('h4', cls=Host, ip='10.0.1.4/23', defaultRoute='via 10.0.1.21')
```

Por final foi testada a velocidade de conexão entre os hosts e comparado com a atividade anterior.

2.3 Atividade 3

A última atividade consistiu na implementação da topologia da figura 3.

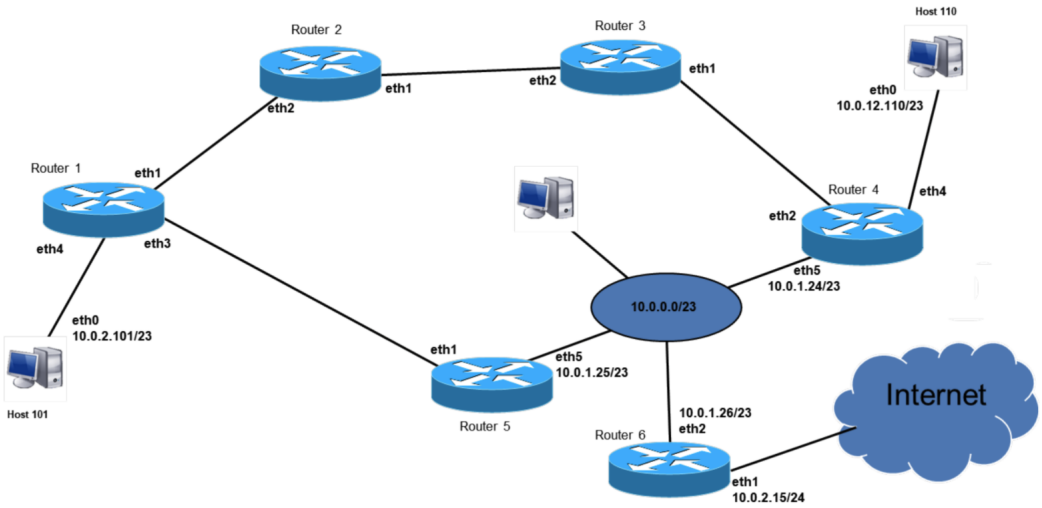

Figura 3: Topologia para atividade 3.

O que exigiu a configuração manual de cada tabela de roteamento, como pode ser visto no código a seguir.

```
98      r1.cmd("ip route add 10.0.12.0/24 via 10.0.1.122 dev r1-eth1")
99      r2.cmd("ip route add 10.0.12.0/24 via 10.0.1.132 dev r2-eth1")
100     r3.cmd("ip route add 10.0.12.0/24 via 10.0.1.142 dev r3-eth1")
101
102     r4.cmd("ip route add 10.0.2.0/24 via 10.0.1.131 dev r4-eth2")
103     r3.cmd("ip route add 10.0.2.0/24 via 10.0.1.121 dev r3-eth2")
104     r2.cmd("ip route add 10.0.2.0/24 via 10.0.1.111 dev r2-eth2")
105
106     r1.cmd("ip route add 10.0.0.0/24 via 10.0.1.151 dev r1-eth3")
107     r2.cmd("ip route add 10.0.0.0/24 via 10.0.1.132 dev r2-eth1")
108     r3.cmd("ip route add 10.0.0.0/24 via 10.0.1.142 dev r3-eth1")
109     r4.cmd("ip route add 10.0.0.0/24 via 10.0.1.104 dev r4-eth5")
110     r5.cmd("ip route add 10.0.0.0/24 via 10.0.1.105 dev r5-eth5")
111     r6.cmd("ip route add 10.0.0.0/24 via 10.0.1.106 dev r6-eth5")
112
113     r0.cmd("ip route add 10.0.2.0/24 via 10.0.1.155 dev r0-eth5")
114     r5.cmd("ip route add 10.0.2.0/24 via 10.0.1.111 dev r5-eth1")
115
116     r0.cmd("ip route add 10.0.12.0/24 via 10.0.1.145 dev r0-eth4")
```

E, por algum motivo que não foi descoberto durante o experimento, as primeiras interfaces criadas ficavam sem ipv4, por tal motivo foi necessário adicionar o código a seguir para arrumar o problema.

```
90      r1.cmd("ifconfig r1-eth1 10.0.1.111 netmask 255.255.255.0 up")
91      r2.cmd("ifconfig r2-eth2 10.0.1.122 netmask 255.255.255.0 up")
92      r3.cmd("ifconfig r3-eth2 10.0.1.132 netmask 255.255.255.0 up")
93      r4.cmd("ifconfig r4-eth2 10.0.1.142 netmask 255.255.255.0 up")
94      r5.cmd("ifconfig r5-eth1 10.0.1.151 netmask 255.255.255.0 up")
95      r6.cmd("ifconfig r6-eth5 10.0.1.165 netmask 255.255.255.0 up")
96      r0.cmd("ifconfig r0-eth1 10.0.0.100 netmask 255.255.255.0 up")
```

Finalmente, foram analisadas todas as métricas dessa topologia assim como na atividade anterior.

3 Resultados e Discussão

3.1 Atividade 1

Foi criada a seguinte rede

```
r6 r6-eth0:s5-eth5
r7 r7-eth0:s5-eth6
r8 r8-eth0:s5-eth7
r9 r9-eth0:s5-eth8
r10 r10-eth0:s5-eth9
h1 h1-eth0:s2-eth1
h2 h2-eth0:s1-eth1
h3 h3-eth0:s4-eth1
h4 h4-eth0:s3-eth1
s1 lo:  s1-eth1:h2-eth0 s1-eth2:s5-eth1
s2 lo:  s2-eth1:h1-eth0 s2-eth2:s5-eth2
s3 lo:  s3-eth1:h4-eth0 s3-eth2:s5-eth3
s4 lo:  s4-eth1:h3-eth0 s4-eth2:s5-eth4
s5 lo:  s5-eth1:s1-eth2 s5-eth2:s2-eth2 s5-eth3:s3-eth2 s5-eth4:s4-eth2 s5-eth5:r6-
eth0 s5-eth6:r7-eth0 s5-eth7:r8-eth0 s5-eth8:r9-eth0 s5-eth9:r10-eth0 s5-eth10:ap1-
eth2
ap1 lo:  ap1-wlan1:wifi ap1-eth2:s5-eth10
```

E foi notado um comportamento normal entre os hosts

```
mininet-wifi> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['25.7 Gbits/sec', '25.8 Gbits/sec']
mininet-wifi> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
.*** Results: ['16.6 Gbits/sec', '16.6 Gbits/sec']
mininet-wifi> iperf h1 h4
*** Iperf: testing TCP bandwidth between h1 and h4
.*** Results: ['17.8 Gbits/sec', '17.8 Gbits/sec']
mininet-wifi> iperf h2 h3
*** Iperf: testing TCP bandwidth between h2 and h3
.*** Results: ['21.1 Gbits/sec', '21.1 Gbits/sec']
mininet-wifi> iperf h2 h4
*** Iperf: testing TCP bandwidth between h2 and h4
.*** Results: ['23.3 Gbits/sec', '23.3 Gbits/sec']
mininet-wifi> iperf h3 h4
*** Iperf: testing TCP bandwidth between h3 and h4
.*** Results: ['25.8 Gbits/sec', '25.9 Gbits/sec']
```

Porém percebe-se que não há os roteadores nas tabelas arp, que deveriam estar pois o roteador implementa a camada de rede.

```
mininet-wifi> h1 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.4                 ether   36:c2:18:7c:b1:c3   C                     h1-eth0
10.0.0.3                 ether   72:8c:2b:5f:90:f1   C                     h1-eth0
10.0.0.2                 ether   22:89:5d:f4:c5:6d   C                     h1-eth0
mininet-wifi> h2 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.1                 ether   f2:b7:72:4c:0a:03   C                     h2-eth0
10.0.0.3                 ether   72:8c:2b:5f:90:f1   C                     h2-eth0
```

```
10.0.0.4                 ether   36:c2:18:7c:b1:c3   C                     h2-eth0
mininet-wifi> h3 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.1                 ether   f2:b7:72:4c:0a:03   C                     h3-eth0
10.0.0.2                 ether   22:89:5d:f4:c5:6d   C                     h3-eth0
10.0.0.4                 ether   36:c2:18:7c:b1:c3   C                     h3-eth0
mininet-wifi> h1 ip route
10.0.0.0/8 dev h1-eth0 proto kernel scope link src 10.0.0.1
mininet-wifi> h2 ip route
10.0.0.0/8 dev h2-eth0 proto kernel scope link src 10.0.0.2
mininet-wifi> h3 ip route
10.0.0.0/8 dev h3-eth0 proto kernel scope link src 10.0.0.3
```

Assim foi analisado o script e percebido que todos os roteadores estavam com o ip 0.0.1.0, assim foi alterado cada roteador para um ip único e obtido o resultado

```
mininet-wifi> h1 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.1.21                ether   8e:bf:ee:a6:3b:a8   C                     h1-eth0
10.0.1.3                 ether   86:a1:a7:9b:39:1b   C                     h1-eth0
10.0.1.23                ether   8e:8e:46:6a:45:1d   C                     h1-eth0
10.0.1.25                ether   1a:3f:a7:a2:40:31   C                     h1-eth0
10.0.1.2                 ether   d2:c6:b6:63:1a:65   C                     h1-eth0
10.0.1.22                ether   ce:d0:48:51:6e:ab   C                     h1-eth0
10.0.1.4                 ether   26:04:05:22:80:42   C                     h1-eth0
10.0.1.24                ether   12:2c:96:15:22:d1   C                     h1-eth0
```

Vale notar que o roteador é implementado como um host executando **sysctl**.

3.2 Atividade 2

Foi criado a seguinte tabela no roteador, onde ele possui **10.0.1.21** na subrede **10.0.0.0/23** e **10.0.2.21** na subrede **10.0.2.0/23**. Pode ser visto também a que porta está associado cada rota, a mácara de rede, métricas, etc. (Obs.: não era necessário usar o **-n** no route, pois este serve para usar valores numéricos invés de símbolos, mas mesmo sem tal argumento ainda é mostrado apenas números pois não há hostnames na rede local)

```
mininet-wifi> r6 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 r6-eth0
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r6-eth1
mininet-wifi> r6 ip route show
10.0.0.0/23 dev r6-eth0 proto kernel scope link src 10.0.1.21
10.0.2.0/23 dev r6-eth1 proto kernel scope link src 10.0.2.21
```

Depois de arrumado o código foram obtidos os seguintes resultados de ping e tracepath, que mostram uma conexão com sucesso, e um caminho de 3 nós até o objetivo

```
mininet-wifi> h21 ping h1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=63 time=0.237 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=63 time=0.062 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=63 time=0.080 ms
^C
--- 10.0.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2025ms
rtt min/avg/max/mdev = 0.062/0.126/0.237/0.078 ms
mininet-wifi> h21 tracepath h1
 1?: [LOCALHOST]                      pmtu 1500
 1:  ???                                                0.041ms
```

```
 1:  ???                                                       0.016ms
 2:  ???                                                       0.032ms reached
     Resume: pmtu 1500 hops 2 back 2
```

Executando o iperf foi notado uma menor velocidade em relação à atividade anterior, o que se deve ao fato de haver um roteador intermediário que precisa consiltar a tabela de rotas para continuar o caminho.

```
mininet-wifi> iperf h21 h1
*** Iperf: testing TCP bandwidth between h21 and h1
.*** Results: ['23.3 Gbits/sec', '23.3 Gbits/sec']
```

3.3 Atividade 3

Foram obtidas as seguintes tabelas de roteamento, onde pode ser visto os gateway para cada próximo roteador até o objetivo

```
mininet-wifi> r1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.1.151      255.255.255.0   UG    0      0        0 r1-eth3
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth3
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth1
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 r1-eth4
10.0.12.0       10.0.1.122      255.255.255.0   UG    0      0        0 r1-eth1
mininet-wifi> r2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.1.132      255.255.255.0   UG    0      0        0 r2-eth1
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
10.0.2.0        10.0.1.111      255.255.255.0   UG    0      0        0 r2-eth2
10.0.12.0       10.0.1.132      255.255.255.0   UG    0      0        0 r2-eth1
mininet-wifi> r3 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.1.142      255.255.255.0   UG    0      0        0 r3-eth1
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r3-eth2
10.0.2.0        10.0.1.121      255.255.255.0   UG    0      0        0 r3-eth2
10.0.12.0       10.0.1.142      255.255.255.0   UG    0      0        0 r3-eth1
mininet-wifi> r4 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.1.104      255.255.255.0   UG    0      0        0 r4-eth5
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth5
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r4-eth2
10.0.2.0        10.0.1.131      255.255.255.0   UG    0      0        0 r4-eth2
10.0.12.0       0.0.0.0         255.255.255.0   U     0      0        0 r4-eth4
mininet-wifi> r5 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.1.105      255.255.255.0   UG    0      0        0 r5-eth5
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r5-eth5
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r5-eth1
10.0.2.0        10.0.1.111      255.255.255.0   UG    0      0        0 r5-eth1
mininet-wifi> r6 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.1.106      255.255.255.0   UG    0      0        0 r6-eth5
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 r6-eth5
```

Pode ser visto pela análise da performance que a distância influencia bastante na banda, evidente na banda menor de h101 a h110 (4 hops), e na maior banda de h110 a h1 (2 hops).

```
mininet-wifi> iperf h101 h110
*** Iperf: testing TCP bandwidth between h101 and h110
*** Results: ['9.73 Gbits/sec', '9.75 Gbits/sec']
mininet-wifi> iperf h101 h1
*** Iperf: testing TCP bandwidth between h101 and h1
.*** Results: ['14.2 Gbits/sec', '14.2 Gbits/sec']
mininet-wifi> iperf h110 h1
*** Iperf: testing TCP bandwidth between h110 and h1
.*** Results: ['15.4 Gbits/sec', '15.4 Gbits/sec']
```

E pelo tracepath dá para notar a quantidade de pulos entre cada host

```
mininet-wifi> h101 tracepath h110
 1?: [LOCALHOST]                      pmtu 1500
 1:  ???                                                0.040ms
 1:  ???                                                0.014ms
 2:  ???                                                0.036ms
 3:  ???                                                0.214ms
 4:  ???                                                0.054ms
 5:  ???                                                0.032ms reached
     Resume: pmtu 1500 hops 5 back 5
mininet-wifi> h101 tracepath h1
 1?: [LOCALHOST]                      pmtu 1500
 1:  ???                                                0.134ms
 1:  ???                                                0.016ms
 2:  ???                                                0.022ms
 3:  ???                                                0.023ms
 4:  ???                                                0.017ms reached
     Resume: pmtu 1500 hops 4 back 4
mininet-wifi> h110 tracepath h1
 1?: [LOCALHOST]                      pmtu 1500
 1:  ???                                                0.039ms
 1:  ???                                                0.018ms
 2:  ???                                                0.033ms
 3:  ???                                                0.020ms reached
     Resume: pmtu 1500 hops 3 back 3
```

4 Conclusão

Pôde ser visto o papel essencial de um computador funcionando como roteador para conectar subredes e encaminhar pacotes. Com a configuração da tabela de roteamento em cada roteador foi possível criar um grafo conectando subredes muito distantes, apenas utilizando alguns hops, porém pôde ser visto também a grande perda de banda para cada hop que foi adicionado no caminho, o que exige um algoritmo eficiente para busca de menores caminhos em uma rede complexa (um desses algoritmos sendo o OSPF).

Além disso foi utilizado a interface de edição gráfica do mininet, que se provou mais prática, porém não tão poderosa, quanto a programação direta em script.