EA080: Laboratório 05

Vinícius Esperança Mantovani,
247395.

Introdução)

Neste experimento, visa-se a conhecer melhor a respeito de protocolos de roteamento com enfoque para o protocolo *OSPF*. Isso se dá como uma forma de continuar o processo feito no laboratório anterior, em que as rotas dos roteadores foram todas configuradas manualmente. Assim, objetiva-se não somente entender e conhecer o protocolo na prática, mas também reconhecer sua importância como automatizador no âmbito de redes.

Exercício 1)

Para iniciar o experimento, foi criada a topologia da *Figura 1*, por meio do *script* "*topo_A3.py*" presente na pasta "lab5" da disciplina.

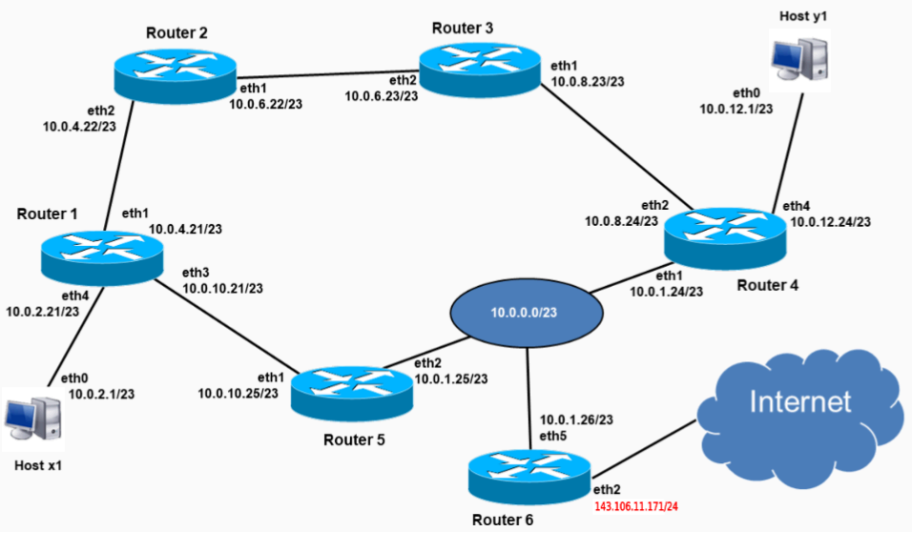


Figura 1: Topologia OSPF usada no experimento

Feito isso, foi executado o comando *net* (*Figura 2*) para se analisar a corretude da topologia gerada. Desse modo, pôde-se observar, conforme esperado, que a rede criada apresenta a mesma estrutura que a da *Figura 1*.

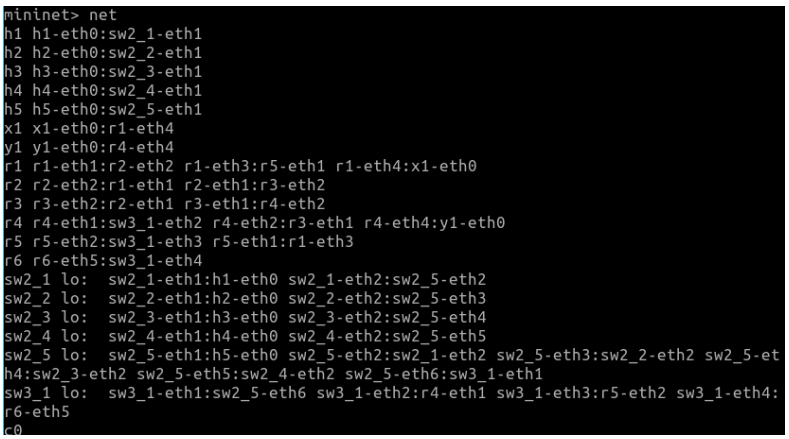


Figura 2: Resposta ao comando net

Seguindo, foi aberto um novo terminal e, nele, foi executado o comando "*sudo ps aux | grep quagga*" como forma de se observar os processos *ospfd* e *zebra* do *software quagga*, responsáveis por coordenar os roteadores da rede gerada. Com isso, obteve-se a resposta contendo tais processos, conforme se segue:

Figura 3: Resposta ao comando "sudo ps aux | grep quagg", processos ospf e zebra

Na figura acima, pode-se notar a existência de 12 processos (*daemons*), dois para cada roteador instanciado na rede criada com *mininet*. Cada par contém, conforme citado, um *daemon zebra* e um *ospfd*. Este último processo citado é responsável por implementar as funcionalidades do roteador referentes ao protocolo *OSPF*, enquanto que o processo *zebra* se incumbe de fazer uma ponte entre o primeiro e o sistema operacional, de modo a permitir a portabilidade do *software* para os mais variados sistemas *UNIX*.

Em sequência ao procedimento experimental adotado, fez-se um teste de conectividade entre os *hosts x1* e *y1*, apresentado na figura abaixo, utilizando-se os comandos de *ping* e *tracepath*. Dessa forma, pôde-se perceber que não há conectividade entre tais *hosts*, pois há problema de repasse já no *Gateway* de *x1*, conforme se nota na *Figura 5*.

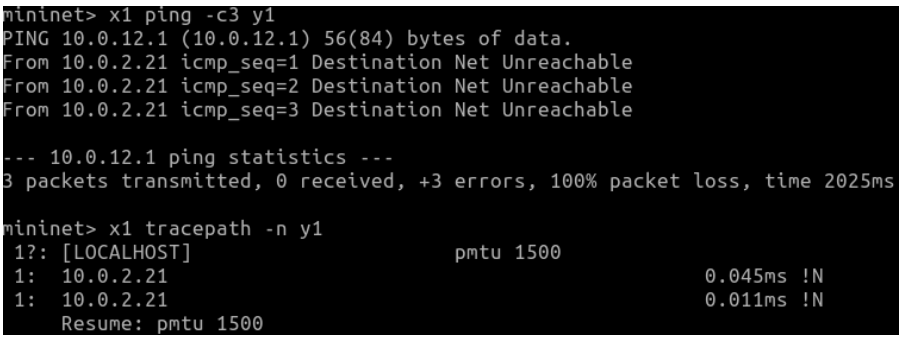


Figura 4: Resposta aos comandos ping e tracepath

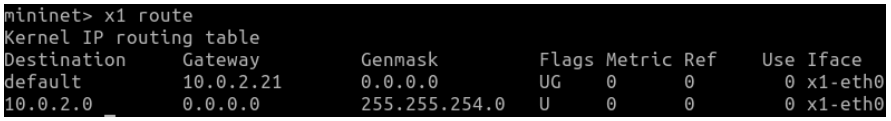


Figura 5: Resposta ao comando route para o host x1

Exercício 2)

Analisando a *Figura 1*, vê-se oito redes ao todo, sendo seus endereços:

1) 10.0.10.0/23
2) 10.0.0.0/23
3) 143.106.11.0/24
4) 10.0.8.0/23
5) 10.0.6.0/23
6) 10.0.4.0/23
7) 10.0.2.0/23
8) 10.0.12.0/23

Seguindo para uma análise das rotas do roteador, executaram-se os comandos "*ifconfig -a*" e "*route -n*" no terminal do roteador 1, do modo como se apresenta na *Figura 6*. Com isso, observou-se uma discrepância entre uma das rotas e as demais, sendo esta, a rota para a rede 10.0.6.0/23, que se dá de modo que o roteador não tem contato direto com ela, logo, precisa repassar os pacotes com destino a ela para o roteador *r2*, que tem contato direto com a rede em questão.

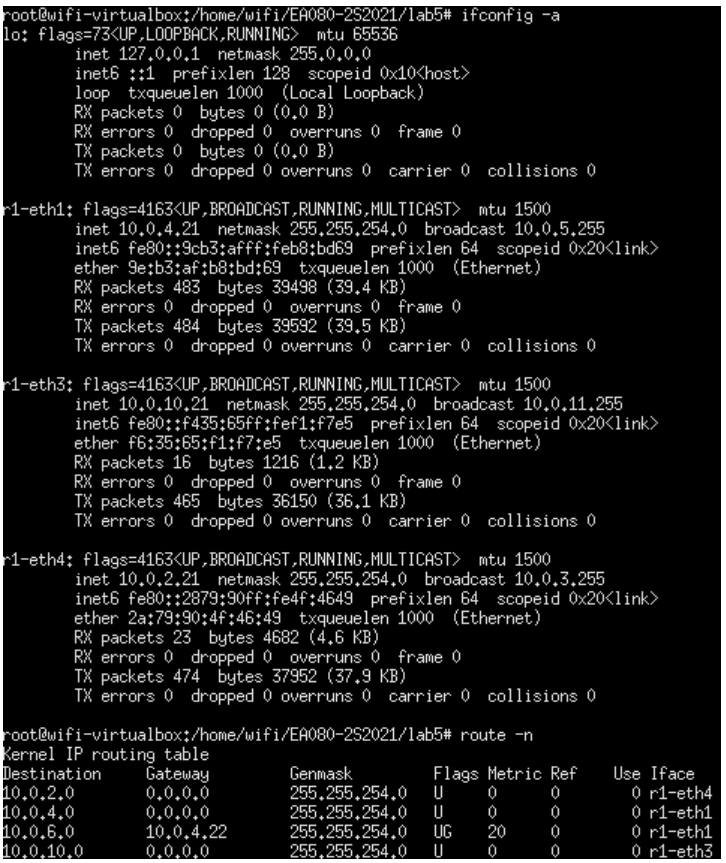


Figura 6: Resposta ao comando route no roteador 1

Na sequência do que se explicou, foi executado um *telnet*, ainda no terminal do roteador 1, conforme "*telnet localhost ospfd*" para verificar que o processo *quagga ospfd* está rodando neste roteador e, podê-lo configurar. Nesse sentido, foram usados os comandos *enable* na comunicação *telnet* estabelecida, de maneira que o roteador passa a operar com o protocolo *OSPF* para roteamento de pacotes. Além disso, foram executados os comandos "*sh ip ospf route*" e "*sh ip ospf neighbor*" para se observar a tabela de roteamento *OSPF* do roteador e os roteadores vizinhos a ele, respectivamente, conforme a figura abaixo.

Figura 7: Resposta a comandos em telnet com processo ospf roteador 1

Sob análise ainda superficial da figura acima, pode-se notar que, apesar da existência de dois roteadores vizinhos ao roteador 1 na topologia estudada, há apenas um na tabela de vizinhos da figura. Isso se dá, por conta de o roteador 5 não estar configurado corretamente para uso de OSPF para roteamento, o que será corrigido no futuro. Sendo assim, as conexões entre o roteador 1 e o roteador 2

se dá, por meio das interfaces *r1-eth1* de endereço 10.0.4.21/23 e *r2-eth2* de endereço 10.0.4.22/23. Já quanto aos roteadores 1 e 5, tem-se a conexão por meio das interfaces *r1-eth3* de endereço 10.0.10.21/23 e *r5-eth1* de endereço 10.0.10.25/23, no entanto, este roteador (*r5*) não está repassando pacotes, pois não está configurado.

Apresentando-se também a tabela de roteamento do roteador 1 com o comando *route*, tem-se a *Figura 6*. Dessa maneira, pode-se notar que a semelhança entre esta forma de se requisitar a tabela e a forma anterior, usando *telnet* para se comunicar com *ospfd*, a exibição dos endereços das redes de destino, do *Gateway* e a interface pela qual ocorre o repasse. No entanto, é possível notar algumas diferenças também, dentre as quais estão a inexistência de apresentação de *Flags, Metrics, Ref e Use* nas informações tiradas de *ospfd*.

Finalmente, vale ressaltar que a rota que difere das outras na tabela foi anunciada pelo roteador que tem como interface na rede 10.0.4.0/23 aquela com endereço 10.0.4.22/23, ou seja, o roteador 2.

Exercício 3)

Em sequência ao que foi feito, utilizou-se agora o terminal do roteador 2 para repetir o processo de análise de tabela de roteamento e interfaces. Deste procedimento, obtiveram-se a figura abaixo:

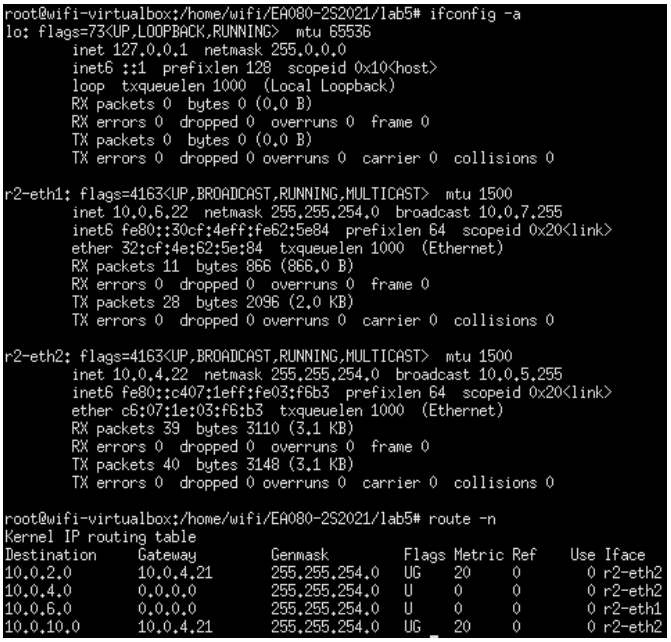


Figura 8: Resposta aos comandos ifconfig e route em r2

Nesse roteador (*r2*), é possível notar que, ao contrário do caso anterior, há duas rotas que passam por *Gateways*, não somente uma. Sendo assim, a única rota que se difere das demais é a rota para 10.0.6.0/23. Isso porque, essa rota se dá pela interface *r2-eth1*, enquanto todas as outras se dão por *r2-eth2*. Tal fato se dá pela conexão direta desse roteador a tal rede pela interface citada. No entanto, se for desejado destacar as redes em que há *Gateway* para repassar pacotes cujo destino seja ela, pode-se apresentar as redes 10.0.2.0/23 e 10.0.10.0/23, para as quais o roteador 2 tem de repassar os pacotes para o roteador 1 (10.0.4.21). Vale ressaltar ainda, que as portas que diferem das demais pelo *Gateway* foram anunciadas por *r1*.

Ainda no roteador *r2*, comunicando-se por telnet com o processo *ospfd* deste roteador, foram executados, como para o roteador 1, comandos para ver sua tabela de roteamento e seus roteadores vizinhos. Isso se apresenta na figura a seguir.

```
ospfd-r2# sh ip ospf route
============ OSPF network routing table ============
N    10.0.2.0/23           [20] area: 0.0.0.0
                           via 10.0.4.21, r2-eth2
N    10.0.4.0/23           [10] area: 0.0.0.0
                           directly attached to r2-eth2
N    10.0.6.0/23           [10] area: 0.0.0.0
                           directly attached to r2-eth1
N    10.0.10.0/23          [20] area: 0.0.0.0
                           via 10.0.4.21, r2-eth2

============ OSPF router routing table =============

============ OSPF external routing table ===========

ospfd-r2# sh ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
10.0.4.21         1 Full/Backup       35.713s 10.0.4.21       r2-eth2:10.0.4.22        0     0     0
```

Figura 8: Resposta a comandos em telnet com processo ospf roteador 2

Conforme se nota na *Figura 8*, o roteador 2 reconhece um único roteador vizinho, o roteador 1. Sendo o endereço da interface de *r1* 10.4.0.21, mesmo endereço da interface pela qual estão ligados os roteadores 1 e 2.

Quanto às tabelas de roteamento dadas por *route* e pelo processo *ospfd*, tem-se de semelhança a presença dos mesmos caminhos (por obviedade) e os dados exibidos. No entanto, há diferenças, conforme explicado para o roteador 1, em algumas informações que são apresentadas em um caso e não no outro (como explicado para *r1*).

Prosseguindo com o experimento, ao se executar o comando "*timeout 10 tcpdump -i r2-eth2 -vvln*", obteve-se o que se apresenta logo a seguir:

```
root@wifi-virtualbox:/home/wifi/EA080-2S2021/lab5# timeout 10 tcpdump -i r2-eth2 -vvln
tcpdump: listening on r2-eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
09:38:34.687359 IP (tos 0xc0, ttl 1, id 26446, offset 0, flags [none], proto OSPF (89), length 68)
    10.0.4.22 > 224.0.0.5: OSPFv2, Hello, length 48
        Router-ID 10.0.6.22, Backbone Area, Authentication Type: none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.254.0, Priority 1
          Designated Router 10.0.4.22, Backup Designated Router 10.0.4.21
          Neighbor List:
            10.0.4.21
09:38:34.687421 IP (tos 0xc0, ttl 1, id 26738, offset 0, flags [none], proto OSPF (89), length 68)
    10.0.4.21 > 224.0.0.5: OSPFv2, Hello, length 48
        Router-ID 10.0.4.21, Backbone Area, Authentication Type: none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.254.0, Priority 1
          Designated Router 10.0.4.22, Backup Designated Router 10.0.4.21
          Neighbor List:
            10.0.6.22

2 packets captured
2 packets received by filter
0 packets dropped by kernel
```

Figura 9: Resposta ao comando "timeout 10 tcpdump -i r2-eth2 -vvln" em r2

Observando-se a figura acima, pode-se perceber que o pacote que contém “10.0.4.21 > 224.0.0.5” é um pacote *OSPF* DO TIPO “Hello” enviado em multicast pelo roteador 1 (10.0.4.21) e tem como destino o roteador 2, cujo endereço está na lista de vizinhos do próprio pacote. Isso se dá, pois o roteador 2 já foi previamente detectado como vizinho do roteador 1 e, por consequência, é agora exibido como tal na lista de vizinhos dos pacotes “Hello” do roteador 1. Cabe destacar, ainda, que esse pacote tem por objetivo informar aos roteadores da rede (por multicast) as vizinhanças do roteador que o envia e, como no caso presente, por vezes, a vizinhança dele contém o próprio roteador receptor, evidentemente, por causa de este ser vizinho daquele.

Exercício 4)

Neste exercício, verificou-se a indefinição de configurações *OSPF* para os roteadores 3, 4, 5 e 6. Isso se deu, usando os mesmos comandos já utilizados para comunicação com o *daemon ospfd* no roteador *r3*, conforme *Figura 10* a seguir.

Figura 10: Resposta ao comando “sh ip ospf route” em r3

Seguindo, após adicionar as rotas *ospf* ao roteador 3 de suas duas interfaces, tem-se:

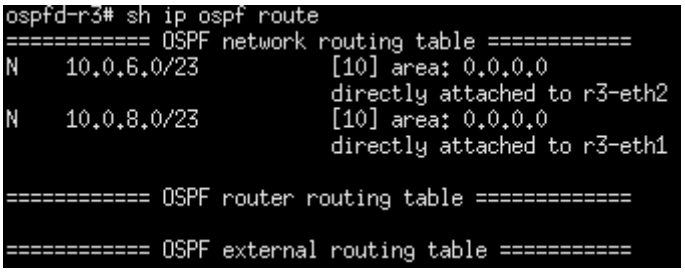


Figura 11: Resposta ao comando “sh ip ospf route” em r3 após configuração

Pouco tempo depois, o que se tem é:

Figura 12: Resposta aos comandos “sh ip ospf route” e “sh ip ospf neighbor” em r3 após configuração

Essa mudança ocorre devido aos sinais de “Hello” trocados e a entrada do roteador recém configurado no multicast de roteadores. Tendo essas novas tabelas, pode-se notar que o custo da rota até 10.0.2.0/23 é o mesmo da rota até 10.0.10.0/23. Isso se dá, em ambos os casos, pois o custo de

cada enlace tem valor de 10 e, a rota pela qual um pacote vai de *r3* para essas redes (passando por *r2*), tem três enlaces envolvidos, totalizando 30 de custo.

Exercício 5)

Continuando o procedimento experimental, foram configurados os três roteadores *ospf* faltantes (*r4*, *r5* e *r6*). Com isso, obtiveram-se as seguintes tabelas (para todos os roteadores, após a configuração).

```
mininet> r1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.10.25      255.255.254.0   UG    20     0        0 r1-eth3
10.0.2.0        0.0.0.0         255.255.254.0   U     0      0        0 r1-eth4
10.0.4.0        0.0.0.0         255.255.254.0   U     0      0        0 r1-eth1
10.0.6.0        10.0.4.22       255.255.254.0   UG    20     0        0 r1-eth1
10.0.8.0        10.0.4.22       255.255.254.0   UG    20     0        0 r1-eth1
10.0.10.0       0.0.0.0         255.255.254.0   U     0      0        0 r1-eth3
10.0.12.0       10.0.10.25      255.255.254.0   UG    20     0        0 r1-eth3
```

Figura 13: Resposta ao comando "route" em r1 após configuração de todos os roteadores

```
mininet> r2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.6.23       255.255.254.0   UG    20     0        0 r2-eth1
10.0.2.0        10.0.4.21       255.255.254.0   UG    20     0        0 r2-eth2
10.0.4.0        0.0.0.0         255.255.254.0   U     0      0        0 r2-eth2
10.0.6.0        0.0.0.0         255.255.254.0   U     0      0        0 r2-eth1
10.0.8.0        10.0.6.23       255.255.254.0   UG    20     0        0 r2-eth1
10.0.10.0       10.0.4.21       255.255.254.0   UG    20     0        0 r2-eth2
10.0.12.0       10.0.6.23       255.255.254.0   UG    20     0        0 r2-eth1
```

Figura 14: Resposta ao comando "route" em r2 após configuração de todos os roteadores

```
mininet> r3 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        10.0.8.24       255.255.254.0   UG    20     0        0 r3-eth1
10.0.2.0        10.0.6.22       255.255.254.0   UG    20     0        0 r3-eth2
10.0.4.0        10.0.6.22       255.255.254.0   UG    20     0        0 r3-eth2
10.0.6.0        0.0.0.0         255.255.254.0   U     0      0        0 r3-eth2
10.0.8.0        0.0.0.0         255.255.254.0   U     0      0        0 r3-eth1
10.0.10.0       10.0.8.24       255.255.254.0   UG    20     0        0 r3-eth1
10.0.12.0       10.0.8.24       255.255.254.0   UG    20     0        0 r3-eth1
```

Figura 15: Resposta ao comando "route" em r3 após configuração de todos os roteadores

```
mininet> r4 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.254.0   U     0      0        0 r4-eth1
10.0.2.0        10.0.1.25       255.255.254.0   UG    20     0        0 r4-eth1
10.0.4.0        10.0.8.23       255.255.254.0   UG    20     0        0 r4-eth2
10.0.6.0        10.0.8.23       255.255.254.0   UG    20     0        0 r4-eth2
10.0.8.0        0.0.0.0         255.255.254.0   U     0      0        0 r4-eth2
10.0.10.0       10.0.1.25       255.255.254.0   UG    20     0        0 r4-eth1
10.0.12.0       0.0.0.0         255.255.254.0   U     0      0        0 r4-eth4
```

Figura 16: Resposta ao comando "route" em r4 após configuração de todos os roteadores

```
mininet> r5 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.254.0   U     0      0        0 r5-eth2
10.0.2.0        10.0.10.21      255.255.254.0   UG    20     0        0 r5-eth1
10.0.4.0        10.0.10.21      255.255.254.0   UG    20     0        0 r5-eth1
10.0.6.0        10.0.1.24       255.255.254.0   UG    20     0        0 r5-eth2
10.0.8.0        10.0.1.24       255.255.254.0   UG    20     0        0 r5-eth2
10.0.10.0       0.0.0.0         255.255.254.0   U     0      0        0 r5-eth1
10.0.12.0       10.0.1.24       255.255.254.0   UG    20     0        0 r5-eth2
```

Figura 17: Resposta ao comando "route" em r5 após configuração de todos os roteadores

```
mininet> r6 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.254.0   U     0      0        0 r6-eth5
10.0.2.0        10.0.1.25       255.255.254.0   UG    20     0        0 r6-eth5
10.0.4.0        10.0.1.25       255.255.254.0   UG    20     0        0 r6-eth5
10.0.6.0        10.0.1.24       255.255.254.0   UG    20     0        0 r6-eth5
10.0.8.0        10.0.1.24       255.255.254.0   UG    20     0        0 r6-eth5
10.0.10.0       10.0.1.25       255.255.254.0   UG    20     0        0 r6-eth5
10.0.12.0       10.0.1.24       255.255.254.0   UG    20     0        0 r6-eth5
```

Figura 18: Resposta ao comando "route" em r6 após configuração de todos os roteadores

Superficialmente, pelo número de *Gateways* e redes presentes nas tabelas, já se nota que os roteadores foram realmente configurados. Ainda assim, para testar o real funcionamento da configuração, testa-se inicialmente uma conexão *ping* entre *x1* e *y1*, recebendo-se como resposta:

```
mininet> x1 ping -c3 y1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.
64 bytes from 10.0.12.1: icmp_seq=1 ttl=61 time=1.46 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=61 time=0.202 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=61 time=0.084 ms

--- 10.0.12.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.084/0.581/1.459/0.622 ms
```

Figura 19: Ping de x1 para y1 bem sucedido

Da *Figura 19* acima, percebe-se que há conectividade entre os dois *hosts* em questão, sendo que o número de saltos entre um e o outro é de 4, conforme se vê abaixo.

```
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.21                                             0.080ms
 1:  10.0.2.21                                             0.016ms
 2:  10.0.10.25                                            0.022ms
 3:  10.0.1.24                                             1.741ms
 4:  10.0.12.1                                             0.923ms reached
     Resume: pmtu 1500 hops 4 back 4
```

Figura 20: tracepath de x1 a y1

Analisando a figura acima, pode-se afirmar, ainda, que o caminho de um pacote que vai de *x1* a *y1* é: 10.0.2.1 (*x1*) → 10.0.2.21 (*r1*) → 10.0.10.25 (*r5*) → 10.0.1.24 (*r4*) → 10.0.12.1 (*y1*). Estudando, então, o caso em que se coloca um custo de 100 na interface *r5-eth1*, vê-se uma diferença natural nos tempos de ping de *x1* para *y1* comparado ao caminho contrário (*Figuras 21* e *22*). No entanto, essa diferença é muito pequena e seria contida caso fosse relevante pela alteração da tabela de roteamento dos roteadores no caso em que fosse muito grande (como ocorre no caso a seguir). Para observar isso, pode-se ver as figuras a seguir.

```
mininet> x1 ping -c10 y1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.
64 bytes from 10.0.12.1: icmp_seq=1 ttl=60 time=0.081 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=60 time=0.132 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=60 time=0.129 ms
64 bytes from 10.0.12.1: icmp_seq=4 ttl=60 time=0.139 ms
64 bytes from 10.0.12.1: icmp_seq=5 ttl=60 time=0.140 ms
64 bytes from 10.0.12.1: icmp_seq=6 ttl=60 time=0.144 ms
64 bytes from 10.0.12.1: icmp_seq=7 ttl=60 time=0.131 ms
64 bytes from 10.0.12.1: icmp_seq=8 ttl=60 time=0.171 ms
64 bytes from 10.0.12.1: icmp_seq=9 ttl=60 time=0.147 ms
64 bytes from 10.0.12.1: icmp_seq=10 ttl=60 time=0.127 ms

--- 10.0.12.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9252ms
rtt min/avg/max/mdev = 0.081/0.134/0.171/0.021 ms
```

Figura 21: ping de x1 a y1 com r5-eth1 de custo 100

```
mininet> y1 ping -c10 x1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=61 time=1.60 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=61 time=0.492 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=61 time=0.140 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=61 time=0.145 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=61 time=0.133 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=61 time=0.138 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=61 time=0.132 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=61 time=0.133 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=61 time=0.177 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=61 time=0.210 ms

--- 10.0.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9152ms
rtt min/avg/max/mdev = 0.132/0.329/1.595/0.434 ms
```

Figura 22: ping de y1 a x1 com r5-eth1 de custo 100

```
mininet> y1 tracepath -n x1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.12.24                                            0.429ms
 1:  10.0.12.24                                            0.014ms
 2:  10.0.8.23                                             0.124ms
 3:  10.0.6.22                                             0.102ms
 4:  10.0.10.21                                            1.948ms asymm  3
 5:  10.0.2.1                                              1.015ms reached
     Resume: pmtu 1500 hops 5 back 4
```

Figura 23: tracepath de y1 a x1

Prosseguindo, foi alterado o custo da outra interface de *r5* para 100 e, obteve-se o que se segue ao fazer um ping de *y1* para *x1*:

```
mininet> y1 ping x1 -c10
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=60 time=0.075 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=60 time=0.106 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=60 time=0.108 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=60 time=0.149 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=60 time=0.140 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=60 time=0.438 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=60 time=0.107 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=60 time=0.134 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=60 time=0.139 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.136 ms

--- 10.0.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9202ms
```

Figura 24: Ping de y1 a x1 com r5-eth1 e r5-eth2 de custo 100

Nesse caso apresentado, com a alteração do custo da outra interface de *r5*, não há diferença alguma para o ping de *y1* para *x1*, uma vez que os pacotes que têm tal direção não passam por nenhum dos enlaces conectados a *r5* diretamente. Logo, não sofrem com o aumento do custo de tais enlaces. Contudo, analisando a resposta ao comando *tracepath x1* para *y1*, nota-se uma diferença na rota de pacotes de *x1* com destino a *y1*, sendo ela, agora, a rota contrária de pacotes que saem de *y1* e vão para *x1*. Assim, os dois *hosts* estão a 5 saltos de distância em ambos os sentidos.

```
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.21                                             0.081ms
 1:  10.0.2.21                                             0.010ms
 2:  10.0.4.22                                             0.022ms
 3:  10.0.6.23                                             0.020ms
 4:  10.0.8.24                                             0.023ms
 5:  10.0.12.1                                             0.024ms reached
     Resume: pmtu 1500 hops 5 back 5
```

Figura 25: Tracepath de x1 a y1 com r5-eth1 e r5-eth2 de custo 100

Em sequência, foi derrubado o enlace entre *r2* e *r3* com o comando “link r2 r3 down”. Nesse contexto, para analisar as consequências disso na rota estudada, foi feito um *tracepath* de *x1* para *y1* novamente, conforme se segue.

```
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.21                                             0.060ms
 1:  10.0.2.21                                             0.011ms
 2:  10.0.10.25                                            0.023ms
 3:  10.0.1.24                                             3.524ms
 4:  10.0.12.1                                             1.828ms reached
     Resume: pmtu 1500 hops 4 back 4
```

Figura 26: Tracepath de x1 a y1 com r5-eth1 e r5-eth2 de custo 100 e enlace entre r2 e r3 derrubado

Feito isso, é possível ver que a nova rota pela qual um pacote vai de *x1* para *y1* é a mesma que a primeira, ou seja, passa por *r5*, mesmo com o alto custo dos enlaces de *r5*. Isso porque, com o enlace entre *r2* e *r3* derrubado, não há outra opção de rota senão essa.

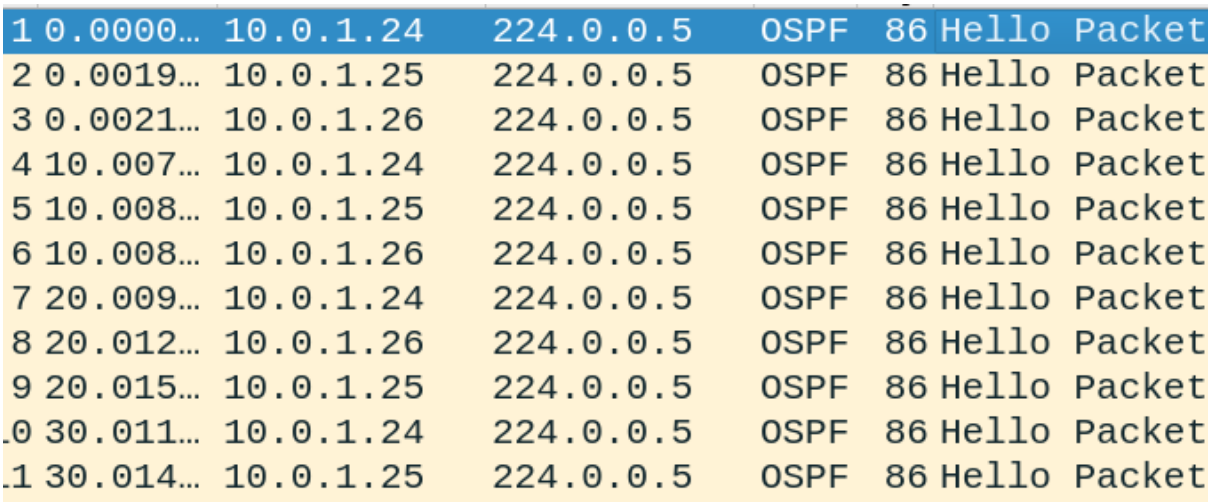
Finalmente, exibe-se a tabela de roteamento final de *r1* e se fazem considerações a respeito dela:

```
ospfd-r1# sh ip ospf route
============ OSPF network routing table ============
N    10.0.0.0/23           [110] area: 0.0.0.0
                           via 10.0.10.25, r1-eth3
N    10.0.2.0/23           [10] area: 0.0.0.0
                           directly attached to r1-eth4
N    10.0.4.0/23           [10] area: 0.0.0.0
                           directly attached to r1-eth1
N    10.0.8.0/23           [120] area: 0.0.0.0
                           via 10.0.10.25, r1-eth3
N    10.0.10.0/23          [10] area: 0.0.0.0
                           directly attached to r1-eth3
N    10.0.12.0/23          [120] area: 0.0.0.0
                           via 10.0.10.25, r1-eth3

============ OSPF router routing table =============

============ OSPF external routing table ===========
```

Figura 27: Tabela de roteamento de r1

Conforme se pode perceber, o custo da rota para a rede 10.0.12.0/23 é de 120, já que há o custo da interface só afeta o custo da rota caso seja a interface de saída do roteador que a contém. Dessa forma, o custo de *r1* a *r5* é de 10, de *r5* a *r4* é de 100 e, finalmente, de *r4* a *y1* é de 10.

Por fim, conforme se vê abaixo, foram capturadas mensagens de diferentes tipos no *wireshark* durante todo o processo citado neste exercício. A mais comum é a mensagem de *Hello* do *OSPF*, usada para comunicação de permanência na rede pelos roteadores e transferência de estado de vizinhança entre eles (*Figura 28*). Em seguida, tem-se também mensagens de atualização (*Figura 29*), transmitidas pelo roteador cuja interface teve custo alterado em multicast para atualizar as tabelas de roteamento dos demais roteadores. Acompanhando o pacote anterior, tem-se os pacotes “*Acknowledge*” (*Figura 30*), que mostram reconhecimento dos vizinhos a respeito da mudança. Finalmente, é importante citar os pacotes *MDNS* responsáveis por transmitir coordenações entre roteadores a respeito dos multicasts feitos pelo *OSPF* (*Figura 31*).

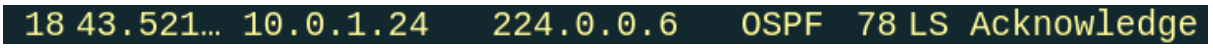


1	0.0000…	10.0.1.24	224.0.0.5	OSPF	86	Hello Packet
2	0.0019…	10.0.1.25	224.0.0.5	OSPF	86	Hello Packet
3	0.0021…	10.0.1.26	224.0.0.5	OSPF	86	Hello Packet
4	10.007…	10.0.1.24	224.0.0.5	OSPF	86	Hello Packet
5	10.008…	10.0.1.25	224.0.0.5	OSPF	86	Hello Packet
6	10.008…	10.0.1.26	224.0.0.5	OSPF	86	Hello Packet
7	20.009…	10.0.1.24	224.0.0.5	OSPF	86	Hello Packet
8	20.012…	10.0.1.26	224.0.0.5	OSPF	86	Hello Packet
9	20.015…	10.0.1.25	224.0.0.5	OSPF	86	Hello Packet
10	30.011…	10.0.1.24	224.0.0.5	OSPF	86	Hello Packet
11	30.014…	10.0.1.25	224.0.0.5	OSPF	86	Hello Packet

Figura 28: Mensagens Hello em multicast

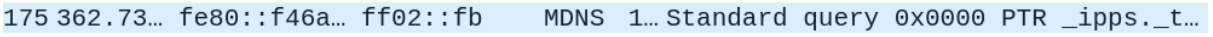
16	43.124…	10.0.1.25	224.0.0.5	OSPF	1…	LS Update

Figura 29: Mensagens de Update OSPF



18	43.521…	10.0.1.24	224.0.0.6	OSPF	78	LS Acknowledge

Figura 30: Mensagens de Acknowledge OSPF



175	362.73…	fe80::f46a…	ff02::fb	MDNS	1…	Standard query 0x0000 PTR _ipps._t…

Figura 31: Mensagens MDNS

Conclusão)

Após todo o processo experimental, pode-se afirmar que o objetivo de se estudar mais profundamente o protocolo de roteamento intra rede *OSPF* foi muito bem sucedido. Isso porque, pôde-se não só analisar o *OSPF* em ação, mas também pô-lo em ação, no sentido de configurar roteadores para usá-lo.