Lab 6: Roteamento Dinâmico OSPF

Rodrigo Seiji Piubeli Hirao (186837)

16 de dezembro de 2021

Conteúdo

1 Introdução

Nest laboratório será introduzido e estudado o protocolo **OSPF** para roteamento de uma rede, bem como suas ferramentas linux **quagga** (com **zebra** e **ospfd**).

2 Metodologia

2.1 Atividade 1

Foi emulado o sistema da figura 01 e estudado sua implementação, bem como os processos do quagga.

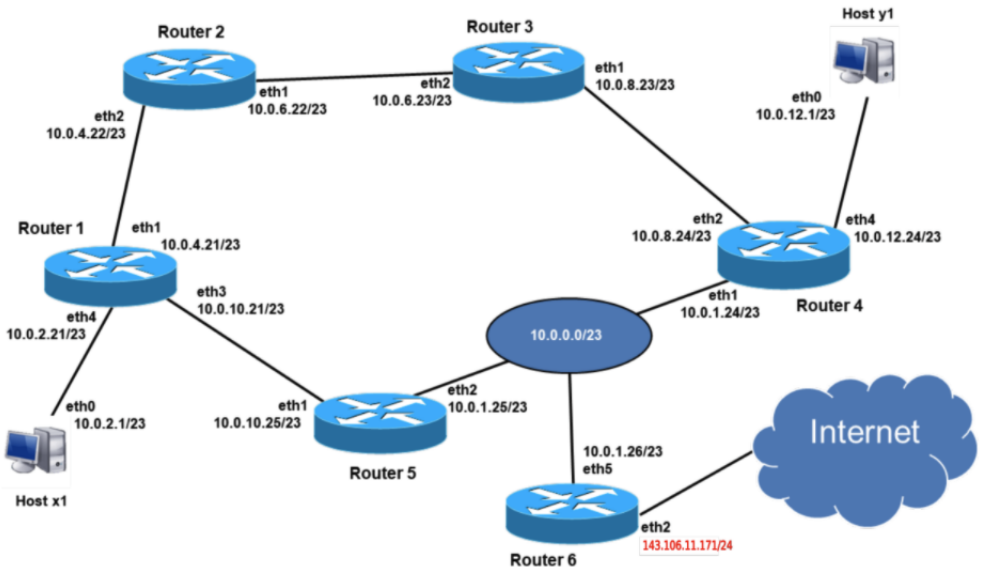

Figura 1: Topologia utilizada no experimento.

2.2 Atividade 2

Foi então ligado o OSPF no roteador **r1** e visto as tabelas que serão enviadas para quais vizinhos.

2.3 Atividade 3

Foi repetido o mesmo procedimento, mas no roteador **r2**, sendo que este está agora com sua tabela de roteamento atualizada com os dados enviados por **r1**.

2.4 Atividade 4

Foi configurado o script do ospf dos demais roteadores de forma que estes mostrassem suas interfaces, como pode ser visto no script a seguir do **ospfd-r3.conf**

```
4 router ospf
5         network 10.0.6.23/23 area 0
6         network 10.0.8.23/23 area 0
7 !
```

2.5 Atividade 5

Para a última atividade foi testada a conexão entre **x1** e **y1**, em seguida foi alterado o peso da interface **r5-eth1** para 100 e testada novamente a conexão, então foi feita a mesma cois para **r5-eth2**.

3 Resultados e Discussão

3.1 Atividade 1

Foi gerada a topologia corretamente, como pode ser visto pelo comando **net** do mininet

```
mininet> net
h1 h1-eth0:sw2_1-eth1
h2 h2-eth0:sw2_2-eth1
h3 h3-eth0:sw2_3-eth1
h4 h4-eth0:sw2_4-eth1
h5 h5-eth0:sw2_5-eth1
x1 x1-eth0:r1-eth4
y1 y1-eth0:r4-eth4
r1 r1-eth1:r2-eth2 r1-eth3:r5-eth1 r1-eth4:x1-eth0
r2 r2-eth2:r1-eth1 r2-eth1:r3-eth2
r3 r3-eth2:r2-eth1 r3-eth1:r4-eth2
r4 r4-eth1:sw3_1-eth2 r4-eth2:r3-eth1 r4-eth4:y1-eth0
r5 r5-eth2:sw3_1-eth3 r5-eth1:r1-eth3
r6 r6-eth5:sw3_1-eth4
sw2_1 lo:  sw2_1-eth1:h1-eth0 sw2_1-eth2:sw2_5-eth2
sw2_2 lo:  sw2_2-eth1:h2-eth0 sw2_2-eth2:sw2_5-eth3
sw2_3 lo:  sw2_3-eth1:h3-eth0 sw2_3-eth2:sw2_5-eth4
sw2_4 lo:  sw2_4-eth1:h4-eth0 sw2_4-eth2:sw2_5-eth5
sw2_5 lo:  sw2_5-eth1:h5-eth0 sw2_5-eth2:sw2_1-eth2 sw2_5-eth3:sw2_2-eth2
sw2_5-eth4:sw2_3-eth2 sw2_5-eth5:sw2_4-eth2 sw2_5-eth6:sw3_1-eth1
sw3_1 lo:  sw3_1-eth1:sw2_5-eth6 sw3_1-eth2:r4-eth1 sw3_1-eth3:r5-eth2
sw3_1-eth4:r6-eth5
c0
```

E foram descobertos 12 processos, sendo que são 2 processos distintos, zebra (o fornecedor do protocolo) e ospf (o protocolo utilizado), que executam em cada roteador.

```
wifi@wifi-virtualbox:~/EA080-2S2021/lab5$ sudo ps aux | grep quagga
quagga      4343  0.0  0.0   5140  3256 ?        Ss   10:39   0:00
/usr/lib/quagga/zebra -f confs/r1/zebra-r1.conf -d -i /tmp/zebra-r1.pid
quagga      4347  0.0  0.0   5216  2960 ?        Ss   10:39   0:00
/usr/lib/quagga/ospfd -f confs/r1/ospfd-r1.conf -d -i /tmp/ospf-r1.pid
quagga      4656  0.0  0.0   5140  3084 ?        Ss   10:39   0:00
/usr/lib/quagga/zebra -f confs/r2/zebra-r2.conf -d -i /tmp/zebra-r2.pid
quagga      4660  0.0  0.0   5212  3032 ?        Ss   10:39   0:00
/usr/lib/quagga/ospfd -f confs/r2/ospfd-r2.conf -d -i /tmp/ospf-r2.pid
quagga      4802  0.0  0.0   5140  3104 ?        Ss   10:39   0:00
/usr/lib/quagga/zebra -f confs/r3/zebra-r3.conf -d -i /tmp/zebra-r3.pid
quagga      4804  0.0  0.0   5216  1180 ?        Ss   10:39   0:00
/usr/lib/quagga/ospfd -f confs/r3/ospfd-r3.conf -d -i /tmp/ospf-r3.pid
quagga      4912  0.0  0.0   5136  3104 ?        Ss   10:39   0:00
/usr/lib/quagga/zebra -f confs/r4/zebra-r4.conf -d -i /tmp/zebra-r4.pid
quagga      4914  0.0  0.0   5212  1180 ?        Ss   10:39   0:00
/usr/lib/quagga/ospfd -f confs/r4/ospfd-r4.conf -d -i /tmp/ospf-r4.pid
quagga      4916  0.0  0.0   5140  3108 ?        Ss   10:39   0:00
/usr/lib/quagga/zebra -f confs/r5/zebra-r5.conf -d -i /tmp/zebra-r5.pid
quagga      4918  0.0  0.0   5216  1184 ?        Ss   10:39   0:00
/usr/lib/quagga/ospfd -f confs/r5/ospfd-r5.conf -d -i /tmp/ospf-r5.pid
quagga      4958  0.0  0.0   5140  3272 ?        Ss   10:39   0:00
/usr/lib/quagga/zebra -f confs/r6/zebra-r6.conf -d -i /tmp/zebra-r6.pid
quagga      4960  0.0  0.0   5216  1180 ?        Ss   10:39   0:00
/usr/lib/quagga/ospfd -f confs/r6/ospfd-r6.conf -d -i /tmp/ospf-r6.pid
wifi        6165  0.0  0.0  10720  2976 pts/21   S+   10:46   0:00 grep
--color=auto quagga
```

Porém não foi possível conectar x1 a y1, o que ocorre pois as tabelas de roteamento dos roteadores ainda não tem ops gateways configurados, como pode ser vito em r1.

```
mininet> x1 ping -c3 y1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.
From 10.0.2.21 icmp_seq=1 Destination Net Unreachable
From 10.0.2.21 icmp_seq=2 Destination Net Unreachable
^C
--- 10.0.12.1 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1011ms

mininet> x1 tracepath -n y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.21                                             0.043ms !N
 1:  10.0.2.21                                             0.011ms !N
     Resume: pmtu 1500

mininet> r1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
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
10.0.10.0       0.0.0.0         255.255.254.0   U     0      0        0 r1-eth3
mininet>
```

3.2 Atividade 2

Foram identificadas 8 subredes:

- 10.0.0.0
- 10.0.1.0
- 10.0.2.0
- 10.0.4.0
- 10.0.6.0
- 10.0.8.0
- 10.0.10.0
- 10.0.12.0

Pelo ospf é possível ver que ele apenas adicionou as rotas diretamente ligadas com o roteador r1, o que se mostra idêntico ao **r1 route** anteriormente mostrado, porém o único vizinho visto é o roteador **r2** em **r1-eth1** que foi dado um endereço na mesma subrede.

```
ospfd-r1# sh ip ospf route

============ OSPF network routing table ============
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    10.0.10.0/23          [10] area: 0.0.0.0
                           directly attached to r1-eth3

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

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

```
ospfd-r1# sh ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface
10.0.6.22         1 Full/DR           34.740s 10.0.4.22       r1-eth1:10.0.4.21
```

3.3 Atividade 3

[commandchars=
{}]

Pode ser visto as rotas anunciadas por **r1** presentes na tabela de roteamento de **r2**

```
mininet> r2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
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
10.0.10.0       10.0.4.21       255.255.254.0   UG    20     0        0 r2-eth2
```

E também é aparente a semelhança entre tabela de roteamento do OSPF e do **r2 route**.
Porém o único vizinho encontrado a **r2** é o **r1**.

```
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
10.0.4.21         1 Full/Backup       39.630s 10.0.4.21       r2-eth2:10.0.4.22
```

Ao ouvir por execuções de tcp em **r2** foi visto 2 pacotes OSPFv2 de tamanho 48, cada um com um roteador, no caso **r1** com o vizinho **r2** e **r2** com o vizinho **r1**

```
mininet> r2 timeout 10 tcpdump -i r2-eth2 -vvln
tcpdump: listening on r2-eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
12:13:06.515134 IP (tos 0xc0, ttl 1, id 644, offset 0, flags [none], proto OSPF (89), leng
    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
12:13:06.515209 IP (tos 0xc0, ttl 1, id 776, offset 0, flags [none], proto OSPF (89), leng
    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
```

3.4 Atividade 4

Após a configuração e habilitação do roteador **r3** foi obtido o resultado a seguir

```
ospfd-r3> sh ip ospf route

============ OSPF network routing table ============
N    10.0.0.0/23           [20] area: 0.0.0.0
                           via 10.0.8.24, r3-eth1
N    10.0.2.0/23           [30] area: 0.0.0.0
                           via 10.0.6.22, r3-eth2
N    10.0.4.0/23           [20] area: 0.0.0.0
                           via 10.0.6.22, r3-eth2
N    10.0.6.0/23           [10] area: 0.0.0.0
                           directly attached to r3-eth2
N    10.0.8.0/23           [10] area: 0.0.0.0
                           directly attached to r3-eth1
N    10.0.10.0/23          [30] area: 0.0.0.0
                           via 10.0.8.24, r3-eth1
                           via 10.0.6.22, r3-eth2
N    10.0.12.0/23          [20] area: 0.0.0.0
                           via 10.0.8.24, r3-eth1

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

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

Onde foi destacado o custo de 30 para a subrede **10.0.2.0/23**, o que é explicado pelos 3 hops que devem ser feitos (**r2**, **r1**, **x1**) sendo cada pulo um custo adicional de 10.

3.5 Atividade 5

Após a configuração de todos os roteadores foi testada a conexão entre os hosts **x1** e **y1**, que mostrou um caminho de 4 pulos, passando por 4 subredes (**x1** -> **r1** -> **r5** -> **r4** -> **y1**)

```
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=14.9 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=61 time=0.268 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=61 time=0.123 ms

--- 10.0.12.1 ping statistics ---
3 packets transmitted, 3 received, 0\% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.123/5.106/14.928/6.945 ms
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.21                                             0.042ms
 1:  10.0.2.21                                             0.011ms
```

```
 2:  10.0.10.25                                            0.019ms
 3:  10.0.1.24                                            14.116ms
 4:  10.0.12.1                                            11.231ms reached
     Resume: pmtu 1500 hops 4 back 4
```

Após a mudança do peso de peso de **r5-eth1** para 100, houve uma mudança nas rotas, o que fez com que o caminho de **x1** para **y1** fosse 4 pulos, mas o caminho de volta fosse 5, para não utilizar **r5-eth1**, assim o caminho de ida continua sendo **x1 ->** **r1 ->** **r5 ->** **r4 ->** **y1**, mas o de volta passa a ser **y1 ->** **r4 ->** **r3 ->** **r2 ->** **r1 ->** **x1**, assim o ttl maior em 1 de **y1** para **x1** é justificado pelo hop a mais na ida.

```
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=0.507 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=60 time=0.101 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=60 time=0.145 ms
64 bytes from 10.0.12.1: icmp_seq=4 ttl=60 time=0.130 ms
64 bytes from 10.0.12.1: icmp_seq=5 ttl=60 time=0.122 ms
64 bytes from 10.0.12.1: icmp_seq=6 ttl=60 time=0.127 ms
64 bytes from 10.0.12.1: icmp_seq=7 ttl=60 time=0.275 ms
64 bytes from 10.0.12.1: icmp_seq=8 ttl=60 time=0.169 ms
64 bytes from 10.0.12.1: icmp_seq=9 ttl=60 time=0.092 ms
64 bytes from 10.0.12.1: icmp_seq=10 ttl=60 time=0.091 ms

--- 10.0.12.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9221ms
rtt min/avg/max/mdev = 0.091/0.175/0.507/0.121 ms
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=6.05 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=61 time=0.238 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=61 time=0.146 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=61 time=0.098 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=61 time=0.093 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=61 time=0.123 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=61 time=0.133 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=61 time=0.149 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=61 time=0.096 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=61 time=0.093 ms

--- 10.0.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9177ms
rtt min/avg/max/mdev = 0.093/0.721/6.049/1.776 ms
mininet> x1 tracepath y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  ???                                                    0.043ms
 1:  ???                                                    0.014ms
 2:  ???                                                    4.866ms asymm  5
 3:  ???                                                   10.527ms asymm  4
 4:  ???                                                    5.758ms reached
     Resume: pmtu 1500 hops 4 back 5
mininet> y1 tracepath x1
 1?: [LOCALHOST]                      pmtu 1500
 1:  ???                                                    0.058ms
 1:  ???                                                    0.015ms
 2:  ???                                                    0.021ms
 3:  ???                                                    0.024ms
 4:  ???                                                    8.211ms asymm  3
 5:  ???                                                    6.514ms reached
     Resume: pmtu 1500 hops 5 back 4
```

Após a mudança do peso de peso de **r5-eth2** também para 100, o caminho de ida e volta de **x1** e **y1** passam a ser iguais (**y1 -> r4 -> r3 -> r2 -> r1 -> x1**), assim o ttl dos 2 passa a ser igual novamente.

```
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=60 time=0.065 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=60 time=0.089 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=60 time=0.080 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=60 time=0.099 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=60 time=0.691 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=60 time=0.069 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=60 time=0.083 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=60 time=0.165 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=60 time=0.077 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.151 ms

--- 10.0.2.1 ping statistics ---
10 packets transmitted, 10 received, 0\% packet loss, time 9211ms
rtt min/avg/max/mdev = 0.065/0.156/0.691/0.180 ms
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.21                                             0.043ms
 1:  10.0.2.21                                             0.010ms
 2:  10.0.4.22                                             0.018ms
 3:  10.0.6.23                                             0.020ms
 4:  10.0.8.24                                             0.022ms
 5:  10.0.12.1                                             0.024ms reached
     Resume: pmtu 1500 hops 5 back 5
```

Então, com o link entre **r2** e **r3** desligado, o caminho de **x1** a **y1** passa a ser, novamente, o caminho por **r5** (**x1 -> r1 -> r5 -> r4 -> y1**)

```
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.21                                             0.052ms
 1:  10.0.2.21                                             0.010ms
 2:  10.0.10.25                                            0.020ms
 3:  10.0.1.24                                            16.976ms
 4:  10.0.12.1                                            16.839ms reached
     Resume: pmtu 1500 hops 4 back 4

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
10.0.8.0        10.0.10.25      255.255.254.0   UG    20     0        0 r1-eth3
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
```

Porém foi visto também que o gasto de **r1** a **y1** passou a ser 120, pois este deve passar por **r5** (+100) e mais 2 hops (+20).

```
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 ===========
```

Durante as medidas foram percebidos diversos pacotes do OSPF sendo recebidos em **r4** (**r4** foi apenas o exemplo utilizado, os pacotes devem ser enviados a todos os roteadores), dentre eles se destacam os pacotes de

- **Hello Packet** - pacotes enviados por um roteador com informações de sua vizinhança (enviado a cada 10 segundos)
- **LS Update** - listando os links atualizados para todos os roteadores (chamadas quando os pesos das interfaces de **r5** foram mudados e quando o link de **r2** e **r3** for derrubado)
- **LS Acknowledge** - indicando que o LS Update foi recebido (respondido após um **LS Update**)

4 Conclusão

Pôde ser visto o funcionamento eficiente do **OSPF** em achar o menor caminho em tempo real se comunicando entre os roteadores por meio de atualizações sempre que uma interface tem alguma alteração, além de pacotes de "hello", que garantem a confiabilidade do sistema como um todo.

Também foi estudado o funcionamento de ferramentas linux como o **quagga**, uma suite de aplicativos como o **zebra**, que gerencia o **ospfd** (nosso daemon de ospf). Configurando tais interfaces por meio da cli **telnet**.