EA080 - O — Roteamento Dinâmico OSPF (Open Shortest Path First)

Professor: Christian Esteve Rothenberg

Leonardo Rodrigues Marques RA: 178610

1 Introdução

Nesse quarto laboratório de redes de computadores, fomos introduzidos ao algoritmo de roteamento OSPF. Open Shortest Pathe First é um protocolo de roteamento para redes que operam com protocolo IP. Atualmente o OSPF é um dos protocolos de roteamento mais empregados, sendo suportado pela maioria dos roteadores, assim como por servidores que implementem os sistemas operacionais Linux e Unix. Valendo-se da importância desse protocolo, fomos orientados a entender seu funcionamento (adaptabilidade e consistência) e a conduzir testes de desempenho para reconhecer as dificuldades enfrentadas na busca de caminhos mais eficientes para transmissão de pacotes.

2 Metodologia

A metodologia usada para desenvolver esse trabalho consistiu em alguns fundamentos de apoio. Em primeiro lugar, os anexos disponíveis no roteiro foram essenciais para entendimento da aplicação *Quagga* e suas daemons adicionais como OSPFD, usado nesse laboratório. Fundamentado, a próxima etapa consistia em executar o o projeto e aplicar novos comandos a fim de interagir e reconhecer os padrões de formação das tabelas de roteamento dos roteadores. Finalmente, fomos levados a alterar as propriedades das interfaces dos roteadores e observar como o algoritmo se comporta na manutenção da consistência da rede, sempre levando em consideração do ping entre os hosts x1 e y1.

3 Resultado, Discussões e Conclusões

3.1 Questão 1

3.1.1

Após executar a topologia do arquivo **topo_A3.py** e verificar a criação do nós e links, um novo terminal foi aberto e o comando `sudo ps aux | grep quagga` executado. Constatei que 12 processos relacionados ao quagga estavam sendo executados.

```
leonardo@leonardo-PC:~/Unicamp/EA080/lab-4/code-git$ ps aux | grep quagga
quagga   17763  0.0  0.0  27804  3244 ?        Ss   11:11   0:00 /usr/sbin/zebra
quagga   17765  0.0  0.0  29936  3092 ?        Ss   11:11   0:00 /usr/sbin/ospfd
quagga   17846  0.0  0.0  27804  3340 ?        Ss   11:11   0:00 /usr/sbin/zebra
quagga   17851  0.0  0.0  29936  3076 ?        Ss   11:11   0:00 /usr/sbin/ospfd
quagga   17917  0.0  0.0  27808  3428 ?        Ss   11:11   0:00 /usr/sbin/zebra
quagga   17919  0.0  0.0  29936   976 ?        Ss   11:11   0:00 /usr/sbin/ospfd
quagga   18121  0.0  0.0  27800  3320 ?        Ss   11:12   0:00 /usr/sbin/zebra
quagga   18123  0.0  0.0  29932  3168 ?        Ss   11:12   0:00 /usr/sbin/ospfd
quagga   18125  0.0  0.0  27804  3348 ?        Ss   11:12   0:00 /usr/sbin/zebra
quagga   18127  0.0  0.0  29932   972 ?        Ss   11:12   0:00 /usr/sbin/ospfd
quagga   18153  0.0  0.0  27804  3292 ?        Ss   11:12   0:00 /usr/sbin/zebra
quagga   18155  0.0  0.0  29932   972 ?        Ss   11:12   0:00 /usr/sbin/ospfd
leonardo 18536  0.0  0.0  21532  1152 pts/22   S+   11:12   0:00 grep --color=aut
leonardo@leonardo-PC:~/Unicamp/EA080/lab-4/code-git$
```

Figura 1: Processos Quagga.

3.1.2

Quagga é um suíte composta por um daemon principal chamada zebra e um daemon adicional responsável pelo roteamento dinâmico, no caso ospfd. Existem 12 processos relacionados ao Quagga, no qual 6 estão associados ao Zebra e 6 ao Ospfd. Cada algoritmo de roteamento exige um processo da Zebra e um processo do Ospfd, oque nos permite concluir que 6 algoritmos de roteamento estão sendo executado.

3.1.3

Ao executar o comando `x1 ping -c3 y1`, obtivemos perda de 100% dos pacotes. Para o comando `x1 tracepath y1`, não obtivemos uma alcance para o endereço IP de **y1**. Isso mostra que não há conectividade entre x1 e y1.

```
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
From 10.0.2.21 icmp_seq=3 Destination Net Unreachable

--- 10.0.12.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2032ms

mininet>
```

Figura 2: Comando Ping entre x1 e y1.

```
mininet> x1 tracepath y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                              0.175ms !N
 1:  _gateway                                              0.147ms !N
     Resume: pmtu 1500
mininet>
```

Figura 3: Comando Tracepath entre x1 e y1.

3.2 Questão 2

3.2.1

Na topologia experimental com Protocolo OSPF, existe apenas 1 sub-rede: 10.0.0.0/23.

3.2.2

A rota 10.0.6.0 difere das outras rotas pelos seguintes motivos: há um gateway especificado diferente de 0.0.0.0, o campo de flags mostrado está UG (up and gateway) e o campo metric está com o valor 20.

```
root@wifi-VirtualBox:~/lab4# route -n
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
10.0.6.0        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
root@wifi-VirtualBox:~/lab4#
```

Figura 4: Tabela de roteamento IP do roteador 1.

3.2.3

Analisando a topologia da figura 1 do roteiro 4, é possível observar que o roteador 1 possui dois roteadores vizinhos (2 e 5) e um host. Os endereços das interfaces dos roteadores vizinhos são **r1-eht1: 10.0.4.21 (2); r1-eth3: 10.0.10.21 (5)**. O roteador **2** está conectado pela interface **10.0.4.22** e o roteador **5** está conectado pela interface **10.0.10.25**. Entretanto ao executarmos os comandos `sh ip ospf route` e `sh ip ospf neighbor`, obtivemos uma tabela de roteamento indicando as rotas para sub-redes aos quais os roteadores vizinhos estão inseridos e uma tabela de roteamento externo em que apenas o roteador vizinho 2 é detalhado com informações de interface. Essas informações de interface condizem com as informações da topologia da figura 1 do rotiero 4.

```
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    10.0.6.0/23           [20] area: 0.0.0.0
                           via 10.0.4.22, 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           RXmtL RqstL DBsmL
10.0.6.22         1 Full/DR           35.178s 10.0.4.22       r1-eth1:10.0.4.21       0     0     0
ospfd-r1#
```

Figura 5: Tabelas de Roteamento de R1.

3.2.4

Ambos os comandos `route n` e `sh ip ospf route` se assemelham ao mostrarem aspectos idênticos na atribuição de endereços IP as rotas e interfaces.

3.2.5

Com base na topologia da figura 4, concluímos que o roteador 2 é responsável pela rota diferente.

3.3 Questão 3

3.3.1

Aos executarmos o mesmos comandos de r1 para r2, obtemos duas rotas diferentes: **10.0.2.0 e 10.0.10 .0**. Essas rotas informam um endereço IP de gateway e apresentam a flag UG ao invés de U. Provavelmente, são rotas externas conhecidas pelo roteador 2.

```
root@wifi-VirtualBox:~/lab4# route -n
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
```

Figura 6: Tabela de Roteamento de R2.

3.3.2

Considerando apenas as tabelas obtidas pelos comandos, concluímos que o roteador 2 possui apenas 1 vizinho. O endereço da interface do roteador vizinho é **r2-eth:10.0.4.22** e ele está conectado pela interface **10.0.4.21**.

```
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       37.111s 10.0.4.21       r2-eth2:10.0.4.22        0     0     0
ospfd-r2#
```

Figura 7: Caption

3.3.3

Ambos os comandos `route n` e `sh ip ospf route` se assemelham ao mostrarem aspectos idênticos na atribuição de endereços IP as rotas e interfaces.

3.3.4

Com base na topologia da figura 4, concluímos que o roteador 1 é responsável pelas rotas diferentes.

3.3.5

O pacote OSPF enviado é do tipo Hello. O endereço listado em Neighbor list é 10.0.6.22. Ele representa uma interface vizinha r2-eth1 em relação a interface r2-eth2 que está se comunicando com a interface r1-eth1: 10.0.4.21 através de um multicast.

```
root@wifi-VirtualBox:~/lab4# timeout 10 tcpdump -i r2-eth2 -vvlan
tcpdump: listening on r2-eth2, link-type EN10MB (Ethernet), capture size 262144
bytes
16:39:11.640402 IP (tos 0xc0, ttl 1, id 9559, 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
16:39:11.640527 IP (tos 0xc0, ttl 1, id 10590, offset 0, flags [none], proto OSP
F (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 8: Captura de Pacotes OSPF em r2-eth2.

3.4 Questão 4

3.4.1

Ao executarmos os comandos para solicitar as tabelas de roteamento dos roteadores r3, r4, r5 e r6, não foi possível obter nenhuma informação por falta de configuração nos arquivos do diretório confs/.

Figura 9: Tabela de Roteamento de r3.

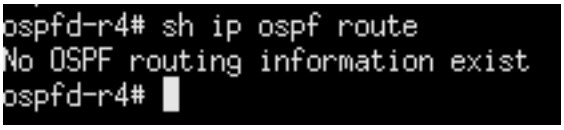

Figura 10: Tabela de Roteamento de r4.

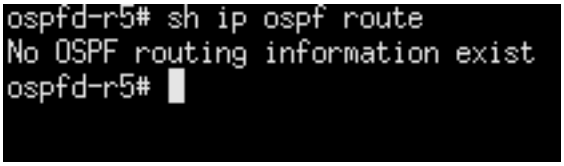

Figura 11: Tabela de Roteamento de r5.

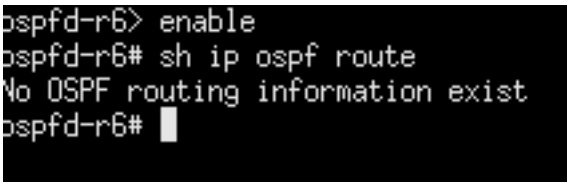

Figura 12: Tabela de Roteamento de r6.

Tomando como base o rotador r3, foi acrescentando duas linhas de configuração no arquivo confs/r1/ospfd-r3.conf: `network 10.0.8.23/23 area 0 network 10.0.6.23/23 area 0`.

```
hostname ospfd-r3
password quagga
!
router ospf
        network 10.0.8.23/23 area 0
        network 10.0.6.23/23 area 0
!
```

Figura 13: Arquivo de confiuração de r3 pós-edição.

Após edição e reexecução do mininet, é possível ver a tabela de roteamento de r3.

```
ospfd-r3# sh ip ospf route
============ OSPF network routing table ============
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 10.0.6.22, r3-eth2

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

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

ospfd-r3# sh ip ospf neighbor

    Neighbor ID Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
10.0.6.22         1 Full/Backup       37.672s 10.0.6.22       r3-eth2:10.0.6.23        0     0     0
ospfd-r3#
```

Figura 14: Tabela de Roteamento de r3 pós-edição.

A rota de rede para 10.0.2.0/23 possui custo 30 pois é necessário passar por **3** roteadores até chegar na sub-rede em questão, sendo que cada relação de transmissão do roteador vale 10.

3.5 Questão 05

3.5.1

Após a configuração das rotas, foi feito um teste de ping entre x1 e y1. O teste mostrou conectividade entre os hosts.

```
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=2.31 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=61 time=0.463 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=61 time=0.078 ms

--- 10.0.12.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.078/0.952/2.316/0.977 ms
mininet>
```

Figura 15: Teste de Ping entre x1 e y1.

Utilizando-se o comando `x1 tracepath y1`, observamos que o número de saltos foi 4. O caminho dos roteadores segui a seguinte ordem: **r1 r5 r4**.

```
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.078/0.952/2.316/0.977 ms
mininet> x1 tracepath y1
 1?: [LOCALHOST]                                         pmtu 1500
 1:  10.0.2.21                                             0.052ms
 1:  10.0.2.21                                             0.018ms
 2:  10.0.10.25                                            0.024ms
 3:  10.0.1.24                                             3.870ms
 4:  10.0.12.1                                             5.033ms reached
     Resume: pmtu 1500 hops 4 back 4
mininet>
```

Figura 16: Teste de Caminho entre x1 e y1.

Ao adicionar custo 100 na interface **r5-eth1**, houve uma redução de rtt para uma média de 0.255 s.

```
--- 10.0.12.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9003ms
rtt min/avg/max/mdev = 0.085/0.197/0.950/0.257 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=1.50 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=61 time=0.377 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=61 time=0.086 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=61 time=0.092 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=61 time=0.086 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=61 time=0.095 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=61 time=0.064 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=61 time=0.070 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=61 time=0.093 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=61 time=0.088 ms

--- 10.0.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9004ms
rtt min/avg/max/mdev = 0.064/0.255/1.501/0.424 ms
mininet>
```

Figura 17: Teste de Ping entre x1 e y1 após adição de custo.

Gerando o comando `tracepath` entre os hosts, notamos que o caminho de x1 e y1 é diferente entre y1 e x1. Entre x1 e y1, persistem os 4 hops pelo mesmo caminho anteriormente citado. Entre y1 e x1, a quantidade de hops vai para 5 e o caminho segue a seguinte orientação: **r4 r3 r2 r1**.

```
mininet> y1 tracepath x1
 1?: [LOCALHOST]                                         pmtu 1500
 1:  10.0.12.24                                            0.038ms
 1:  10.0.12.24                                            0.011ms
 2:  10.0.8.23                                             0.017ms
 3:  10.0.6.22                                             0.021ms
 4:  10.0.10.21                                            1.701ms asymm  3
 5:  10.0.2.1                                              0.796ms reached
     Resume: pmtu 1500 hops 5 back 4
mininet> x1 tracepath y1
 1?: [LOCALHOST]                                         pmtu 1500
 1:  10.0.2.21                                             0.059ms
 1:  10.0.2.21                                             0.019ms
 2:  10.0.1.25                                             2.839ms asymm  5
 3:  10.0.8.24                                             2.245ms asymm  4
 4:  10.0.12.1                                             1.219ms reached
     Resume: pmtu 1500 hops 4 back 5
mininet>
```

Figura 18: Tracepath entre x1 - y1 e y1 - x1.

Figura 20: Tabela de Roteamento OSPF após inoperação de r2 e r3.

Adicionado custo 100 a interface **r5-eth2**, houve uma melhora no tempo rtt, que foi para uma média de 0.075 s.

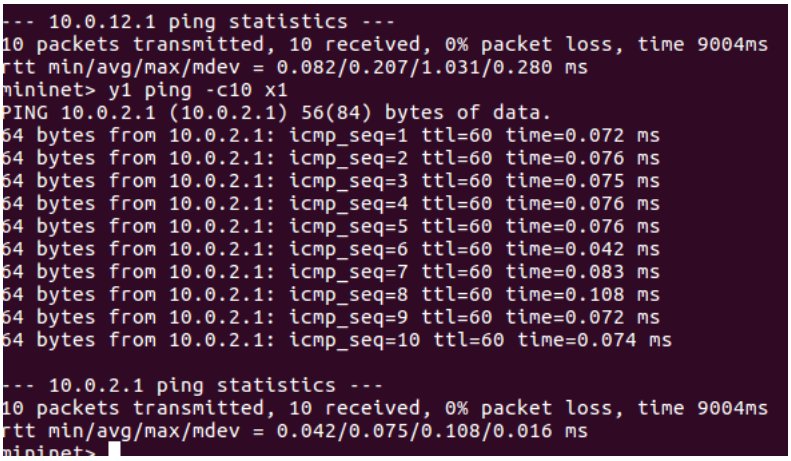

Figura 19: Ping entre y1 e x1 com nova adição de custo.

Finalmente, executamos o comando `tracepath` entre x1 e y1. Nessa situação, aconteceu 5 hops entre os hosts e o caminho abordado foi **r4 r3 r2 r1**.

Com o comando `link r2 r3 down`, os roteadores r2 e r3 ficaram inoperantes. Devido a isso, as tabelas de roteamento OSPF foram alteradas para manter a consistência da rede. Essa alteração mudou o caminho entre x1 e y1 passando pelos roteadores **r5** e **r4**, que apesar de terem maior custo, garantem a chegada do sinal.

Como os roteadores **r2** e **r3** ficaram inoperantes, a comunicação foi feita pelo caminho com maior custo. Esse custo foi definido pela adição do custo da interface (previamente definida em 100) a custo de saltos dos dois roteadores (no caso 20), totalizando em 120.

3.5.2

Pacote Hello: Os pacotes Hello são do pacote OSPF Tipo 1. Esses pacotes são multicast periodicamente para o endereço multicast 224.0.0.5 em todas as interfaces (unicast em links virtuais), permitindo a

descoberta dinâmica de vizinhos e mantendo relacionamentos de vizinhos.

Pacote Update: Os pacotes Link State Update (LSU) são pacotes OSPF Tipo 4. Esses pacotes implementam a inundação de LSAs. Cada LSA contém informações de roteamento, métrica e topologia para descrever uma parte da rede OSPF. O roteador local anuncia o LSA dentro de um pacote LSU para os roteadores vizinhos. Além disso, o roteador local anuncia o pacote LSU com informações em resposta a um pacote LSR.

Pacote Acknowledgment: Pacotes de reconhecimento de estado de link (LSAck) são pacote OSPF tipo 5. O OSPF requer reconhecimento para o recebimento de cada LSA. Vários LSAs podem ser reconhecidos em um único pacote LSAck.

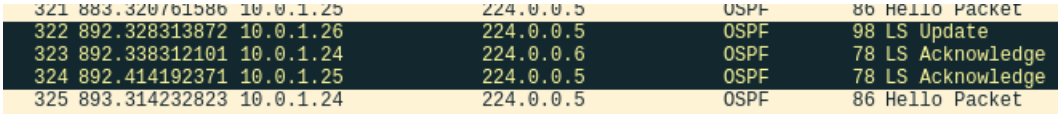

Figura 21: Tipos de Pacotes do OSPF.

Referências

https://sites.google.com/site/amitsciscozone/home/important-tips/ospf/ospf-packet-types

https://www.cisco.com/c/pt_br/support/docs/ip/open-shortest-path-first-ospf/13688-16.html