EA080 - Laboratório de redes de computadores Relatório 5 - Roteamento Dinâmico OSPF

Gabriela Surita RA 139095

Introdução

Configurar rotas IP entre roteadores é uma parte fundamental para o funcionamento correto de uma rede de computadores. Configurar rotas manualmente é em geral trabalhoso em escala e propenso a diversos erros humanos. Para evitar este trabalho, foram desenhados os chamados protocolos de configuração autônoma de redes. Sendo mais conhecidos:

Protocolos redes internas (ou *Interior Gateway Protocol* - IGP): usados na configuração de redes internas, como redes empresariais ou domésticas para configuração autônoma da topologia.

Protocolos de roteador de borda (ou *Border Gateway Protocol* - BGP): usados entre redes de internas pré-configuradas.

Dedicando o estudo aos protocolos IGP, destacam-se duas classes básicas de protocolos, cada qual com sua base algorítmica:

Protocolos Baseados em vetor de distâncias: baseados no algoritmo de Bellman-Ford, nestes protocolos, cada roteador armazena apenas a sua tabela de roteamento e os respectivos pesos (vetor de distâncias). Uma vez que acontece uma mudança conhecida na topologia, um roteador envia uma mensagem para seus vizinhos contendo seu vetor de distâncias atualizado. Quando um roteador recebe um vetor de distâncias, deve comparar o dado com seu próprio vetor e verificar se um caminho novo ou mais barato existe. Em caso positivo atualiza sua tabela de roteamento e envia seu vetor de distâncias para seus vizinhos. Os roteadores continuam enviando vetores de distância até que o sistema atinja um estado estável. Um exemplo de protocolo que usa esta estratégia é o RIP (Routing Information Protocol).

Protocolos baseados em estado de enlace: em geral baseados no algoritmo de Dijkstra, nestes protocolos, cada roteador armazena o estado de toda a topologia e é responsável por conhecer seus vizinhos. Uma vez que acontece uma mudança entre seus vizinhos, um roteador deve fazer um broadcast para toda a rede. Cada roteador deve receber o pacote e atualizar sua própria tabela de roteamento. Exemplos de

protocolos que usam esta estratégia são o OSPF (Open Shortest Path First) e o IS-IS (Intermediate system to intermediate system).

Estudar vantagens e desvantagens de cada uma das estratégias não é o foco deste experimento, mas protocolos baseados em estado de enlace se tornaram mais populares ao longo do tempo em redes de médio e grande porte em especial por problemas relacionados a convergência da topologia.

Neste experimento, estudamos o comportamento do protocolo baseado em estado de enlace OSPF em uma rede com redundância de conexões, verificando cenários como a alteração de custos dentro da topologia e a interrupção da conexão entre parte da topologia.

Implementação

1. Primeiros passos

Esta seção contém experimentos entendidos como necessários antes de nos debruçarmos sobre o estudo dos protocolos de roteamento propriamente ditos.

Estudo da topologia de rede

A topologia de rede utilizada durante este experimento foi previamente fornecida e foi emulada usando o software mininet. O código fonte com a descrição da topologia está disponível em https://github.com/vcleal/ea080.

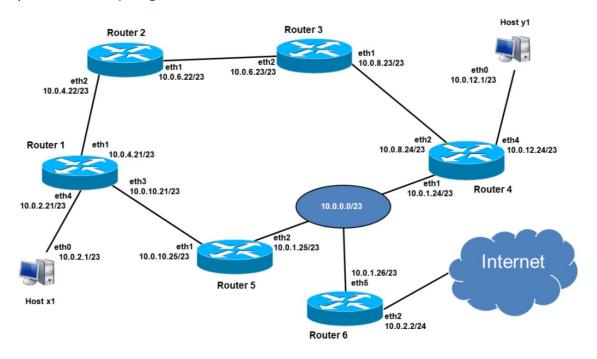


Figura 1: Esquema contendo roteadores e hosts na topologia da rede fornecida.

Podemos verificar através do comando nodes que a rede contém os seguintes elementos de rede:

- 7 hosts h1 a h5, x1 e y1
- 6 roteadores r1 a r6
- 5 switches sw2 1 a sw2 5

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 r1 r2 r3 r4 r5 r6 sw2_1 sw2_2 sw2_3 sw2_4 sw2_5 sw3_1 x1 y1
```

Também podemos verificar as conexões entre os elementos. Vemos que cada um dos hosts (h1 a h5) está conectado a um switch (sw2_1 a sw2_5), que por sua vez está conectado a um roteador (r1 a r5) respeitando a numeração. As exceções são os hosts x1 e y1, que estão conectados diretamente aos roteadores r1 e r4 respectivamente.

```
mininet> links
h1-eth0<->sw2_1-eth1 (OK OK)
h2-eth0<->sw2_2-eth1 (OK OK)
h3-eth0<->sw2_3-eth1 (OK OK)
h4-eth0<->sw2_4-eth1 (OK OK)
h5-eth0<->sw2_5-eth1 (OK OK)
r1-eth1<->r2-eth2 (OK OK)
r1-eth3<->r5-eth1 (OK OK)
r2-eth1<->r3-eth2 (OK OK)
r3-eth1<->r4-eth2 (OK OK)
sw2_1-eth2<->sw2_5-eth2 (OK OK)
sw2_2-eth2<->sw2_5-eth3 (OK OK)
sw2_3-eth2<->sw2_5-eth4 (OK OK)
sw2_4-eth2<->sw2_5-eth5 (OK OK)
sw2_5-eth6<->sw3_1-eth1 (OK OK)
sw3_1-eth2<->r4-eth1 (OK OK)
sw3_1-eth3<->r5-eth2 (OK OK)
sw3_1-eth4<->r6-eth5 (OK OK)
x1-eth0<->r1-eth4 (OK OK)
y1-eth0<->r4-eth4 (OK OK)
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
mininet>
```

Podemos verificar também os processos que estão rodando na máquina hospedeira da topologia. Observamos que existem dois processos executando para cada instância de roteador, sendo um ospfd e um zebra.

```
wifi@wifi-VirtualBox:~/ea080$ sudo ps aux | grep quagga
[sudo] password for wifi:
quagga
        25853 0.0 0.0 24368 2872 ?
                                              Ss
                                                   16:07
                                                          0:00
/usr/lib/quagga/zebra -f confs/r1/zebra-r1.conf -d -i /tmp/zebra-r1.pid
        25855 0.0 0.0 26984 2960 ?
                                                   16:07
                                              Ss
/usr/lib/quagga/ospfd -f confs/r1/ospfd-r1.conf -d -i /tmp/ospf-r1.pid
        25857 0.0 0.0 24368 2808 ?
                                              Ss
                                                   16:07
                                                          0:00
/usr/lib/quagga/zebra -f confs/r2/zebra-r2.conf -d -i /tmp/zebra-r2.pid
        25859 0.0 0.0 26984 2964 ?
                                              Ss
                                                   16:07
                                                          0:00
/usr/lib/quagga/ospfd -f confs/r2/ospfd-r2.conf -d -i /tmp/ospf-r2.pid
                                608 ?
        25861 0.0 0.0 24368
                                              Ss
                                                   16:07
                                                          0:00
/usr/lib/quagga/zebra -f confs/r3/zebra-r3.conf -d -i /tmp/zebra-r3.pid
quagga
        25863 0.0 0.0 26984 2868 ?
                                              Ss
                                                   16:07
                                                           0:00
/usr/lib/quagga/ospfd -f confs/r3/ospfd-r3.conf -d -i /tmp/ospf-r3.pid
        25865 0.0 0.0 24368 2824 ?
                                                   16:07
                                                           0:00
quagga
                                              Ss
/usr/lib/quagga/zebra -f confs/r4/zebra-r4.conf -d -i /tmp/zebra-r4.pid
        25867 0.0 0.0 26984 2872 ?
                                              Ss
                                                   16:07
                                                          0:00
/usr/lib/quagga/ospfd -f confs/r4/ospfd-r4.conf -d -i /tmp/ospf-r4.pid
        25869 0.0 0.0 24368 2868 ?
guagga
                                              Ss
                                                   16:07
                                                          0:00
/usr/lib/quagga/zebra -f confs/r5/zebra-r5.conf -d -i /tmp/zebra-r5.pid
        25871 0.0 0.0 26984 2804 ?
                                              Ss
                                                   16:07
                                                          0:00
/usr/lib/quagga/ospfd -f confs/r5/ospfd-r5.conf -d -i /tmp/ospf-r5.pid
        25873 0.0 0.0 24368 2768 ?
                                              Ss
                                                   16:07
quagga
/usr/lib/quagga/zebra -f confs/r6/zebra-r6.conf -d -i /tmp/zebra-r6.pid
        25875 0.0 0.0 26984
                               716 ?
                                              Ss
                                                   16:07
/usr/lib/quagga/ospfd -f confs/r6/ospfd-r6.conf -d -i /tmp/ospf-r6.pid
wifi 25972 0.0 0.0 21292 1084 pts/38 S+
                                              16:10
                                                       0:00 grep
--color=auto quagga
```

Testando a conectividade

Com alguns testes simples, podemos verificar que não existe conexão configurada 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 1998ms
mininet> x1 tracepath y1
 1?: [LOCALHOST]
                                                       pmtu 1500
                                                             0.029ms !N
 1:
    10.0.2.21
 1: 10.0.2.21
                                                             0.013ms !N
      Resume: pmtu 1500
mininet> y1 tracepath x1
 1?: [LOCALHOST]
                                                       pmtu 1500
                                                             0.049ms !N
 1:
    10.0.12.24
 1: 10.0.12.24
                                                             0.013ms !N
      Resume: pmtu 1500
```

2. Desbravando

Identificando subredes e interfaces

São no total 7 subredes na topologia, sendo que todas elas tem o formato 10.0.x.0/23, onde x pertence ao conjunto {0, 2, 4, 6, 8 10, 12}.

Nos debruçando em um único roteador, podemos estudar suas interfaces. Inspecionando r1, Vemos que o mesmo possuí 3 interfaces físicas e uma virtual de loopback, ou seja, apontando para o próprio host.

```
mininet> r1 ifconfig -a
lo Link encap:Local Loopback
   inet addr:127.0.0.1 Mask:255.0.0.0
   inet6 addr: ::1/128 Scope:Host
   UP LOOPBACK RUNNING MTU:65536 Metric:1
   RX packets:0 errors:0 dropped:0 overruns:0 frame:0
   TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:1
   RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

```
r1-eth1
         Link encap:Ethernet HWaddr c6:90:32:5c:9d:ca
     inet addr:10.0.4.21 Bcast:10.0.5.255 Mask:255.255.254.0
     inet6 addr: fe80::c490:32ff:fe5c:9dca/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:97 errors:0 dropped:0 overruns:0 frame:0
     TX packets:99 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:7834 (7.8 KB) TX bytes:8046 (8.0 KB)
         Link encap:Ethernet HWaddr ba:b0:6f:9c:25:db
     inet addr:10.0.10.21 Bcast:10.0.11.255 Mask:255.255.254.0
     inet6 addr: fe80::b8b0:6fff:fe9c:25db/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:8 errors:0 dropped:0 overruns:0 frame:0
     TX packets:87 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:648 (648.0 B) TX bytes:6714 (6.7 KB)
         Link encap:Ethernet HWaddr 96:96:08:c2:d2:a9
r1-eth4
     inet addr:10.0.2.21 Bcast:10.0.3.255 Mask:255.255.254.0
     inet6 addr: fe80::9496:8ff:fec2:d2a9/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:16 errors:0 dropped:0 overruns:0 frame:0
     TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:4144 (4.1 KB) TX bytes:8356 (8.3 KB)
```

Também podemos verificar a tabela de roteamento de r1, que possuí entradas para as suas subredes conhecidas.

```
mininet> r1 route -n
Kernel IP routing table
                                 Flags Metric Ref Use Iface
Destination Gateway
                      Genmask
10.0.2.0
           0.0.0.0
                      255.255.254.0
                                     U 0
                                            0
                                                  0 r1-eth4
10.0.4.0
                      255.255.254.0
                                     U 0
                                                  0 r1-eth1
           0.0.0.0
                                             0
                                            0
10.0.6.0
          10.0.4.22
                      255.255.254.0
                                     UG20
                                                  0 r1-eth1
10.0.10.0 0.0.0.0
                      255.255.254.0
                                     U 0
                                             0
                                                  0 r1-eth3
```

Vemos que a conexão com a subrede 10.0.6.0 foi previamente configurada usando alguma estratégia, provavelmente com o protocolo OSPF ativado.

Ativando OSPF em R1

```
root@wifi-VirtualBox:~/ea080# telnet localhost ospfd
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
 Hello, this is Quagga (version 0.99.24.1).
Copyright 1996–2005 Kunihiro Ishiguro, et al.
 User Access Verification
 Password:
irectly attached to ri-ethi
[20] area; 0.0.0.0
via 10.0.4.22, ri-ethi
[10] area; 0.0.0.0
directly attached to ri-ethi
        10.0.10.0/23
  ======== OSPF router routing table =========
   ======= OSPF external routing table =========
 ospfd-r1# sh ip ospf neighbor
      Neighbor ID Pri State
0.6.22 1 Full/DR
                                                     Dead Time Address
30.750s 10.0.4.22
                                                                                           Interface
                                                                                                                         RXmtL RqstL DBsmL
10.0.6.22 1 Full/DR
ospfd-r1# exit
Connection closed by foreign host,
root@wifi-VirtualBox:~/ea080#
                                                                                          r1-eth1:10.0.4.21
```

Ativando OSPF em R2

Podemos verificar também através do comando tcpdump que existe trocas de mensagens OSPF entre r1 e r2

```
root@wifi-VirtualBox: "/ea080# timeout 10 tcpdump -i r2-eth2 -vvln
tcpdump: listening on r2-eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
19;50;58.611307 IP (tos 0xc0, ttl 1, id 30241, 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
19;50;58,611410 IP (tos 0xc0, ttl 1, id 30644, 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 captured
2 packets received by filter
0 packets dropped by kernel
root@wifi-VirtualBox: "/ea080#
```

Ativando OSPF e configurando R3

Vale notar que existe uma configuração faltando nesta seção, que é discutida na seção **Depurando falhas de conectividade.**

```
ospfd–r4> enable
ospfd-r4# configure
% Command incomplete.
ospfd-r4# configure terminal
ospfd-r4(config)# router ospf
ospfd-r4(config-router)# network 10.0.8.24/23 area 0
ospfd-r4(config-router)# network 10.0.8.24/23 area 0
There is already same network statement.
ospfd-r4(config-router)# network 10.0.1.24/23 area 0
ospfd-r4(config-router)# exit
ospfd-r4(config)# sh ip ospf route
% [OSPF] Unknown command: sh ip ospf route
ospfd-r4(config)# router)# performer 10 0 12 24/23 area 0
ospfd-r4(config)# router)# performer 10 0 12 24/23 area 0
directly attached to r4-eth1
                                               10 area: 0.0.0.0 via 10.0.8.23, r4-eth2 [30] area: 0.0.0.0 via 10.0.8.23, r4-eth2 [20] area: 0.0.0.0 via 10.0.8.23, r4-eth2 via 10.0.8.23, r4-eth2 [10] area: 0.0.0.0 via 10.0.8.23, r4-eth2 [10] area: 0.0.0.0 via 10.0.8.23
        10.0.2.0/23
        10.0.4.0/23
        10.0.6.0/23
        10.0.8.0/23
                                                directly attached to r4-eth2
                                                [40] area: 0.0.0.0
via 10.0.8.23, r4-eth2
[10] area: 0.0.0.0
        10.0.10.0/23
        10.0.12.0/23
                                                directly attached to r4-eth4
======== OSPF router routing table =========
 ======== OSPF external routing table ========
ospfd-r4# sh ip ospf neighbor
      Neighbor ID Pri State
).8.23 1 Full/Backup
                                                               Dead Time Address
                                                                                                              Interface
                                                                                                                                                   RXmtL RqstL DBsmL
                                                                                                              r4-eth2:10.0.8.24
                                                                   32,460s 10.0.8.23
                                                                                                                                                          Ů.
```

Neste ponto, espera-se que y1 seja acessível a partir de x1, mas isso não foi o comportamento observado. A seção a seguir se destina e entender e resolver o problema de conectividade identificado.

```
mininet> x1 ping 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
From 10.0.2.21 icmp_seq=4 Destination Net Unreachable
^C
--- 10.0.12.1 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 2999ms
```

Depurando falhas de conectividade

Para encontrar a falha na conectividade entre os hosts x1 e y1, podemos fazer testes de conexão iterativos entre cada um dos hops no caminho esperado pelo pacote.

```
mininet> x1 ping r1
PING 10.0.4.21 (10.0.4.21) 56(84) bytes of data.
64 bytes from 10.0.4.21: icmp_seq=1 ttl=64 time=0.031 ms
64 bytes from 10.0.4.21: icmp_seq=2 ttl=64 time=0.046 ms
^C
--- 10.0.4.21 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.031/0.038/0.046/0.009 ms
mininet> x1 ping r2
PING 10.0.4.22 (10.0.4.22) 56(84) bytes of data.
64 bytes from 10.0.4.22: icmp_seq=1 ttl=63 time=0.039 ms
64 bytes from 10.0.4.22: icmp_seq=2 ttl=63 time=0.038 ms
64 bytes from 10.0.4.22: icmp_seq=3 ttl=63 time=0.038 ms
--- 10.0.4.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.038/0.038/0.039/0.005 ms
mininet> x1 ping r3
PING 10.0.6.23 (10.0.6.23) 56(84) bytes of data.
64 bytes from 10.0.6.23: icmp_seg=1 ttl=62 time=0.070 ms
64 bytes from 10.0.6.23: icmp_seq=2 ttl=62 time=0.043 ms
64 bytes from 10.0.6.23: icmp_seq=3 ttl=62 time=0.050 ms
64 bytes from 10.0.6.23: icmp_seq=4 ttl=62 time=0.064 ms
--- 10.0.6.23 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.043/0.056/0.070/0.014 ms
mininet> x1 ping r4
PING 10.0.1.24 (10.0.1.24) 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
From 10.0.2.21 icmp_seq=4 Destination Net Unreachable
--- 10.0.1.24 ping statistics ---
5 packets transmitted, 0 received, +4 errors, 100% packet loss, time 4006ms
```

A falha de conexão foi isolada entre os hops r3 e r4. Inspecionando o roteador r3 vemos que faltou a configuração do protocolo OSPF na subrede 10.0.6.0/23.

```
ospfd-r3# configure terminal
ospfd-r3(config)# network 10.0.8.23/23 area 0
% [OSPF] Unknown command: network 10.0.8.23/23 area 0
ospfd-r3(config)# router ospf
ospfd-r3(config-router)# network 10.0.8.23/23 area 0
ospfd-r3(config-router)# network 10.0.8.23/23 area 0
ospfd-r3(config-router)# network 10.0.6.23/23 area 0
There is already same network statement.
ospfd-r3(config-router)# end
```

Verificando novamente, agora observamos conectividade entre x1 e y1. Podemos observar o caminho do pacote percorrendo o caminho esperado, sendo ele:

O TTL observado é de 60, representando um decréscimo de 4 pelos hops percorridos.

```
mininet> x1 ping 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=0.055 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=60 time=0.069 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=60 time=0.062 ms
--- 10.0.12.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.055/0.062/0.069/0.005 ms
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]
                                                      pmtu 1500
 1: 10.0.2.21
                                                            0.071ms
 1: 10.0.2.21
                                                            0.010ms
 2: 10.0.4.22
                                                            0.025ms
 3: 10.0.6.23
                                                            0.022ms
 4: 10.0.8.24
                                                            0.021ms
 5: 10.0.12.1
                                                            0.019ms reached
      Resume: pmtu 1500 hops 5 back 5
```

Configurando R5 e obtendo uma nova rota

Até agora configuramos o caminho mais longo entre os hosts x1 e y1, mas de acordo com a topologia também existe outro caminho viável entre os dois hosts através de r5, sendo ele:

Esperamos que o protocolo OSPF atribua este como o novo caminho uma vez configurado. Podemos observar que este é o comportamento obtido através do comando tracepath. O TTL que antes era de 60 agora exibe 61.

```
directly attached to r5-
[20] area: 0.0.0.0
via 10.0.10.21, r5-eth1
[20] area: 0.0.0.0
via 10.0.10.21, r5-eth1
[30] area: 0.0.0.0
via 10.0.1.24, r5-eth2
via 10.0.1.21, r5-eth1
[20] area: 0.0.0.0
via 10.0.1.21, r5-eth1
       10.0.2.0/23
       10,0,4,0/23
       10,0,6,0/23
       10.0.8.0/23
                                          via 10.0.1.24, r5-eth2
[10] area: 0.0.0.0
directly attached to r5-eth1
       10,0,10,0/23
                                         [20] area: 0.0.0.0
via 10.0.1.24, r5-eth2
       10,0,12,0/23
  ======== OSPF router routing table =========
       ====== OSPF external routing table ========
ospfd-r5# sh ip ospf neighbot
% [OSPF] Unknown command: sh ip ospf neighbot
ospfd-r5# sh ip ospf neighbor
      RXmtL RqstL DBsmL
                                                       Dead Time Address
32,390s 10,0,1,24
                                                                                               Interface
r5-eth2:10.0.1.25
                                                           38,286s 10,0,10,21
                                                                                                r5-eth1:10.0.10.25
```

```
mininet> x1 ping 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=2.14 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=61 time=0.400 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=61 time=0.061 ms
^ C
--- 10.0.12.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.061/0.868/2.145/0.913 ms
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]
                                                       pmtu 1500
 1: 10.0.2.21
                                                             0.037ms
 1:
    10.0.2.21
                                                             0.007ms
 2:
    10.0.10.25
                                                             0.015ms
     10.0.1.24
                                                             2.949ms
 3:
 4:
    10.0.12.1
                                                             6.073ms reached
      Resume: pmtu 1500 hops 4 back 4
```

Alterando custos de conexões

Até este ponto todas as interfaces foram configuradas com custos constantes, de maneira que não eram feitas distinções entre um hop e outro. No entanto, outros modelos de configuração de custo podem ser adotados. Iremos configurar

manualmente o custo de algumas interfaces. A princípio, a interface eth1 do roteador r1:

```
ospfd-r5> enable
ospfd-r5# configure termina;
% [OSPF] Unknown command: configure termina;
ospfd-r5# configure terminal
ospfd-r5(config)# interface r5-eth1
ospfd-r5(config-if)# ospf cost 100
ospfd-r5(config-if)# end
ospfd-r5# 
ospfd-r5#
```

Testando a conectividade entre x1 e y1 e vice-versa, vemos que o caminho entre x1 e y1 permanece o mesmo enquanto o caminho inverso, entre y1 e x1 foi alterado para o caminho inicial de 4 hops.

O comportamento não é dual entre as duas chamadas pois ao alterar o custo de uma interface em um roteador estamos alterando apenas a sua percepção de saída do hop, enquanto que o custo de entrada é definido pelo seu vizinho.

```
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_seg=1 ttl=60 time=0.578 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=60 time=0.454 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=60 time=0.107 ms
64 bytes from 10.0.12.1: icmp_seq=4 ttl=60 time=0.079 ms
64 bytes from 10.0.12.1: icmp_seq=5 ttl=60 time=0.069 ms
64 bytes from 10.0.12.1: icmp_seq=6 ttl=60 time=0.089 ms
64 bytes from 10.0.12.1: icmp_seq=7 ttl=60 time=0.072 ms
64 bytes from 10.0.12.1: icmp_seq=8 ttl=60 time=0.085 ms
64 bytes from 10.0.12.1: icmp_seq=9 ttl=60 time=0.066 ms
64 bytes from 10.0.12.1: icmp_seq=10 ttl=60 time=0.096 ms
--- 10.0.12.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9026ms
rtt min/avg/max/mdev = 0.066/0.169/0.578/0.176 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=1.29 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=61 time=0.298 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=61 time=0.104 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=61 time=0.102 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=61 time=0.074 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=61 time=0.066 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=61 time=0.131 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=61 time=0.083 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=61 time=0.061 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=61 time=0.066 ms
```

```
--- 10.0.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 0.061/0.227/1.294/0.362 ms
mininet> y1 tracepath -n x1
 1?: [LOCALHOST]
                                                       pmtu 1500
 1:
    10.0.12.24
                                                              0.034 \text{ms}
    10.0.12.24
 1:
                                                              0.006ms
 2: 10.0.8.23
                                                              0.017ms
 3:
    10.0.6.22
                                                              0.016ms
 4: 10.0.10.21
                                                              1.165ms asymm 3
 5:
    10.0.2.1
                                                              1.011ms reached
      Resume: pmtu 1500 hops 5 back 4
mininet> x1 tracepath -n y1
 1?: [LOCALHOST]
                                                       pmtu 1500
 1: 10.0.2.21
                                                              0.033ms
 1:
    10.0.2.21
                                                              0.006ms
 2:
    10.0.1.25
                                                              1.433ms asymm 5
 3:
    10.0.8.24
                                                              1.369ms asymm 4
                                                              1.508ms reached
 4:
    10.0.12.1
      Resume: pmtu 1500 hops 4 back 5
mininet>
```

Podemos alterar também a interface eth2 do roteador r1 para também um custo de 100. Neste caso, vemos que ambas as chamadas passam a utilizar o caminho por r2 e r3.

```
ospfd-r5> enable
ospfd-r5# configure terminal
ospfd-r5(config)# interface r5-eth2
ospfd-r5(config-if)# ospf cost 100
ospfd-r5(config-if)# end
ospfd-r5# 

| Ospfd-r5# | Ospf cost 100 | Ospfd-r5# | Os
```

```
ininet> 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=0.055 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=60 time=0.100 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=60 time=0.056 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=60 time=0.054 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=60 time=0.078 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=60 time=0.075 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=60 time=0.044 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=60 time=0.144 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=60 time=0.091 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=60 time=0.086 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
65 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
66 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
67 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
68 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
69 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
60 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
60 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
61 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
62 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
63 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
64 bytes from 10.0.2.1: icmp_seq=10 ttl=60 time=0.088 ms
```

```
mininet> y1 tracepath -n x1
 1?: [LOCALHOST]
                                                      pmtu 1500
 1: 10.0.12.24
                                                            0.043ms
    10.0.12.24
                                                            0.007ms
 1:
 2: 10.0.8.23
                                                            0.020ms
 3: 10.0.6.22
                                                            0.015ms
 4: 10.0.4.21
                                                            0.016ms
 5: 10.0.2.1
                                                            0.016ms reached
      Resume: pmtu 1500 hops 5 back 5
```

Interrompendo conexões

Outra característica do protocolo OSPF é reagir rapidamente em estabelecer uma novo caminho entre hosts uma vez que o caminho mais curto é interrompido em uma rede com redundância. Podemos verificar este comportamento usando o comando tracepath e derrubando um dos links da topologia. Neste caso, voltamos ao caminho através de r5 cujo custo foi definido como alto na seção anterior.

```
mininet> x1 tracepath y1
 1?: [LOCALHOST]
                                                      pmtu 1500
 1: 10.0.2.21
                                                            0.042ms
 1:
    10.0.2.21
                                                            0.016ms
 2: 10.0.4.22
                                                            0.016ms
 3: 10.0.6.23
                                                            0.019ms
 4: 10.0.8.24
                                                            0.020ms
 5: 10.0.12.1
                                                            0.141ms reached
      Resume: pmtu 1500 hops 5 back 5
mininet> link r2 r3 down
mininet> x1 tracepath y1
 1?: [LOCALHOST]
                                                      pmtu 1500
 1: 10.0.2.21
                                                            0.036ms
 1:
    10.0.2.21
                                                            0.014ms
 2: 10.0.10.25
                                                            0.020ms
 3: 10.0.1.24
                                                            2.642ms
 4: 10.0.12.1
                                                            2.675ms reached
      Resume: pmtu 1500 hops 4 back 4
```

Acessando a tabela de roteamento de r1 vemos que o custo é o esperado pelo caminho através de r5, como definido na seção anterior.

```
root@wifi-VirtualBox:~/ea080# route -n
Kernel IP routing table
                                                            Flags Metric Ref
UG 110 0
Destination
                                        Genmask
                                                                                     Use Iface
                    Gateway
10.0.0.0
10.0.2.0
10.0.4.0
10.0.8.0
                    10.0.10.25
0.0.0.0
0.0.0.0
10.0.10.25
                                        255,255,254,0
255,255,254,0
255,255,254,0
255,255,254,0
                                                                                        0 r1-eth3
                                                            U
                                                                    0
                                                                             0
                                                                                        0 r1-eth4
                                                            U
                                                                    0
                                                                             0
                                                                                        0 r1-eth1
                                                            UG
                                                                    120
                                                                                        0 r1-eth3
                                                                             0
                                                                                        0 r1-eth3
10,0,10,0
                                        255,255,254,0
                    0.0.0.0
                                                            Ш
                                                                    0
                                                                             0
                                        255,255,254,0
                                                                                        0 r1-eth3
10.0.12.0
                    10.0.10.25
                                                            UG
                                                                    120
                                                                             0
root@wifi-VirtualBox:~/ea080# telnet localhost ospfd
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Hello, this is Quagga (version 0.99.24.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
User Access Verification
directly attached to r1-eth4
N
      10.0.4.0/23
                                  [10] area: 0.0.0.0
                                  directly attached to r1-eth1 [120] area: 0.0.0.0 via 10.0.10.25, r1-eth3 [10] area: 0.0.0.0
N
      10,0,8,0/23
      10.0.10.0/23
                                  directly attached to r1-eth3 [120] area: 0.0.0.0 via 10.0.10.25, r1-eth3
      10,0,12,0/23
 ======== OSPF external routing table ========
ospfd-r1>
```

Inspecionando pacotes

Alteração de custo - LS Update

10 40.966543269	10.0.1.25	224.0.0.5	0SPF	82 Hello Packet
11 50.004132512	10.0.1.24	224.0.0.5	0SPF	82 Hello Packet
12 50.967113531	10.0.1.25	224.0.0.5	OSPF	82 Hello Packet
13 55.051419662	10.0.1.24	224.0.0.5	0SPF	110 LS Update
14 55.051908263	10.0.1.25	224.0.0.5	OSPF	110 LS Update
15 55.725429790	10.0.1.25	224.0.0.5	0SPF	78 LS Acknowledge
16 60.005520922	10.0.1.24	224.0.0.5	0SPF	82 Hello Packet
17 60.967160238	10.0.1.25	224.0.0.5	0SPF	82 Hello Packet

```
Frame 14: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface 0
Ethernet II, Src: 56:bb:22:43:9d:da (56:bb:22:43:9d:da), Dst: IPv4mcast_05 (01:00:5e:00:00:05)
Internet Protocol Version 4, Src: 10.0.1.25, Dst: 224.0.0.5
 Open Shortest Path First

▼ OSPF Header

           Version: 2
           Message Type: LS Update (4)
Packet Length: 76
Source OSPF Router: 10.0.10.25
           Area ID: 0.0.0.0 (Backbone)
Checksum: 0xb7c6 [correct]
           Auth Type: Null (0)
Auth Data (none): 00000000000000000
  ▼ LS Update Packet
Number of LSAs: 1
      Number of LSAS: 1

➤ LSA-type 1 (Router-LSA), len 48

.000 0000 0000 0001 = LS Age (seconds): 1

0...... = Do Not Age Flag: 0

➤ Options: 0x02 ((E) External Routing)

LS Type: Router-LSA (1)

Link State ID: 10.0.10.25
                 Advertising Router: 10.0.10.25
                 Sequence Number: 0x80000007
Checksum: 0x4494
                 Length: 48
            ▶ Flags: 0x00
            Number of Links: 2

Type: Transit ID: 10.0.1.24 Data: 10.0.1.25
Link ID: 10.0.1.24 - IP address of Designated Router
Link Data: 10.0.1.25
                                                                                                                           Metric: 10
                      Link Type: 2 - Connection to a transit network
Number of Metrics: 0 - TOS
                      0 Metric: 10
            ▼ Type: Transit ID: 10.0.10.21 Data: 10.0.10.25
Link ID: 10.0.10.21 - IP address of Designated Router
                                                                                                                           Metric: 100
                      Link Data: 10.0.10.25
                      Link Type: 2 - Connection to a transit network
                      Number of Metrics: 0 - TOS
                      0 Metric
```

Desconexão - LS Update

	211 831.189141288	10.0.1.25	224.0.0.5	0SPF	82 Hello Packet
	212 839.639963083	10.0.1.24	224.0.0.5	OSPF	98 LS Update
	213 839.640792661	10.0.1.25	224.0.0.5	OSPF	130 LS Update
	214 840.233976899	10.0.1.24	224.0.0.5	0SPF	82 Hello Packet
	215 840.482954955	10.0.1.24	224.0.0.5	OSPF	98 LS Acknowledge
	216 840.484045482	10.0.1.25	224.0.0.5	0SPF	78 LS Acknowledge
1	217 841.190129438	10.0.1.25	224.0.0.5	0SPF	82 Hello Packet
	218 847.438028002	10.0.2.1	10.0.12.1	UDP	1514 59671 → 44447 Len=1472
	219 847.438047274	10.0.1.24	10.0.2.1	ICMP	590 Time-to-live exceeded (Time to live exceeded in tr
	220 847.441626494	10.0.2.1	10.0.12.1	DCERPC	1514 Request: seq: 0 opnum: 0 len: 0 NULL V0
	221 847.441655778	10.0.12.1	10.0.2.1	ICMP	590 Destination unreachable (Port unreachable)
	222 847.559202996	10.0.1.24	224.0.0.5	0SPF	98 LS Update
	223 848.503959664	10.0.1.25	224.0.0.5	0SPF	78 LS Acknowledge
	224 849.267329352	10.0.1.25	224.0.0.5	OSPF	98 LS Update
	225 849.503841711	10.0.1.24	224.0.0.5	0SPF	78 LS Acknowledge
	226 850.234063176	10.0.1.24	224.0.0.5	0SPF	82 Hello Packet
	227 851.193472692	10.0.1.25	224.0.0.5	0 PF	82 Hello Packet
	228 852.448474522	9e:60:58:e5:43:ce	56:bb:22:43:9d:da	ARP	42 Who has 10.0.1.25? Tell 10.0.1.24
	229 852.449624577	56:bb:22:43:9d:da	9e:60:58:e5:43:ce	ARP	42 Who has 10.0.1.24? Tell 10.0.1.25
	230 852.449635125	9e:60:58:e5:43:ce	56:bb:22:43:9d:da	ARP	42 10.0.1.24 is at 9e:60:58:e5:43:ce
		56:bb:22:43:9d:da	9e:60:58:e5:43:ce	ARP	42 10.0.1.25 is at 56:bb:22:43:9d:da
	219 847.438047274 220 847.441626494 221 847.441655778 222 847.559202996 223 848.503959664 224 849.267329352 225 849.503841711 226 859.234063176 227 851.193472692 228 852.448474522 229 852.449635125	10.0.1.24 10.0.2.1 10.0.12.1 10.0.1.24 10.0.1.25 10.0.1.25 10.0.1.25 10.0.1.24 10.0.1.24 10.0.1.25 9e:60:58:e5:43:ce 56:bb:22:43:9d:da 9e:60:58:e5:43:ce	10.0.2.1 10.0.2.1 10.0.2.1 224.0.0.5 224.0.0.5 224.0.0.5 224.0.0.5 224.0.0.5 224.0.0.5 224.0.0.5 6:bb:22:43:9d:da 9e:60:58:e5:43:ce 56:bb:22:43:9d:da	ICMP DCERPC ICMP OSPF OSPF OSPF OSPF OSPF OSPF ARP ARP	590 Time-to-live exceeded (Time to live exceeded in tr 1514 Request: seq: 0 opnum: 0 len: 0 NULL V0 590 Destination unreachable (Port unreachable) 98 LS Update 78 LS Acknowledge 98 LS Update 78 LS Acknowledge 82 Hello Packet 82 Hello Packet 42 Who has 10.01.257 Tell 10.01.24 42 Who has 10.01.247 Tell 10.01.25 42 10.01.24 is at 9e:60:58:e5:43:ce

```
Frame 224: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0 Ethernet II, Src: 56:bb:22:43:9d:da (56:bb:22:43:9d:da), Dst: IPv4mcast_05 (01:00:5e:00:00:05) Internet Protocol Version 4, Src: 10.0.1.25, Dst: 224.0.0.5
▼ Open Shortest Path First
    Version: 2
            Message Type: LS Update (4)
            Packet Length: 64
            Source OSPF Router: 10.0.10.25
            Area ID: 0.0.0.0 (Backbone)
Checksum: 0x5827 [correct]
            Auth Type: Null (0)
Auth Data (none): 000000000000000000
    ▼ LS Update Packet
            Number of LSAs: 1
         Value of LSAS. 1

LSA-type 1 (Router-LSA), len 36
.000 0000 0000 1100 = LS Age (seconds): 12
0...... = Do Not Age Flag: 0

Doptions: 0x02 ((E) External Routing)
LS Type: Router-LSA (1)
                 Link State ID: 10.0.6.22
                 Advertising Router: 10.0.6.22
                 Sequence Number: 0x8000000b
                 Checksum: 0xd0da
             Length: 36
Flags: 0x00
Number of Links: 1
Type: Transit ID: 10.0.4.22
                                                                          Data: 10.0.4.22
                                                                                                              Metric: 10
                      Link ID: 10.0.4.22 - IP address of Designated Router
                     Link Data: 10.0.4.22
Link Type: 2 - Connection to a transit network
Number of Metrics: 0 - TOS
                      0 Metric: 10
```

https://tools.ietf.org/html/rfc2328