Trabalho Prático 1 - Encaminhamento de Tráfego

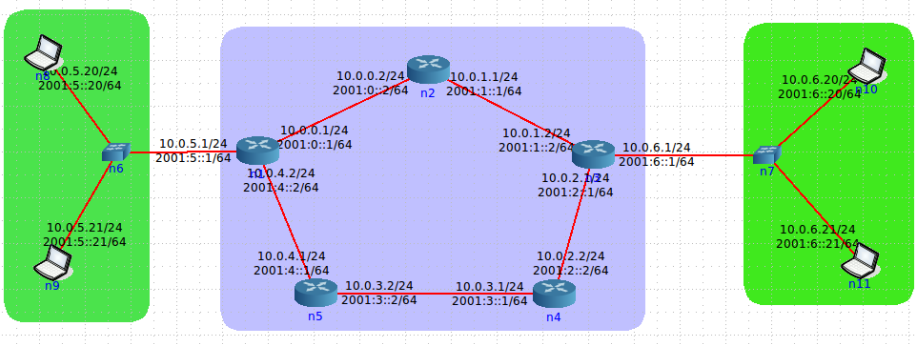
Grupo 2
Bruno Dantas, Gonçalo Pereira, José Silva
a74027, a74413, a75280

17 de outubro de 2017

1 Protocolo RIP (*Routing Information Protocol*)

1.1 Questão 1

A partir do enunciado, optamos por construir uma topologia similar à topologia de exemplo para se manter coerente com o resto que era pedido.



1.2 Questão 2

Através da observação da topologia, conseguimos observar que temos cinco *routers* que entre eles geram cinco subredes:

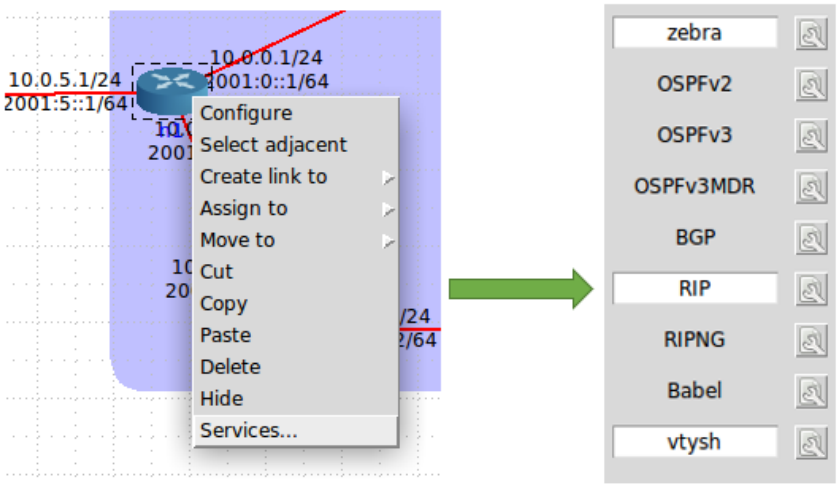
- Subrede 1 - *routers* n1 e n2: 10.0.0.0/24
- Subrede 2 - *routers* n2 e n3: 10.0.1.0/24
- Subrede 3 - *routers* n3 e n4: 10.0.2.0/24
- Subrede 4 - *routers* n4 e n5: 10.0.3.0/24
- Subrede 5 - *routers* n5 e n1: 10.0.4.0/24

Além das subredes entre *routers*, podemos observar também, as duas subredes que ligam os *hosts* aos *routers* n1 e n3 através de um *switch*:

- Subrede 6 - *router* n1: 10.0.5.0/24
- Subrede 7 - *router* n3: 10.0.6.0/24

1.3 Questão 3

De modo a todos os *routers* utilizarem o protocolo RIP, aplicámos as seguintes alterações (em relação à configuração inicial) em todos os *routers*:



1.4 Questão 4

Para esta alínea fizemos alguns testes de *ping* modo a verificar conetividade entre todos os equipamentos.

```
root@n1:/tmp/pycore.54416/n1.conf# ping 10.0.6.20
PING 10.0.6.20 (10.0.6.20) 56(84) bytes of data.
64 bytes from 10.0.6.20: icmp_req=1 ttl=62 time=0.063 ms
64 bytes from 10.0.6.20: icmp_req=2 ttl=62 time=0.065 ms
64 bytes from 10.0.6.20: icmp_req=3 ttl=62 time=0.044 ms
64 bytes from 10.0.6.20: icmp_req=4 ttl=62 time=0.068 ms
64 bytes from 10.0.6.20: icmp_req=5 ttl=62 time=0.090 ms
64 bytes from 10.0.6.20: icmp_req=6 ttl=62 time=0.061 ms
64 bytes from 10.0.6.20: icmp_req=7 ttl=62 time=0.056 ms
64 bytes from 10.0.6.20: icmp_req=8 ttl=62 time=0.065 ms
64 bytes from 10.0.6.20: icmp_req=9 ttl=62 time=0.060 ms
```

Figure 1: Teste de *ping* do *router* n1 para o *host* n10

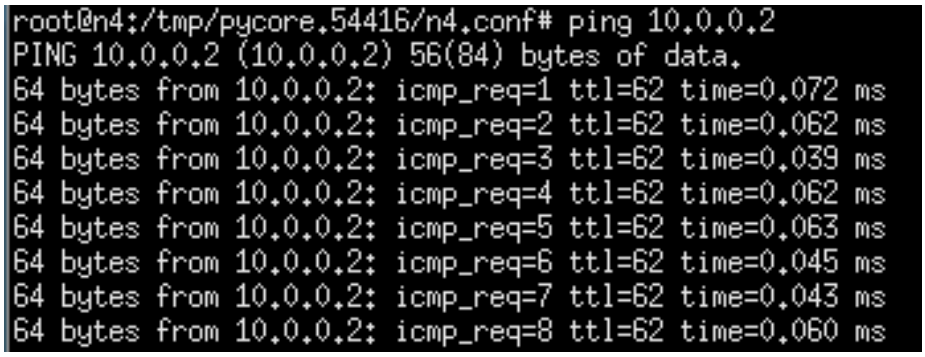


Figure 2: Teste de *ping* do *router* n4 para o *router* n2

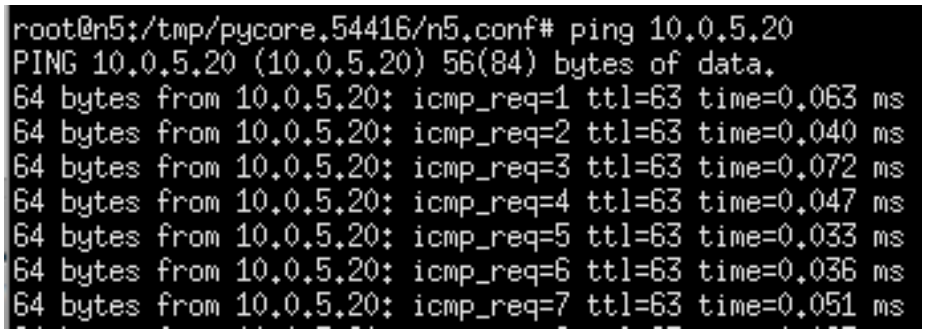


Figure 3: Teste de *ping* do *router* n5 para o *host* n10

1.5 Questão 5)

1.5.1 i)

Para o *router* n1, a tabela de *routing* contém as rotas para todas as subredes.

Relativo aos *gateways* e *flags*, existe um padrão visível entre as *gateways* que são diferentes de `0.0.0.0` e a existência da *flag* `'G'`. Tal se deve ao facto de a rota passar, de facto, por um gateway, e não pelo default `0.0.0.0`.

No que toca à interface, é possível verificar que existem três interfaces diferentes, sendo que os mais utilizados são o `eth0` e o `eth1`. Estas interfaces são as centrais da topologia, pelas quais passa a grande maioria das ligações.

Nas restantes linhas, o *Gateway* mostra qual é o próximo salto necessário para atingir a subrede especificada em *Destination*, assim como a interface de saída.

Outros parâmetros são a *Gensmask* (máscara de rede), o *MSS* (tamanho máximo dos segmentos transmitidos ao longo da rota), a *Window* (tamanho de janela) e o *irtt* (tempo inicial de um RTT)

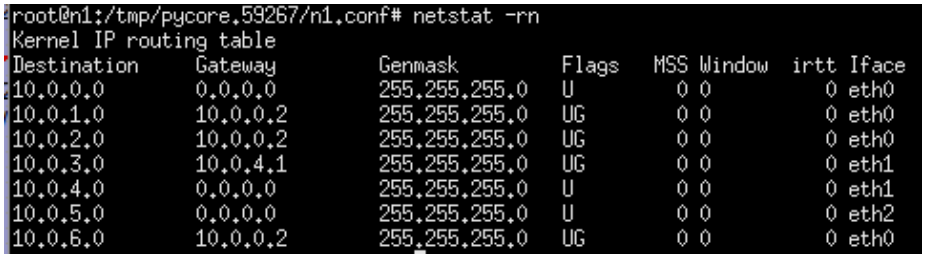


Figure 4: Tabela de *routing* do *router* n1

1.5.2 ii)

Para a tabela de *routing* de um cliente, podemos constatar que, no caso da topologia anteriormente definida, é uma tabela bastante reduzida, sendo que a sua sub-rede é apenas constituída por um outro cliente e todo o tráfego é redirecionado para o *router* mais próximo (identificado por n6). No que toca aos parâmetros, a definição dos mesmos é igual à da alínea anterior.

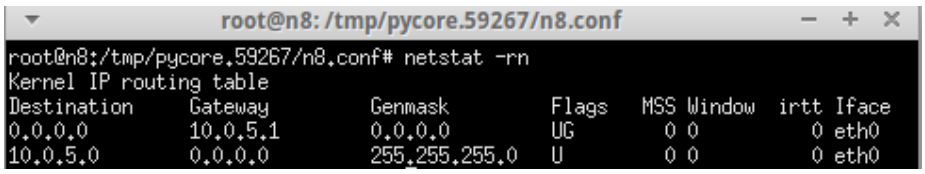


Figure 5: Tabela de *routing* do *host* n8

1.6 Questão 6

De modo a alterar o intervalo de tempo da geração de *updates* o comando a utilizar seria: `timers basic`, onde executamos como exemplo `timers basic 30 40 90 240` sendo que o parâmetro que altera a data é o 30 que indica o tempo de *update* sendo que os seguintes parâmetros representam o tempo que uma rota é considerada inválida, *holddown* intervalo para que as informações acerca do *routing* estabilizem prevenindo que ocorram *loops* durante a convergência da topologia e o último indica o tempo de *flush*, ou seja o intervalo de tempo que indica até quando a rota é eliminada da tabela de *routing*.

1.7 Questão 7)

1.7.1 i)

Nas tabelas a seguir podemos observar a evolução do custo dos links. Na primeira observamos que o custo de chegar à subrede 10.0.5.0 é de 2 e que segue pela interface *eth0*. Na segunda tabela observamos que esse mesmo trajeto agora segue pelo *eth1* e que por sua vez o custo aumenta, uma vez que contorna toda a topologia. Outra coisa que podemos tirar são os tempos de propagação que em certos casos aumentaram de 3 para 30 segundos.

```
n2# sh ip route rip
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, o - OSPF6, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

R>* 10.0.2.0/24 [120/2] via 10.0.1.2, eth1, 00:00:02
R>* 10.0.3.0/24 [120/3] via 10.0.1.2, eth1, 00:00:02
R>* 10.0.4.0/24 [120/2] via 10.0.0.1, eth0, 00:00:03
R>* 10.0.5.0/24 [120/2] via 10.0.0.1, eth0, 00:00:03
R>* 10.0.6.0/24 [120/2] via 10.0.1.2, eth1, 00:00:02
n2# configure terminal
n2(config)# interface eth0
n2(config-if)# shutdown
```

Figure 6: Antes desativação da interface

```
n2# sh ip route rip
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, o - OSPF6, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

R>* 10.0.0.0/24 [120/5] via 10.0.1.2, eth1, 00:00:00
R>* 10.0.2.0/24 [120/2] via 10.0.1.2, eth1, 00:00:30
R>* 10.0.3.0/24 [120/3] via 10.0.1.2, eth1, 00:00:30
R>* 10.0.4.0/24 [120/4] via 10.0.1.2, eth1, 00:00:03
R>* 10.0.5.0/24 [120/5] via 10.0.1.2, eth1, 00:00:03
R>* 10.0.6.0/24 [120/2] via 10.0.1.2, eth1, 00:00:30
```

Figure 7: Após desativação da interface

1.7.2 ii)

Na figura seguinte conseguimos observar que se desligarmos ambas as interfaces do *router* n2, este fica ligado a ele mesmo sem qualquer tipo de comunicação com os outros *routers* e estes sem qualquer tipo de ligação de o poder alcançar.

1.8 Questão 8

Configuramos o *router* n1 de forma a aumentar a métrica em 5 unidades para qualquer pacote que seja destinado à rede 10.0.6.0.

Primeiro criamos uma entrada na *access-list* através do comando `access-list 1 permit 10.0.6.1 0.0.0.255`. Após isto, recorremos ao comando `offset -list 1 in 5 eth0`. O primeiro parâmetro refere-se ao número da *access-list*, para que a métrica não seja aumentada para qualquer pacote (assim só é aumentada para pacotes destinados à rede 10.0.6.0).

O parâmetro *"in"* é utilizado porque queremos que a métrica aumente para tráfego que chegue ao *router* (e que seja destinado à rede). Os últimos dois são a métrica e a interface de saída afetada pela métrica.

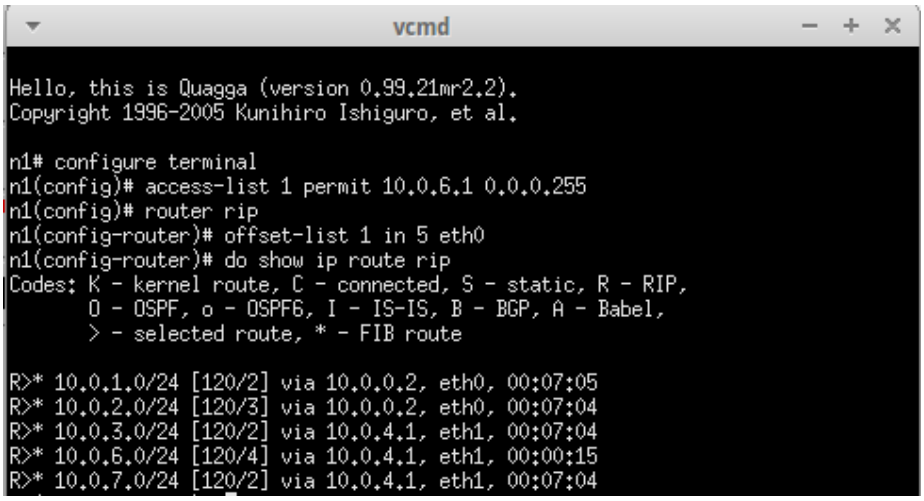


Figure 8: Tabela RIP com as métricas associadas no *router* n1

Através da tabela percebe-se que o caminho utilizado pelo protocolo deixou de ser o que possuía um menor número de saltos (3 saltos, que passaram a ser 7) e passou a ser o outro possível (4 saltos).

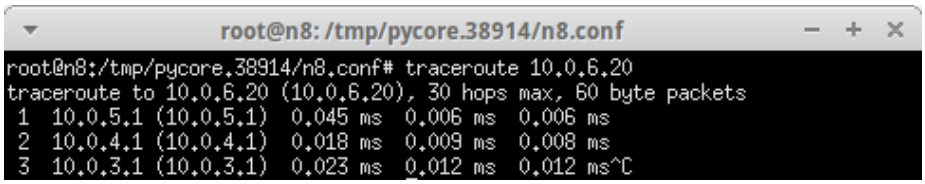


Figure 9: Rota efetuada do *host* n8 para o *host* n10 após alterações

Para testar se o caminho mais curto era efetivamente utilizado recorremos ao comando `traceroute 10.0.6.0` de um host da rede 10.0.5.0 para a outra rede cliente. A tabela comprova que a rota alterou.

1.9 Questão 9

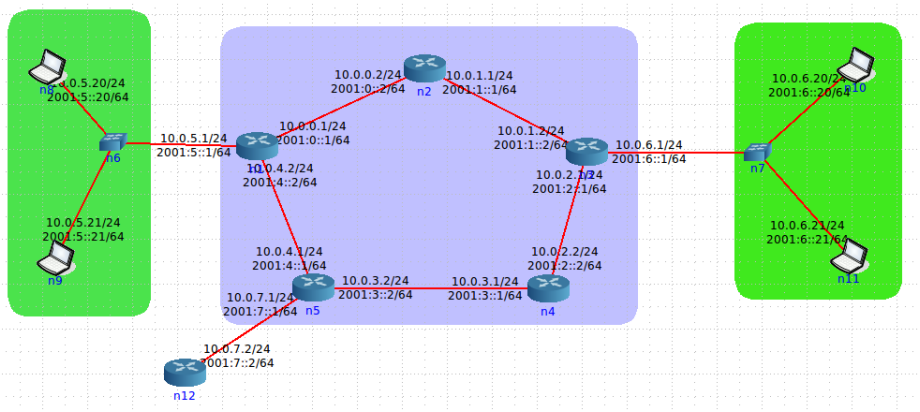


Figure 10: Topologia ilustrativa do objetivo pretendido

Para que todo o tráfego externo saia pelo *router* n5 é necessário realizar uma de duas possibilidades. 1) adicionar manualmente rotas estáticas em cada *router* da rede de interligação; 2) adicionar manualmente uma rota estática no *router* de saída de tráfego e gerar um anúncio para todos os *routers* da rede de interligação.

Optamos pela opção mais prática, a segunda. Para isso, fizemos uso dos comandos `ip route 10.0.7.2 eth2` e `default-information originate` no *router* n5.

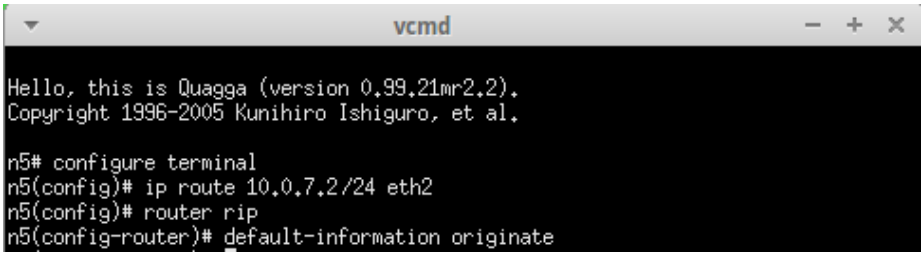
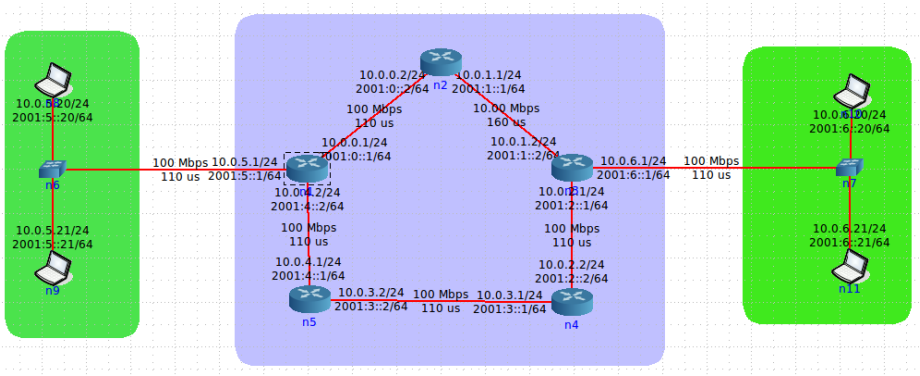


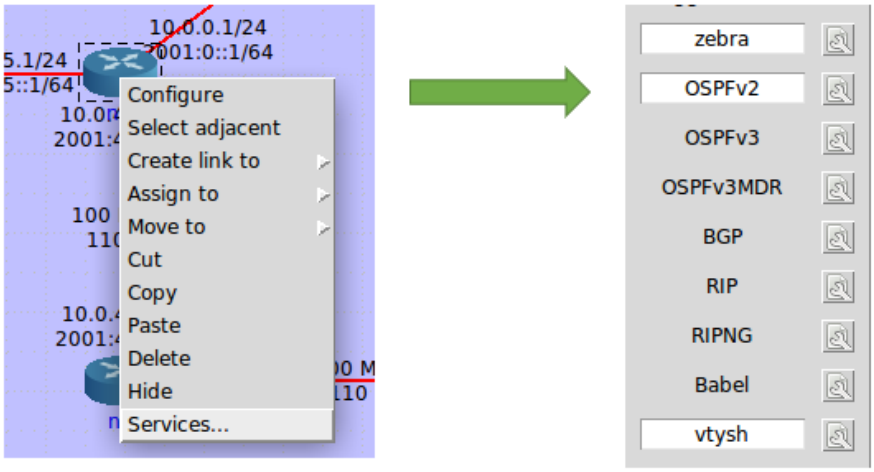
Figure 11: Exemplo dos comandos a utilizar

2 Protocolo OSPF (*Open Shortest Path First*)

2.1 Questão 10



2.2 Questão 11



2.3 Questão 12

Para esta alínea começamos por efetuar alguns testes de *ping* de modo a testar a conetividade entre ligações e equipamentos.

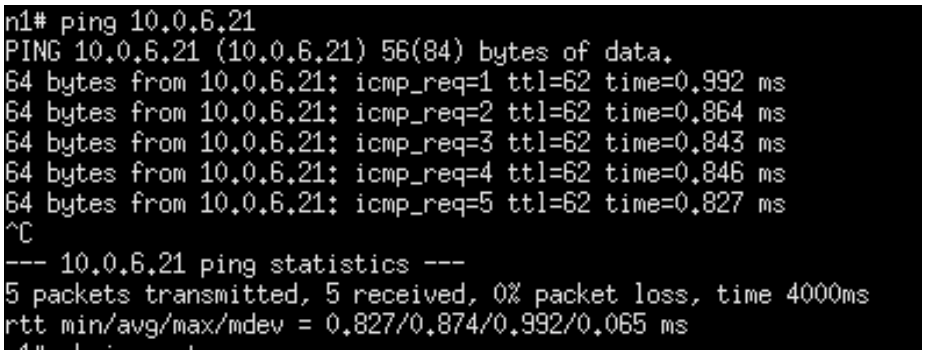


Figure 12: Teste de *ping* do *router* n1 para o *host* n11

Através da utilização do comando `ping 10.0.6.21` no *host* n8, testamos a conetividade de uma rede cliente para outra.

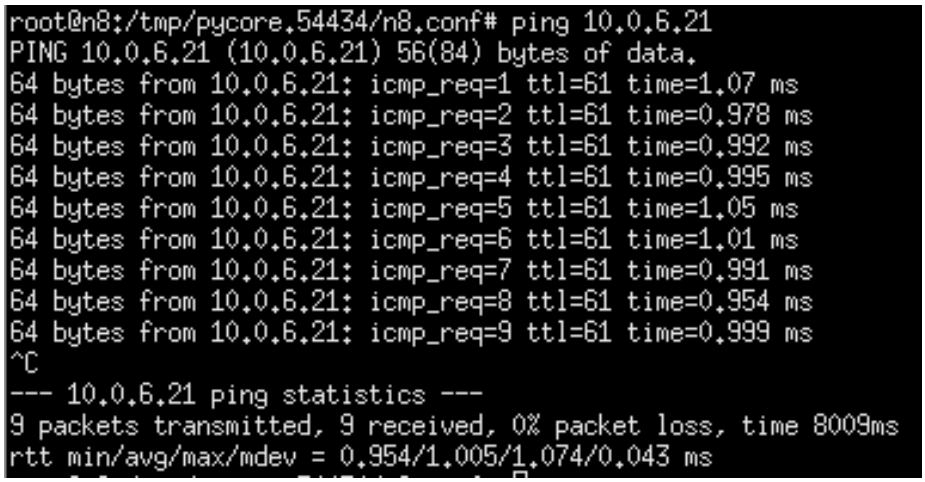


Figure 13: Teste de *ping* do *host* n8 para *host* n11

Com o comando `tracepath 10.0.6.20` verificamos que o caminho mais curto para o protocolo OSPF é o caminho com o menor número de saltos (apesar de o último *link* ter uma largura de banda dez vezes inferior às restantes da rede de interligação), ou seja, o caminho superior que passa pelos *routers* n1, n2 e n3. Logicamente esta é a única rota utilizada de uma rede cliente para a outra, uma vez que só existem duas possibilidades.

Figure 14: Rota efetuada do *host* n8 para o *host* n10

2.4 Questão 13

Primeiro é necessário permitir o protocolo OSPF no *router* em questão, que é feito no início. Logo na próxima linha, vem a identificação do *router* n1.

Em baixo, descrevem-se as redes que queremos que sejam avisadas pelo protocolo, assim como a identificação da área que queremos que façam parte. No nosso caso, só existe uma área na rede de interligação, pelo que a implementação do protocolo torna-se mais simples.

A tabela seguinte segue o mesmo raciocínio da primeira, que configurou o protocolo nas redes que o *router* cobre. A diferença é que esta assegura a implementação do protocolo nas interfaces do *router*.

```
router ospf
  router-id 10.0.0.1
  network 10.0.0.0/24 area 0
  network 10.0.4.0/24 area 0
  network 10.0.5.0/24 area 0
!
router ospf6
  router-id 10.0.0.1
  interface eth0 area 0.0.0.0
  interface eth1 area 0.0.0.0
  interface eth2 area 0.0.0.0
```

Figure 15: Configurações OSPF do *router* n1

2.5 Questão 14

```
root@n1:/tmp/pycore.43703/n1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.1.0        10.0.0.2        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        10.0.0.2        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        10.0.4.1        255.255.255.0   UG        0 0          0 eth1
10.0.4.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.0.5.0        0.0.0.0         255.255.255.0   U         0 0          0 eth2
10.0.6.0        10.0.0.2        255.255.255.0   UG        0 0          0 eth0
```

Figure 16: Tabela de *routing* do *router* n1

Para o *router* n1, a tabela de *routing* contém as rotas para todas as subredes.

Cada uma das três interfaces está diretamente ligada a uma subrede diferente, logo nessas linhas não existe uma *gateway* (fica com o valor 0.0.0.0). A *flag* 'U' indica que a rota está ativada. Ao contrário das restantes linhas, não possuem a *flag* 'G' porque a rota não passa por nenhuma *gateway*.

Para as outras linhas, o *Gateway* mostra qual é o próximo salto necessário para atingir a subrede especificada em *Destination*, assim como a interface de saída.

Outros parâmetros são a *Gensmask* (máscara de rede), o *MSS* (tamanho máximo dos segmentos transmitidos ao longo da rota), a *Window* (tamanho de janela) e o *irtt* (tempo inicial de um RTT).

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.5.1        0.0.0.0         UG        0 0          0 eth0
10.0.5.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

Figure 17: Tabela de *routing* do *host* n8

Para a análise de uma tabela diferente, analisamos o *host* n8. A tabela só possui duas entradas. A primeira entrada refere-se à rota *default*, ou seja, caso o pacote a enviar tenha uma subrede destino que a tabela desconheça, essa rota será a utilizada. A segunda corresponde à rota para a própria subrede, como já foi explicado anteriormente.

2.6 Questão 15

O comando que permite verificar os custos atribuídos às várias interfaces pelo protocolo OSPF é `show ip route ospf`.

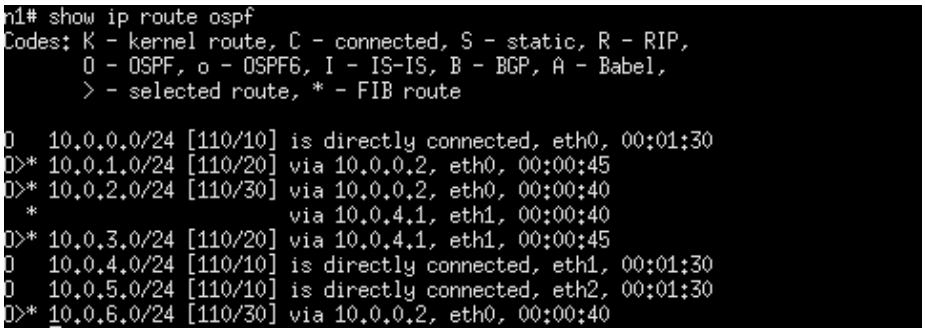


Figure 18: Custos atribuídos ás várias interfaces

Foi atribuído um custo de 10 a todos os *links*. Este custo é predefinido, uma vez que no *router* n1 o protocolo optou pelo caminho com menor largura de banda (quando os dois têm o mesmo número de saltos). Ou seja, para qualquer largura de banda o custo associado ao *link* é sempre 10.

2.7 Questão 16

Para alterar os custos dos links da rede de interligação utilizamos o comando `ip ospf cost 1` sendo que o 1 é o custo do link cuja largura de banda é 100Mbps. Não efetuamos alterações às ligações de 10Mbps, pois o custo de 10 já se encontrava predefinido em todas as rotas.

```
n2# conf terminal
n2(config)# int eth0
n2(config-if)# ip ospf cost 1
n2(config-if)# exit
n2(config)# exit
```

Figure 19: Exemplo para o *router* N2

2.8 Questão 17

Optamos por escolher o *router* n2, pois segundo a nossa topologia é um bom candidato para demonstrar que houve de facto alterações nas rotas. Na figura seguinte podemos observar as rotas que saem pelas duas interfaces presentes no *router* e que se mantém equilibrada a distribuição de rotas.

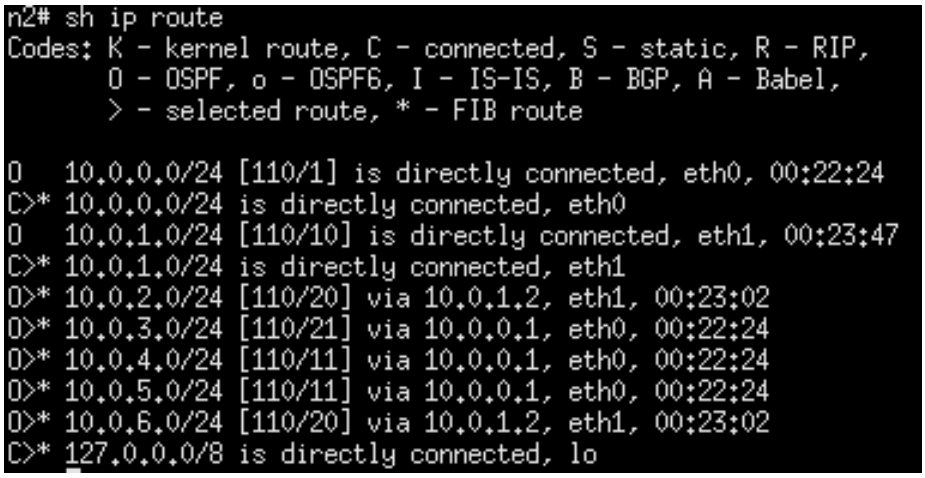


Figure 20: Tabela de rotas antes de executar `ip ospf cost 1`

De seguida após a execução dos comandos, observamos que a maioria das rotas utiliza agora a interface `eth0`, contornando o *link* de custo 10 seguindo pela de menor custo e de maior largura de banda.

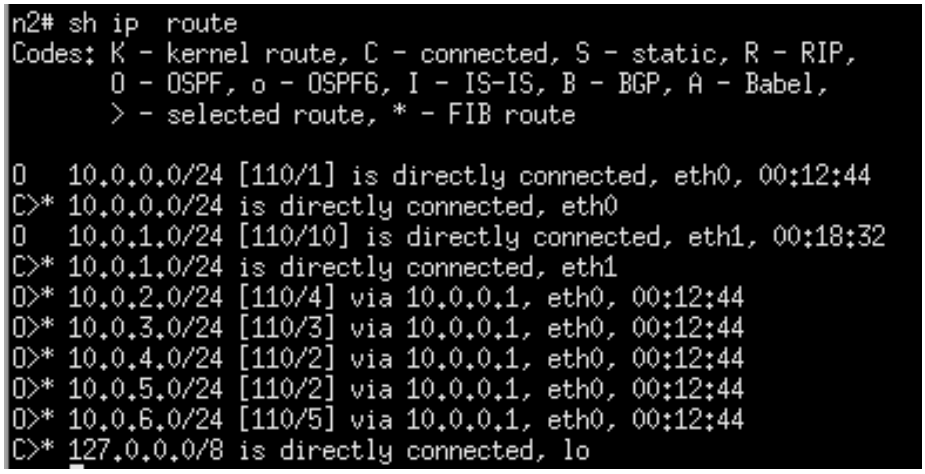


Figure 21: Tabela de rotas após executar `ip ospf cost 1`

2.9 Questão 18

Nos routers que usem o protocolo OSPF, e como a constante positiva e inteira mais baixa é o 1, atribuímos este valor como custo unitário a todos os links que possuam o valor-base tabelado na documentação (que, por sua vez, é o de 100Mbps, ou seja, 100000000).

Deste modo, e tendo já uma referência, podemos afirmar que, pela fórmula `custo link = 100000000 / banda larga`, é possível o cálculo de todos os links consoante a sua capacidade. Assim chegamos então aos valores de 1 e 10 nos casos de 100Mbps e 10Mbps.

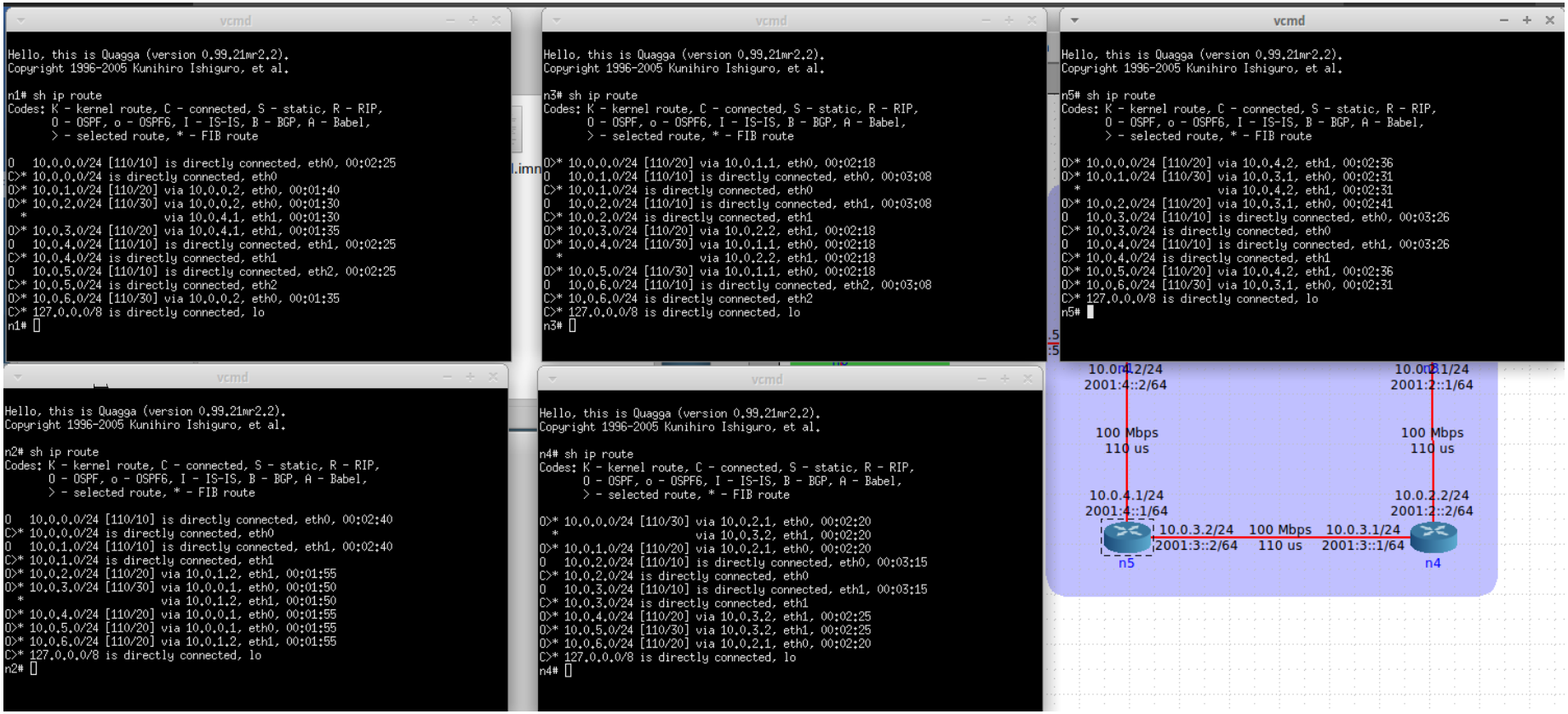
2.10 Questão 19

Caso existam várias rotas com o mesmo custo, o protocolo OSPF encarrega-se de as guardar todas na tabela de routing, possibilitando a escolha de múltiplos caminhos (e, caso uma das rotas falhe, poderá optar por outra sem que tenha de refazer a tabela).

A desvantagem desta abordagem reside no facto de, caso existam vários caminhos, a tabela de routing poderá ficar sobrecarregada, levando a elevados custos de memória (pelo armazenamento de uma tabela com elevado número de registos).

2.11 Questão 20

Na eventualidade de estarem os dois protocolos activos, as rotas escolhidas seriam as do protocolo OSPF, pelo facto de serem as que remetem para o caminho mais curto.


vcmd
Hello, this is Quagga (version 0.99.21mr2.2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
n1# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
O - OSPF, o - OSPF6, I - IS-IS, B - BGP, A - Babel,
> - selected route, * - FIB route
O 10.0.0.0/24 [110/10] is directly connected, eth0, 00:02:25
C>* 10.0.0.0/24 is directly connected, eth0
O>* 10.0.1.0/24 [110/20] via 10.0.0.2, eth0, 00:01:40
O>* 10.0.2.0/24 [110/30] via 10.0.0.2, eth0, 00:01:30
* via 10.0.4.1, eth1, 00:01:30
O>* 10.0.3.0/24 [110/20] via 10.0.4.1, eth1, 00:01:35
O 10.0.4.0/24 [110/10] is directly connected, eth1, 00:02:25
C>* 10.0.4.0/24 is directly connected, eth1
O 10.0.5.0/24 [110/10] is directly connected, eth2, 00:02:25
C>* 10.0.5.0/24 is directly connected, eth2
O>* 10.0.6.0/24 [110/30] via 10.0.0.2, eth0, 00:01:35
C>* 127.0.0.0/8 is directly connected, lo
n1#
vcmd
Hello, this is Quagga (version 0.99.21mr2.2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
n3# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
O - OSPF, o - OSPF6, I - IS-IS, B - BGP, A - Babel,
> - selected route, * - FIB route
O>* 10.0.0.0/24 [110/20] via 10.0.1.1, eth0, 00:02:18
O 10.0.1.0/24 [110/10] is directly connected, eth0, 00:03:08
C>* 10.0.1.0/24 is directly connected, eth0
O 10.0.2.0/24 [110/10] is directly connected, eth1, 00:03:08
C>* 10.0.2.0/24 is directly connected, eth1
O>* 10.0.3.0/24 [110/20] via 10.0.2.2, eth1, 00:02:18
O>* 10.0.4.0/24 [110/30] via 10.0.1.1, eth0, 00:02:18
* via 10.0.2.2, eth1, 00:02:18
O>* 10.0.5.0/24 [110/30] via 10.0.1.1, eth0, 00:02:18
O 10.0.6.0/24 [110/10] is directly connected, eth2, 00:03:08
C>* 10.0.6.0/24 is directly connected, eth2
C>* 127.0.0.0/8 is directly connected, lo
n3#
vcmd
Hello, this is Quagga (version 0.99.21mr2.2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
n5# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
O - OSPF, o - OSPF6, I - IS-IS, B - BGP, A - Babel,
> - selected route, * - FIB route
O>* 10.0.0.0/24 [110/20] via 10.0.4.2, eth1, 00:02:36
O>* 10.0.1.0/24 [110/30] via 10.0.3.1, eth0, 00:02:31
* via 10.0.4.2, eth1, 00:02:31
O>* 10.0.2.0/24 [110/20] via 10.0.3.1, eth0, 00:02:41
O 10.0.3.0/24 [110/10] is directly connected, eth0, 00:03:26
C>* 10.0.3.0/24 is directly connected, eth0
O 10.0.4.0/24 [110/10] is directly connected, eth1, 00:03:26
C>* 10.0.4.0/24 is directly connected, eth1
O>* 10.0.5.0/24 [110/20] via 10.0.4.2, eth1, 00:02:36
O>* 10.0.6.0/24 [110/30] via 10.0.3.1, eth0, 00:02:31
C>* 127.0.0.0/8 is directly connected, lo
n5#
vcmd
Hello, this is Quagga (version 0.99.21mr2.2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
n2# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
O - OSPF, o - OSPF6, I - IS-IS, B - BGP, A - Babel,
> - selected route, * - FIB route
O 10.0.0.0/24 [110/10] is directly connected, eth0, 00:02:40
C>* 10.0.0.0/24 is directly connected, eth0
O 10.0.1.0/24 [110/10] is directly connected, eth1, 00:02:40
C>* 10.0.1.0/24 is directly connected, eth1
O>* 10.0.2.0/24 [110/20] via 10.0.1.2, eth1, 00:01:55
O>* 10.0.3.0/24 [110/30] via 10.0.0.1, eth0, 00:01:50
* via 10.0.1.2, eth1, 00:01:50
O>* 10.0.4.0/24 [110/20] via 10.0.0.1, eth0, 00:01:55
O>* 10.0.5.0/24 [110/20] via 10.0.0.1, eth0, 00:01:55
O>* 10.0.6.0/24 [110/20] via 10.0.1.2, eth1, 00:01:55
C>* 127.0.0.0/8 is directly connected, lo
n2#
vcmd
Hello, this is Quagga (version 0.99.21mr2.2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
n4# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
O - OSPF, o - OSPF6, I - IS-IS, B - BGP, A - Babel,
> - selected route, * - FIB route
O>* 10.0.0.0/24 [110/30] via 10.0.2.1, eth0, 00:02:20
* via 10.0.3.2, eth1, 00:02:20
O>* 10.0.1.0/24 [110/20] via 10.0.2.1, eth0, 00:02:20
O 10.0.2.0/24 [110/10] is directly connected, eth0, 00:03:15
C>* 10.0.2.0/24 is directly connected, eth0
O 10.0.3.0/24 [110/10] is directly connected, eth1, 00:03:15
C>* 10.0.3.0/24 is directly connected, eth1
O>* 10.0.4.0/24 [110/20] via 10.0.3.2, eth1, 00:02:25
O>* 10.0.5.0/24 [110/30] via 10.0.3.2, eth1, 00:02:25
O>* 10.0.6.0/24 [110/20] via 10.0.2.1, eth0, 00:02:20
C>* 127.0.0.0/8 is directly connected, lo
n4#
10.0.4.2/24
2001:4::2/64
10.0.3.1/24
2001:2::1/64
100 Mbps
110 us
100 Mbps
110 us
10.0.4.1/24
2001:4::1/64
10.0.2.2/24
2001:2::2/64
10.0.3.2/24
2001:3::2/64
100 Mbps
110 us
10.0.3.1/24
2001:3::1/64
n5
n4