



Departamento de Informática

Licenciatura em Engenharia Informática

Redes de Computadores

**Universidade do Minho**  
Escola de Engenharia

## **Trabalho Prático n.º 2**

**Grupo n.º 18**

**Artur Carneiro Neto de Nóbrega Luís, n.º A95414**

**Hugo Ricardo Macedo Gomes, n.º A96842**

**Orlando José da Cunha Palmeira, n.º A97755**

**Braga, abril de 2022**

# Índice

<b>1 - Primeira parte</b>	<b>3</b>
1.1 Exercício 1	3
1.1.a Alínea a)	3
1.1.b Alínea b)	4
1.1.c Alínea c)	4
1.1.d Alínea d)	5
1.1.e Alínea e)	5
1.2 Exercício 2	5
1.2.a Alínea a)	5
1.2.b Alínea b)	6
1.2.c Alínea c)	6
1.2.d Alínea d)	7
1.2.e Alínea e)	7
1.2.f Alínea f)	7
1.2.g Alínea g)	7
1.3 Exercício 3	8
1.3.a Alínea a)	8
1.3.b Alínea b)	8
1.3.c Alínea c)	9
1.3.d Alínea d)	9
1.3.e Alínea e)	9
1.3.f Alínea f)	10
1.3.g Alínea g)	10
<b>2 - Segunda parte</b>	<b>11</b>
2.1 Exercício 1	11
2.1.a Alínea a)	11
2.1.b Alínea b)	12
2.1.c Alínea c)	12
2.1.d Alínea d)	12
2.1.e Alínea e)	14
2.1.f Alínea f)	15
2.2 Exercício 2	16
2.2.a Alínea a)	16
2.2.b Alínea b)	17
2.2.c Alínea c)	18
2.2.d Alínea d)	19
2.2.e Alínea e)	20

2.3 Exercício 3	22
2.3.1 Alínea 1)	22
2.3.2 Alínea 2)	24
2.3.3 Alínea 3)	24
<b>3 - Conclusões</b>	<b>27</b>

# 1 - Primeira parte

## 1.1 Exercício 1

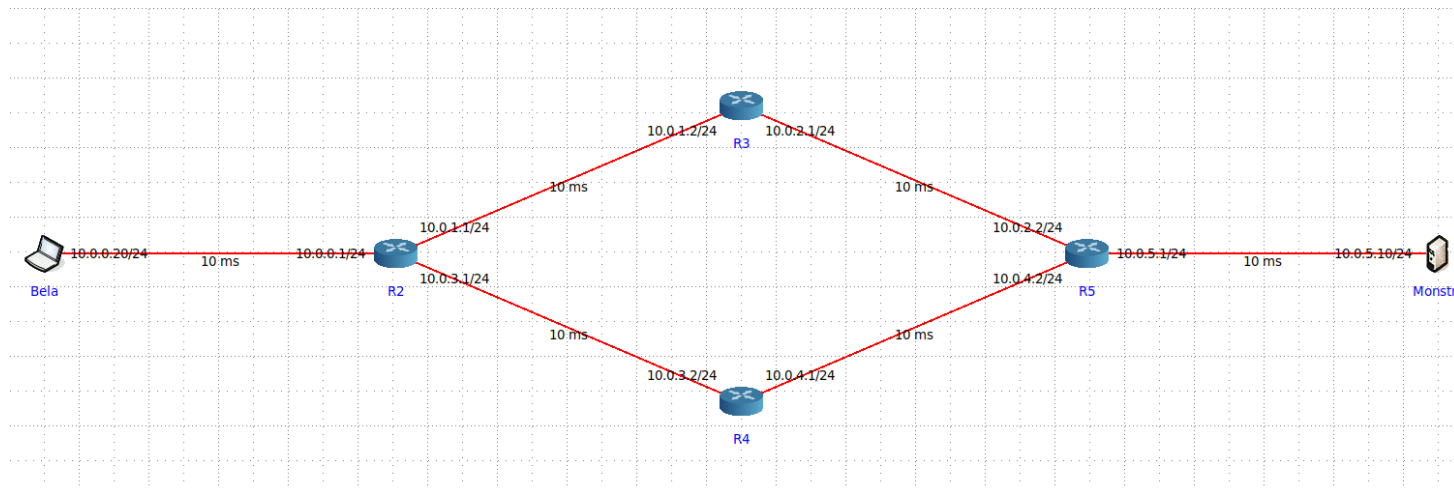


Figura 1 - Topologia Core (Exercício 1)

### 1.1.a Alínea a)

- *Output* comando `traceroute -I 10.0.5.10`

```
root@Bela:/tmp/pycore.43739/Bela.conf# traceroute -I 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  20.700 ms  20.675 ms  20.675 ms
 2  10.0.1.2 (10.0.1.2)  41.592 ms  41.590 ms  41.590 ms
 3  10.0.2.2 (10.0.2.2)  62.082 ms  62.080 ms  62.080 ms
 4  10.0.5.10 (10.0.5.10)  82.825 ms  82.825 ms  82.824 ms
root@Bela:/tmp/pycore.43739/Bela.conf#
```

Figura 2 - Output do traceroute

- Output Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
27	13.041780279	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
9	13.021090864	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=1/256, ttl=1 (no response found!)
10	13.021104903	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=2/512, ttl=1 (no response found!)
11	13.021106525	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=3/768, ttl=1 (no response found!)
12	13.021108207	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=4/1024, ttl=2 (no response found!)
13	13.021109679	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=5/1280, ttl=2 (no response found!)
14	13.021111281	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=6/1536, ttl=2 (no response found!)
15	13.021113304	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=7/1792, ttl=3 (no response found!)
16	13.021114846	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=8/2048, ttl=3 (no response found!)
17	13.021116238	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=9/2304, ttl=3 (no response found!)
18	13.021117960	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=10/2560, ttl=4 (reply in 43)
19	13.021119853	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=11/2816, ttl=4 (reply in 44)
20	13.021121285	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=12/3072, ttl=4 (reply in 45)
21	13.021122837	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=13/3328, ttl=5 (reply in 46)
22	13.021124159	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=14/3584, ttl=5 (reply in 47)
23	13.021125450	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=15/3840, ttl=5 (reply in 48)
24	13.021127013	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=16/4096, ttl=6 (reply in 49)
28	13.042515091	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=17/4352, ttl=6 (reply in 50)
29	13.042523812	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=18/4608, ttl=6 (reply in 51)
30	13.042528308	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=19/4864, ttl=7 (reply in 52)
34	13.062982676	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=20/5120, ttl=7 (reply in 53)
35	13.062987523	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=21/5376, ttl=7 (reply in 54)
36	13.062989626	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=22/5632, ttl=8 (reply in 55)
40	13.083410397	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=23/5888, ttl=8 (reply in 56)
41	13.083414072	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=24/6144, ttl=8 (reply in 57)
42	13.083416285	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001c, seq=25/6400, ttl=9 (reply in 58)
31	13.062692595	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

Frame 18: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface veth1.0.71, id 0  
 Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:01 (00:00:00:aa:00:01)  
 Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.5.10  
   0100 .... = Version: 4  
   .... 0101 = Header Length: 20 bytes (5)  
   Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
   Total Length: 60  
   Identification: 0x8aba (35514)  
   Flags: 0x0000  
   Fragment offset: 0  
   Time to live: 4  
   Protocol: ICMP (1)  
   Header checksum: 0x12ea [validation disabled]  
   [Header checksum status: Unverified]  
   Source: 10.0.0.20  
   Destination: 10.0.5.10  
 Internet Control Message Protocol

```

0000  00 00 00 aa 00 01 00 00 00 aa 00 00 08 00 45 00  ....E.
0010  00 3c 8a ba 00 00 04 01 12 ea 0a 00 00 14 0a 00  -<.....
0020  05 0a 08 00 82 54 00 1c 00 0a 48 49 4a 4b 4c 4d  ....T...HIJKLM
  
```

Figura 3 - Output Wireshark (com relevo para o primeiro pacote que chegou ao destino sem erros)

### 1.1.b Alínea b)

O comportamento esperado dos saltos do tráfego entre o pc *Bela* e o servidor *Monstro* é dado pelo resultado do *traceroute* evidenciado na Figura 2. Como se pode verificar, o TTL esperado será de 4.

Ao analisar os resultados do *Wireshark*, verificámos que, dos pacotes enviados pelo pc *Bela*, o primeiro que chegou ao destino sem retornar erros foi o pacote com ID 0x8aba (35514) cujo TTL é igual a 4. Com isto, podemos realmente verificar que o TTL necessário para alcançar o servidor *Monstro* é 4.

### 1.1.c Alínea c)

O valor mínimo do campo TTL é 4, tal como se verificou na alínea anterior.

### 1.1.d Alínea d)

Com a execução do comando `tracert 10 -I 10.0.5.10`, obtivemos o seguinte *output*:

```
tracert to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 21.055 ms 21.032 ms 21.031 ms 21.031 ms 21.032 ms 21.031 ms * * * *
 2 10.0.1.2 (10.0.1.2) 41.433 ms 41.433 ms 41.433 ms 41.433 ms 41.432 ms 41.432 ms * * * *
 3 10.0.2.2 (10.0.2.2) 61.768 ms 61.768 ms 61.052 ms 61.025 ms 61.022 ms 61.021 ms * * * *
 4 10.0.5.10 (10.0.5.10) 81.938 ms 81.937 ms 83.314 ms 83.294 ms 83.058 ms 83.029 ms 83.025 ms 83.024 ms 82.666 ms 82.644 ms
root@Bela:/tmp/pycore.43739/Bela.conf#
```

Figura 4 - Execução do `tracert` com a especificação de envio de 10 pacotes

A média do RTT do envio dos 10 pacotes é de, aproximadamente, 82.7389 ms.

### 1.1.e Alínea e)

Não, porque existem várias razões para influenciar o resultado deste cálculo. Algumas delas são o pacote ter feito um caminho diferente da origem para o destino e do destino para a origem, filas de espera muito ocupadas e sobrecarga na rede.

## 1.2 Exercício 2

### 1.2.a Alínea a)

Obtivemos o seguinte *output* no Wireshark (após a execução dos comandos `tracert -I marco.uminho.pt` e `tracert -I marco.uminho.pt 4018`):

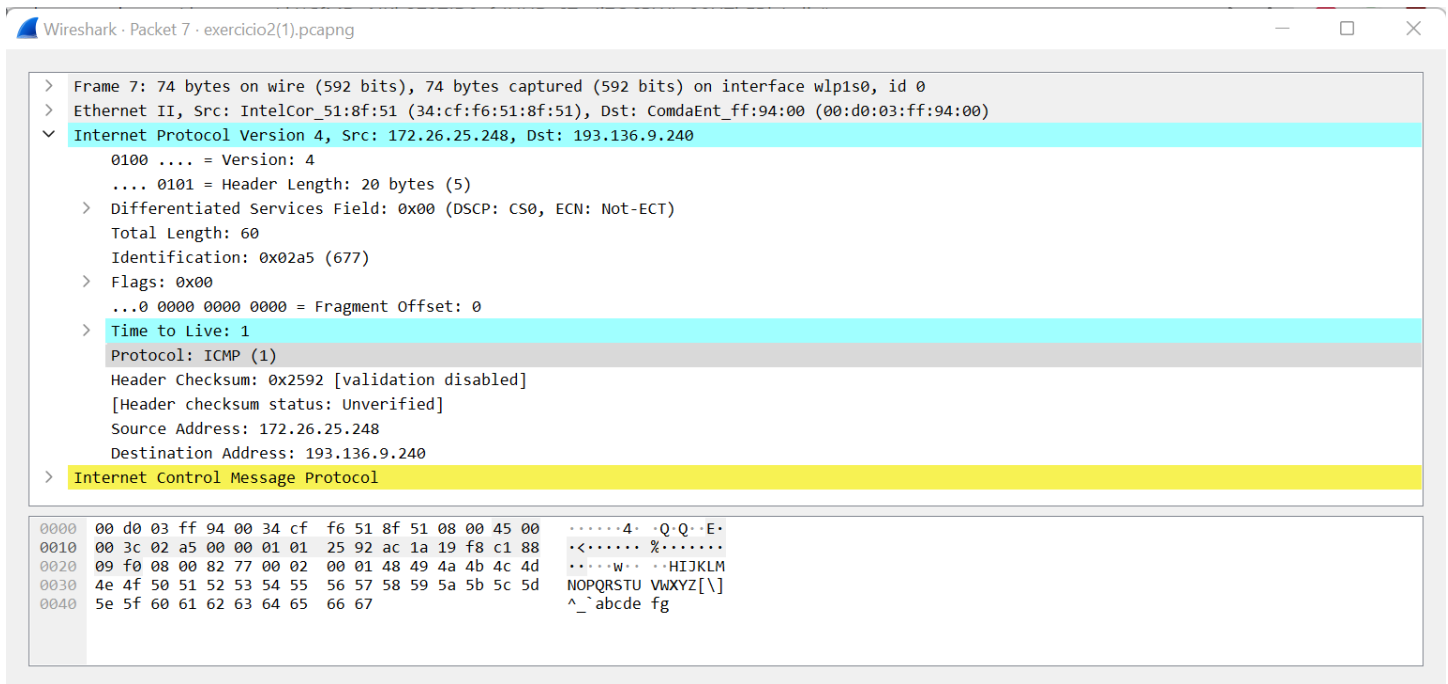
No.	Time	Source	Destination	Protocoll	Length	Info
7	0.433857082	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=1/256, ttl=1 (no response found!)
8	0.433920076	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=2/512, ttl=1 (no response found!)
9	0.433934723	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=3/768, ttl=1 (no response found!)
10	0.433967606	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=4/1024, ttl=2 (no response found!)
11	0.433960083	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=5/1280, ttl=2 (no response found!)
12	0.433972356	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=6/1536, ttl=2 (no response found!)
13	0.433984969	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=7/1792, ttl=3 (no response found!)
14	0.433996496	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=8/2048, ttl=3 (no response found!)
15	0.434008023	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=9/2304, ttl=3 (no response found!)
16	0.434020568	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=10/2560, ttl=4 (reply in 24)
17	0.434032502	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=11/2816, ttl=4 (reply in 26)
18	0.434044233	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=12/3072, ttl=4 (reply in 28)
19	0.434056167	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=13/3328, ttl=5 (reply in 30)
20	0.434068644	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=14/3584, ttl=5 (reply in 31)
21	0.434079900	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=15/3840, ttl=5 (reply in 32)
22	0.434091428	172.26.25.248	193.136.9.240	ICMP	74	Echo (ping) request id=0x0002, seq=16/4096, ttl=6 (reply in 33)
23	0.437845161	172.26.254.254	172.26.25.248	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
24	0.437845161	193.136.9.240	172.26.25.248	ICMP	74	Echo (ping) reply id=0x0002, seq=10/2560, ttl=61 (request in 16)
26	0.437846110	193.136.9.240	172.26.25.248	ICMP	74	Echo (ping) reply id=0x0002, seq=11/2816, ttl=61 (request in 17)
27	0.437846110	172.26.254.254	172.26.25.248	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
28	0.437847466	193.136.9.240	172.26.25.248	ICMP	74	Echo (ping) reply id=0x0002, seq=12/3072, ttl=61 (request in 18)
29	0.437847466	172.26.254.254	172.26.25.248	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
30	0.437848822	193.136.9.240	172.26.25.248	ICMP	74	Echo (ping) reply id=0x0002, seq=13/3328, ttl=61 (request in 19)
31	0.437850179	193.136.9.240	172.26.25.248	ICMP	74	Echo (ping) reply id=0x0002, seq=14/3584, ttl=61 (request in 20)

Figura 5 - *Output* Wireshark (máquina nativa)

Como se pode verificar na figura 5, o endereço IP da interface ativa na máquina que executou o `tracert` é 172.26.25.248

## 1.2.b Alínea b)

Na figura seguinte apresentam-se os detalhes da primeira mensagem ICMP capturada.



**Figura 6** - Detalhes da primeira mensagem ICMP capturada (com destaque no campo 'Protocol')

O valor do campo 'Protocol' é 0x01 (ICMP (1)).

O valor deste campo identifica o protocolo específico da camada de transporte para o qual a porção de dados deste datagrama IP deve ser passada. (Fonte: Livro 'Computer Networking: A Top-Down Approach').

## 1.2.c Alínea c)

Como se verifica na figura 6 (alínea anterior), o tamanho do cabeçalho IPv4 é de 20 bytes.

O tamanho total de um datagrama IP é o tamanho do cabeçalho mais o *payload*, i.e., os dados. (fonte:Livro 'Computer Networking: A Top-Down Approach'). Deste modo, conseguimos facilmente obter o tamanho do *payload* subtraindo ao *total length* o valor de *header length*.

Assim, o campo de dados tem 40 bytes.

### 1.2.d Alínea d)

Na figura seguinte apresentam-se detalhes da primeira mensagem ICMP capturada, destacando as flags do datagrama.

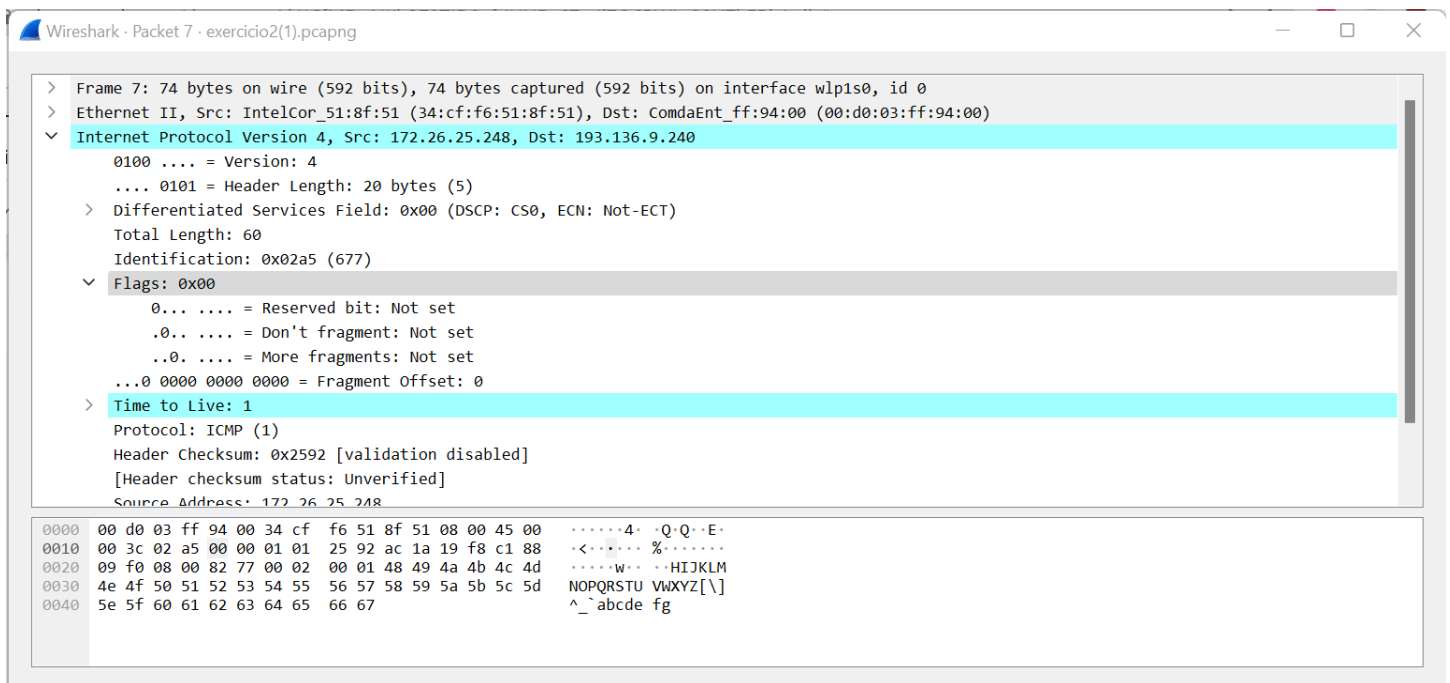


Figura 7 - Detalhes da primeira mensagem ICMP capturada (com destaque no campo das flags e fragment offset)

Uma vez que na figura acima conseguimos ver que a flag 'More fragments' e o fragment offset estão a zero. Logo, é verdade que o pacote não foi fragmentado.

### 1.2.e Alínea e)

Os campos que variam de pacote para pacote são o 'Identification', 'Time to Live' e 'Header checksum'.

### 1.2.f Alínea f)

Ao analisar o tráfego, conseguimos verificar que o campo 'Identification' aumenta de pacote para pacote e o campo 'Time to Live' aumenta de três em três pacotes. Isto deve-se ao facto do traceroute enviar três pacotes por predefinição em que todos têm o mesmo TTL.

### 1.2.g Alínea g)

O TTL não permanece constante porque os pacotes são provenientes de dispositivos que estão a distâncias diferentes do host de origem. Isto acontece uma vez que o traceroute envia pacotes com TTL diferentes que permitiu que estes atingissem distâncias diferentes na rede (em termos de saltos).



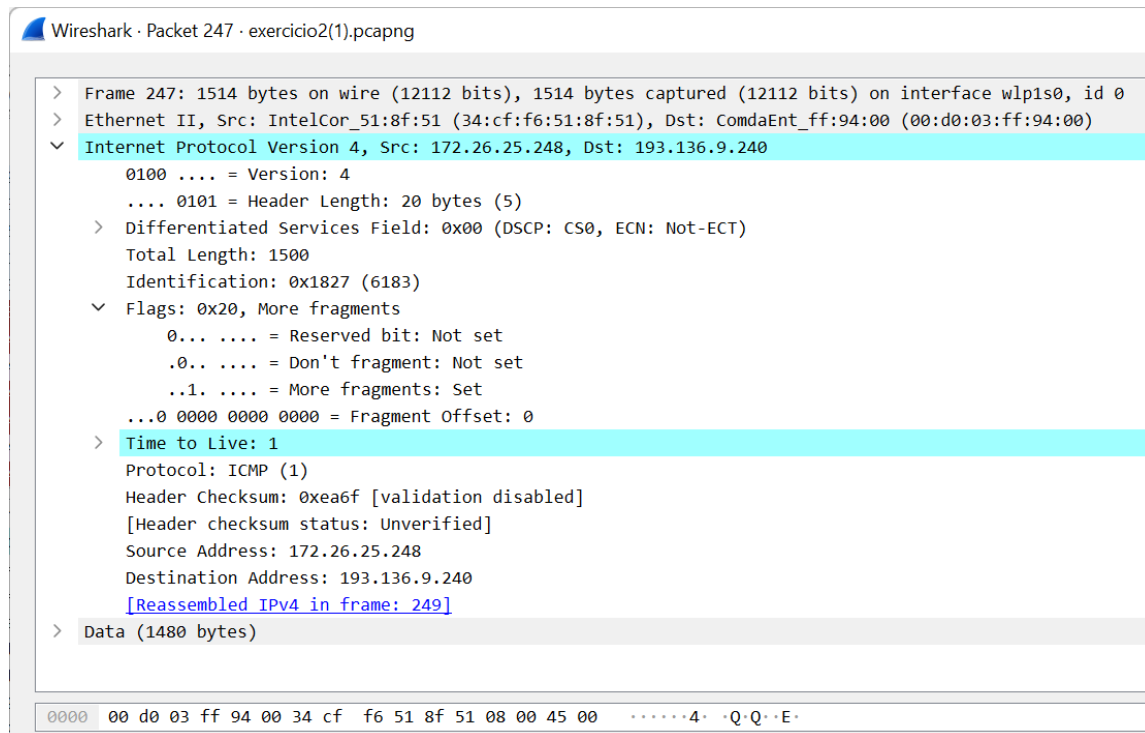
## 1.3 Exercício 3

### 1.3.a Alínea a)

Houve a necessidade de fragmentar o pacote uma vez que o seu tamanho (4018 bytes) ultrapassa o MTU de 1500 bytes.

### 1.3.b Alínea b)

Na figura seguinte, apresentámos o primeiro fragmento da primeira mensagem ICMP captada no contexto dos pacotes com tamanho de 4018 bytes.



**Figura 8 - Primeiro fragmento IP**

Como a flag *More fragments* tem valor 1, conseguimos concluir que o pacote foi fragmentado. Uma vez que sabemos que existem mais fragmentos e que o *fragment offset* está a 0, então verificámos que este é o primeiro fragmento do pacote. Conforme indicado na figura acima, o tamanho deste datagrama é de 1500 bytes.

### 1.3.c Alínea c)

Na figura seguinte, apresentámos o segundo fragmento do datagrama IP original.

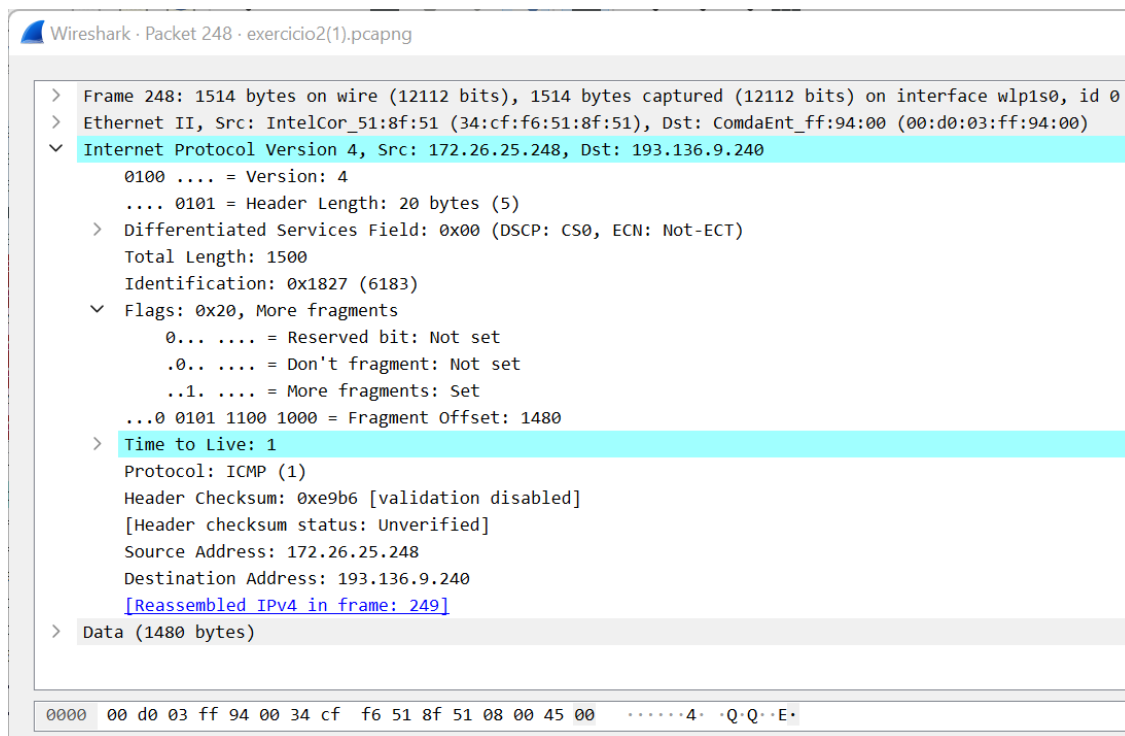


Figura 9 - Segundo fragmento

Como o *fragment offset* tem o valor 1480, podemos concluir que este não é o primeiro fragmento do datagrama original. Existem mais fragmentos a seguir a este, pois a flag *More fragments* é diferente de 0.

### 1.3.d Alínea d)

O datagrama original foi dividido em três fragmentos.

### 1.3.e Alínea e)

Os campos do cabeçalho IP que mudam entre os fragmentos são a flag *More fragments* e o *fragment offset*. À medida que os fragmentos vão chegando ao *host* de destino, este sabe que recebeu o primeiro fragmento quando a flag *More fragments* está a 1 e o *fragment offset* a 0. Posteriormente, ele vai recebendo os seguintes fragmentos cujos valores da flag *More fragments* e do *fragment offset* são diferentes de 0. Ao receber o último fragmento, o *fragment offset* será diferente de 0 e a flag *More fragments* terá o valor 0. No fim deste processo, ele faz o *reassembly* destes fragmentos cuja ordem é dada pelo *fragment offset*.

### 1.3.f Alínea f)

L = 4018 bytes

MTU = 1500 bytes

TTL = 1

ID = 0x1827 (6183)

**Fragmento 1:** 3998 - 1480 = 2518 bytes

Dados (*payload*): 1480 bytes

TTL = 0

ID = 0x1827 (6183)

More fragments = 1

Fragment offset = 0

**Fragmento 2:** 2518 - 1480 = 1038 bytes

Dados (*payload*): 1480 bytes

TTL = 0

ID = 0x1827 (6183)

More fragments = 1

Fragment offset = 1480

**Fragmento 3:**

Dados (*payload*): 1038 bytes

TTL = 0

ID = 0x1827 (6183)

More fragments = 0

Fragment offset = 2960

### 1.3.g Alínea g)

*More Fragments* == 0 && *fragment offset* > 0

## 2 - Segunda parte

### 2.1 Exercício 1

#### 2.1.a Alínea a)

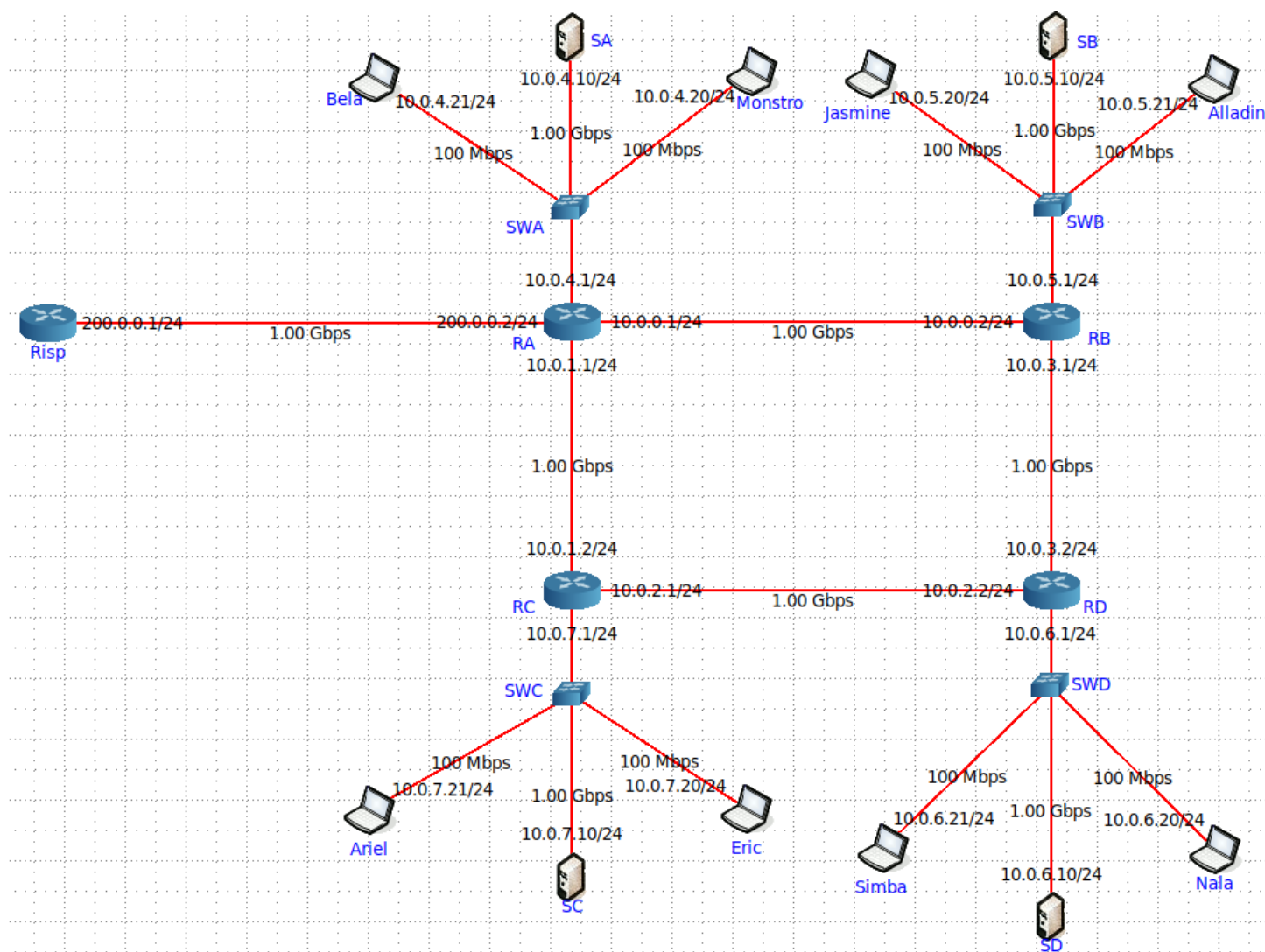


Figura 10 - Topologia Core conforme descrita no enunciado

### 2.1.b Alínea b)

Existem três gamas de endereços IP privados, apresentadas a seguir:

- 10.0.0.0 até 10.255.255.255
- 172.16.0.0 até 172.31.255.255
- 192.168.0.0 até 192.168.255.255

Uma vez que os endereços da nossa topologia pertencem à gama [10.0.0.0, 10.255.255.255], então podemos concluir que os endereços são privados.

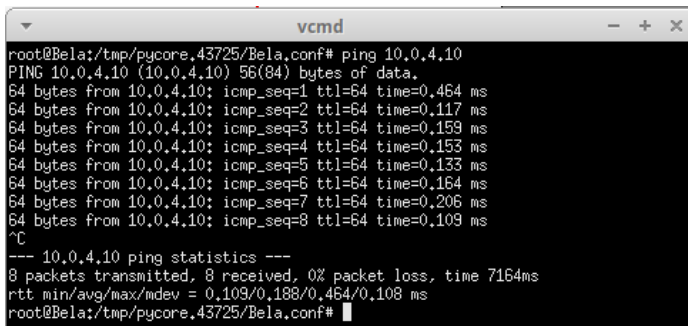
Em particular, os endereços 200.0.0.1 e 200.0.0.2 são públicos e correspondem à conectividade externa.

### 2.1.c Alínea c)

Os *switches* atuam na camada de rede de nível 2 enquanto que o IP atua na camada de rede de nível 3. Por este motivo, um *switch* é “invisível” no contexto do protocolo IP e, deste modo, não lhe é atribuído um endereço.

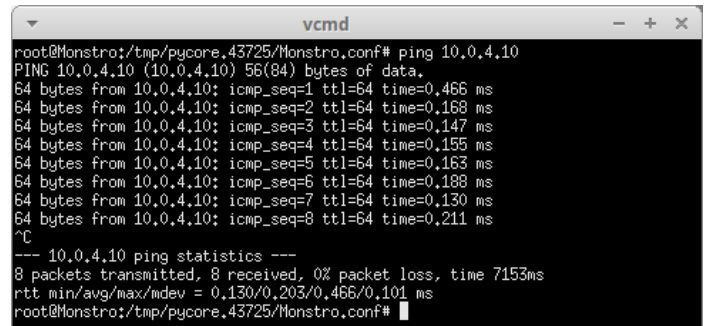
### 2.1.d Alínea d)

- Departamento A (Endereço IP do servidor SA: 10.0.4.10)



```
vcmd
root@Bela:/tmp/pycore.43725/Bela.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=64 time=0.464 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.117 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=0.159 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=64 time=0.153 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=64 time=0.133 ms
64 bytes from 10.0.4.10: icmp_seq=6 ttl=64 time=0.164 ms
64 bytes from 10.0.4.10: icmp_seq=7 ttl=64 time=0.206 ms
64 bytes from 10.0.4.10: icmp_seq=8 ttl=64 time=0.109 ms
^C
--- 10.0.4.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7164ms
rtt min/avg/max/mdev = 0.109/0.188/0.464/0.108 ms
root@Bela:/tmp/pycore.43725/Bela.conf#
```

Figura 11 - Comando ping 10.0.4.10 no pc Bela



```
vcmd
root@Monstro:/tmp/pycore.43725/Monstro.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=64 time=0.466 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.168 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=0.147 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=64 time=0.155 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=64 time=0.163 ms
64 bytes from 10.0.4.10: icmp_seq=6 ttl=64 time=0.188 ms
64 bytes from 10.0.4.10: icmp_seq=7 ttl=64 time=0.130 ms
64 bytes from 10.0.4.10: icmp_seq=8 ttl=64 time=0.211 ms
^C
--- 10.0.4.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7153ms
rtt min/avg/max/mdev = 0.130/0.203/0.466/0.101 ms
root@Monstro:/tmp/pycore.43725/Monstro.conf#
```

Figura 12 - Comando ping 10.0.4.10 no pc Monstro

Como podemos observar nas figuras 11 e 12, existe conexão entre os PC's Bela e Monstro e o servidor do departamento A. Deste modo, podemos concluir que existe conectividade interna no departamento A.

- Departamento B (Endereço IP do servidor SB: 10.0.5.10)

```

root@Jasmine:/tmp/pycore.43725/Jasmine.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=64 time=0.546 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=64 time=0.088 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=64 time=0.109 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=64 time=0.237 ms
64 bytes from 10.0.5.10: icmp_seq=5 ttl=64 time=0.221 ms
64 bytes from 10.0.5.10: icmp_seq=6 ttl=64 time=0.220 ms
64 bytes from 10.0.5.10: icmp_seq=7 ttl=64 time=0.182 ms
64 bytes from 10.0.5.10: icmp_seq=8 ttl=64 time=0.166 ms
^C
--- 10.0.5.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7151ms
rtt min/avg/max/mdev = 0.088/0.221/0.546/0.132 ms
root@Jasmine:/tmp/pycore.43725/Jasmine.conf#
    
```

Figura 13 - Comando ping 10.0.5.10 no pc Jasmine

```

root@Alladin:/tmp/pycore.43725/Alladin.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=64 time=0.560 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=64 time=0.169 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=64 time=0.141 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=64 time=0.148 ms
64 bytes from 10.0.5.10: icmp_seq=5 ttl=64 time=0.148 ms
64 bytes from 10.0.5.10: icmp_seq=6 ttl=64 time=0.109 ms
64 bytes from 10.0.5.10: icmp_seq=7 ttl=64 time=0.140 ms
64 bytes from 10.0.5.10: icmp_seq=8 ttl=64 time=0.092 ms
^C
--- 10.0.5.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7151ms
rtt min/avg/max/mdev = 0.092/0.188/0.560/0.142 ms
root@Alladin:/tmp/pycore.43725/Alladin.conf#
    
```

Figura 14 - Comando ping 10.0.5.10 no pc Alladin

Como podemos observar nas figuras 13 e 14, existe conexão entre os PC's Jasmine e Alladdin e o servidor do departamento B. Deste modo, podemos concluir que existe conectividade interna no departamento B.

- Departamento C (Endereço IP do servidor SC: 10.0.7.10)

```

root@Ariel:/tmp/pycore.43725/Ariel.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=64 time=0.543 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=64 time=0.179 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=64 time=0.094 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=64 time=0.178 ms
64 bytes from 10.0.7.10: icmp_seq=5 ttl=64 time=0.129 ms
64 bytes from 10.0.7.10: icmp_seq=6 ttl=64 time=0.137 ms
64 bytes from 10.0.7.10: icmp_seq=7 ttl=64 time=0.136 ms
64 bytes from 10.0.7.10: icmp_seq=8 ttl=64 time=0.156 ms
^C
--- 10.0.7.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7148ms
rtt min/avg/max/mdev = 0.094/0.194/0.543/0.134 ms
root@Ariel:/tmp/pycore.43725/Ariel.conf#
    
```

Figura 15 - Comando ping 10.0.7.10 no pc Ariel

```

root@Eric:/tmp/pycore.43725/Eric.conf# ping 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data.
64 bytes from 10.0.7.10: icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=64 time=0.102 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=64 time=0.159 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=64 time=0.179 ms
64 bytes from 10.0.7.10: icmp_seq=5 ttl=64 time=0.223 ms
64 bytes from 10.0.7.10: icmp_seq=6 ttl=64 time=0.166 ms
64 bytes from 10.0.7.10: icmp_seq=7 ttl=64 time=0.227 ms
64 bytes from 10.0.7.10: icmp_seq=8 ttl=64 time=0.140 ms
^C
--- 10.0.7.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7147ms
rtt min/avg/max/mdev = 0.102/0.231/0.653/0.164 ms
root@Eric:/tmp/pycore.43725/Eric.conf#
    
```

Figura 16 - Comando ping 10.0.7.10 no pc Eric

Como podemos observar nas figuras 15 e 16, existe conexão entre os PC's Ariel e Eric e o servidor do departamento C. Deste modo, podemos concluir que existe conectividade interna no departamento C.

- Departamento D (Endereço IP do servidor SD: 10.0.6.10)

```

root@Simba:/tmp/pycore.43725/Simba.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_seq=1 ttl=64 time=0.521 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=64 time=0.164 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=64 time=0.164 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=64 time=0.162 ms
64 bytes from 10.0.6.10: icmp_seq=6 ttl=64 time=0.261 ms
64 bytes from 10.0.6.10: icmp_seq=7 ttl=64 time=0.175 ms
64 bytes from 10.0.6.10: icmp_seq=8 ttl=64 time=0.232 ms
^C
--- 10.0.6.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7150ms
rtt min/avg/max/mdev = 0.162/0.232/0.521/0.114 ms
root@Simba:/tmp/pycore.43725/Simba.conf#
    
```

Figura 17 - Comando ping 10.0.6.10 no pc Simba

```

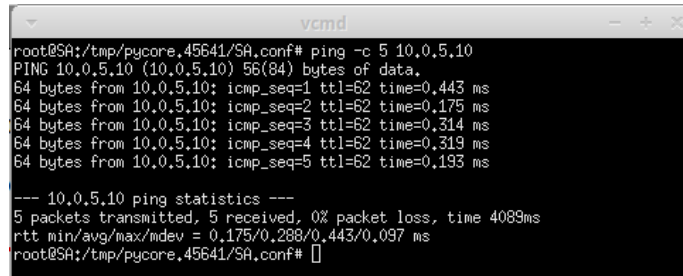
root@Nala:/tmp/pycore.43725/Nala.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_seq=1 ttl=64 time=0.535 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=64 time=0.172 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=64 time=0.088 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=64 time=0.153 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=64 time=0.103 ms
64 bytes from 10.0.6.10: icmp_seq=6 ttl=64 time=0.104 ms
64 bytes from 10.0.6.10: icmp_seq=7 ttl=64 time=0.142 ms
64 bytes from 10.0.6.10: icmp_seq=8 ttl=64 time=0.158 ms
^C
--- 10.0.6.10 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7158ms
rtt min/avg/max/mdev = 0.088/0.181/0.535/0.136 ms
root@Nala:/tmp/pycore.43725/Nala.conf#
    
```

Figura 18 - Comando ping 10.0.6.10 no pc Nala

Como podemos observar nas figuras 17 e 18, existe conexão entre os PC's Simba e Nala e o servidor do departamento D. Deste modo, podemos concluir que existe conectividade interna no departamento D.

### 2.1.e Alínea e)

- Comunicação do servidor A com os servidores B, C e D.

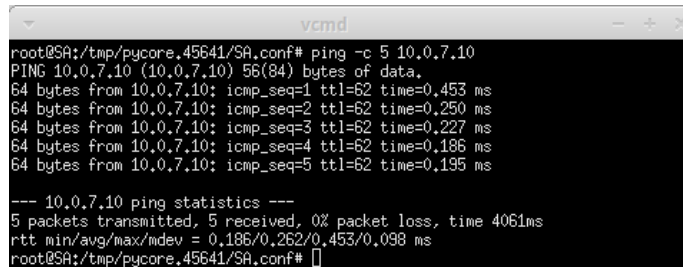


```

vcmd
root@SA:/tmp/pycore.45641/SA.conf# ping -c 5 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
64 bytes from 10.0.5.10: icmp_seq=1 ttl=62 time=0.443 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=62 time=0.175 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=62 time=0.314 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=62 time=0.319 ms
64 bytes from 10.0.5.10: icmp_seq=5 ttl=62 time=0.193 ms

--- 10.0.5.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4089ms
rtt min/avg/max/mdev = 0.175/0.288/0.443/0.097 ms
root@SA:/tmp/pycore.45641/SA.conf#
    
```

Figura 19 - Comunicação bem sucedida com o servidor B

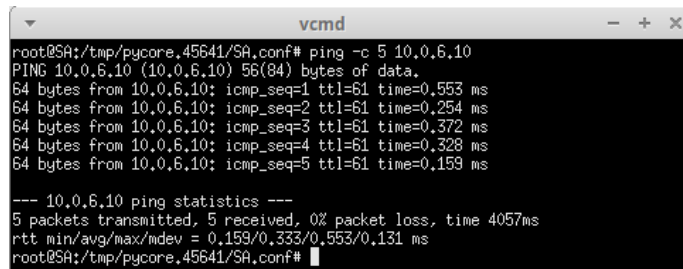


```

vcmd
root@SA:/tmp/pycore.45641/SA.conf# ping -c 5 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data:
64 bytes from 10.0.7.10: icmp_seq=1 ttl=62 time=0.453 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=62 time=0.250 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=62 time=0.227 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=62 time=0.186 ms
64 bytes from 10.0.7.10: icmp_seq=5 ttl=62 time=0.195 ms

--- 10.0.7.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4061ms
rtt min/avg/max/mdev = 0.186/0.262/0.453/0.098 ms
root@SA:/tmp/pycore.45641/SA.conf#
    
```

Figura 20 - Comunicação bem sucedida com o servidor C



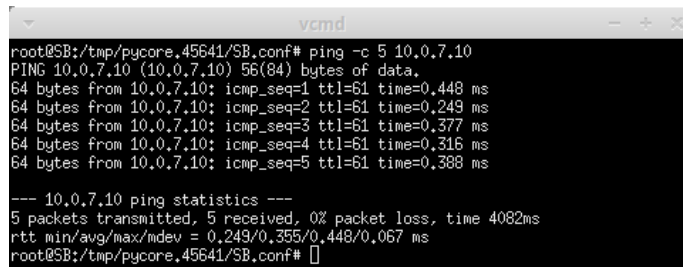
```

vcmd
root@SA:/tmp/pycore.45641/SA.conf# ping -c 5 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data:
64 bytes from 10.0.6.10: icmp_seq=1 ttl=61 time=0.553 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=61 time=0.254 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=61 time=0.372 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=61 time=0.328 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=61 time=0.159 ms

--- 10.0.6.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4057ms
rtt min/avg/max/mdev = 0.159/0.333/0.553/0.131 ms
root@SA:/tmp/pycore.45641/SA.conf#
    
```

Figura 21 - Comunicação bem sucedida com o servidor D

- Comunicação do servidor B com os servidores C e D.

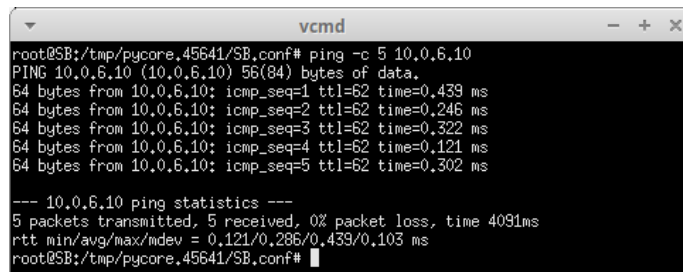


```

vcmd
root@SB:/tmp/pycore.45641/SB.conf# ping -c 5 10.0.7.10
PING 10.0.7.10 (10.0.7.10) 56(84) bytes of data:
64 bytes from 10.0.7.10: icmp_seq=1 ttl=61 time=0.448 ms
64 bytes from 10.0.7.10: icmp_seq=2 ttl=61 time=0.249 ms
64 bytes from 10.0.7.10: icmp_seq=3 ttl=61 time=0.377 ms
64 bytes from 10.0.7.10: icmp_seq=4 ttl=61 time=0.316 ms
64 bytes from 10.0.7.10: icmp_seq=5 ttl=61 time=0.388 ms

--- 10.0.7.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4082ms
rtt min/avg/max/mdev = 0.249/0.355/0.448/0.067 ms
root@SB:/tmp/pycore.45641/SB.conf#
    
```

Figura 22 - Comunicação bem sucedida com o servidor C

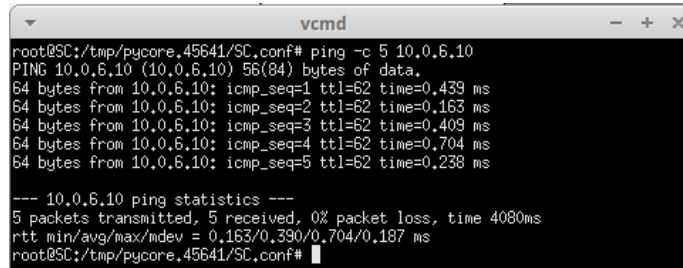


```
vcmd
root@SB:/tmp/pycore.45641/SB.conf# ping -c 5 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data:
64 bytes from 10.0.6.10: icmp_seq=1 ttl=62 time=0.439 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=62 time=0.246 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=62 time=0.322 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=62 time=0.121 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=62 time=0.302 ms

--- 10.0.6.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4091ms
rtt min/avg/max/mdev = 0.121/0.286/0.439/0.103 ms
root@SB:/tmp/pycore.45641/SB.conf#
```

Figura 23 - Comunicação bem sucedida com o servidor D

- Comunicação do servidor C com o servidor D.

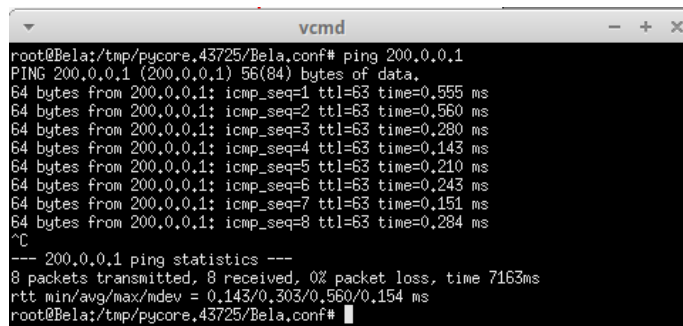


```
vcmd
root@SC:/tmp/pycore.45641/SC.conf# ping -c 5 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data:
64 bytes from 10.0.6.10: icmp_seq=1 ttl=62 time=0.439 ms
64 bytes from 10.0.6.10: icmp_seq=2 ttl=62 time=0.163 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=62 time=0.409 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=62 time=0.704 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=62 time=0.238 ms

--- 10.0.6.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4080ms
rtt min/avg/max/mdev = 0.163/0.390/0.704/0.187 ms
root@SC:/tmp/pycore.45641/SC.conf#
```

Figura 24 - Comunicação bem sucedida com o servidor D

## 2.1.f Alínea f)



```
vcmd
root@Bela:/tmp/pycore.43725/Bela.conf# ping 200.0.0.1
PING 200.0.0.1 (200.0.0.1) 56(84) bytes of data:
64 bytes from 200.0.0.1: icmp_seq=1 ttl=63 time=0.555 ms
64 bytes from 200.0.0.1: icmp_seq=2 ttl=63 time=0.560 ms
64 bytes from 200.0.0.1: icmp_seq=3 ttl=63 time=0.280 ms
64 bytes from 200.0.0.1: icmp_seq=4 ttl=63 time=0.143 ms
64 bytes from 200.0.0.1: icmp_seq=5 ttl=63 time=0.210 ms
64 bytes from 200.0.0.1: icmp_seq=6 ttl=63 time=0.243 ms
64 bytes from 200.0.0.1: icmp_seq=7 ttl=63 time=0.151 ms
64 bytes from 200.0.0.1: icmp_seq=8 ttl=63 time=0.284 ms
^C
--- 200.0.0.1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7163ms
rtt min/avg/max/mdev = 0.143/0.303/0.560/0.154 ms
root@Bela:/tmp/pycore.43725/Bela.conf#
```

Figura 25 - Comando ping 200.0.0.1 no pc Bela

Como podemos observar através do comando ping 200.0.0.1, existe conectividade entre o pc Bela e o router Risp.



## 2.2 Exercício 2

### 2.2.a Alínea a)

Ao executar o comando `netstat -rn` no pc Bela e no router RA, obtemos as seguintes tabelas de encaminhamento:

```

vcmd
root@Bela:/tmp/pycore.43725/Bela.conf# netstat -rn
Kernel IP routing table
Destination    Gateway         Genmask         Flags   MSS Window  irtt  Iface
0.0.0.0        10.0.4.1        0.0.0.0         UG      0  0        0     eth0
10.0.4.0       0.0.0.0         255.255.255.0   U        0  0        0     eth0
root@Bela:/tmp/pycore.43725/Bela.conf#

```

Figura 26 - Comando `netstat -rn` no pc Bela

```

vcmd
root@RA:/tmp/pycore.43725/RA.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       0.0.0.0         255.255.255.0   U        0  0        0     eth1
10.0.2.0       10.0.1.2        255.255.255.0   UG      0  0        0     eth1
10.0.3.0       10.0.0.2        255.255.255.0   UG      0  0        0     eth0
10.0.4.0       0.0.0.0         255.255.255.0   U        0  0        0     eth2
10.0.5.0       10.0.0.2        255.255.255.0   UG      0  0        0     eth0
10.0.6.0       10.0.0.2        255.255.255.0   UG      0  0        0     eth0
10.0.7.0       10.0.1.2        255.255.255.0   UG      0  0        0     eth1
200.0.0.0      0.0.0.0         255.255.255.0   U        0  0        0     eth3
root@RA:/tmp/pycore.43725/RA.conf#

```

Figura 27 - Comando `netstat -rn` no router RA

Pela tabela de encaminhamento do pc Bela, conseguimos saber o seguinte:

- Um datagrama cujo destino é a rede *default* (qualquer destino que não o 10.0.4.0) será enviado à interface com o endereço 10.0.4.1 (router de acesso).
- Um datagrama cujo destino é a própria rede (10.0.4.0) não será enviado a nenhuma interface uma vez que a rede 10.0.4.0 é diretamente ligada.

Pela tabela de encaminhamento do router RA, conseguimos saber o seguinte:

- Um datagrama cujo destino seja um dos endereços [10.0.0.0, 10.0.1.0, 10.0.4.0, 200.0.0.0] não será enviado a nenhuma interface uma vez que as redes com esses endereços são diretamente ligadas.
- Um datagrama cujo destino seja um dos endereços [10.0.2.0, 10.0.7.0] será enviado à interface 10.0.1.2.
- Um datagrama cujo destino seja um dos endereços [10.0.3.0, 10.0.5.0, 10.0.6.0] será enviado à interface 10.0.0.2.

## 2.2.b Alínea b)

- Comando `ps -ax` no pc Bela



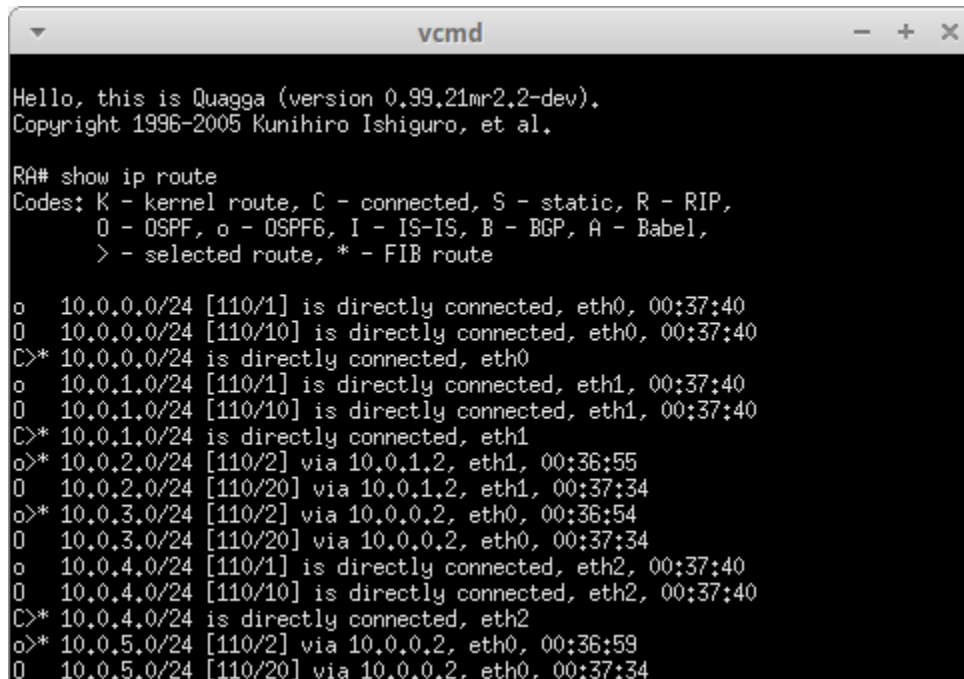
```

vcmd
root@Bela:/tmp/pycore.43725/Bela.conf# ps -ax
  PID TTY          STAT TIME COMMAND
    1 ?           S      0:00 vnodes -v -c /tmp/pycore.43725/Bela -l /tmp/pycore.
   125 pts/2    Ss      0:00 /bin/bash
   132 pts/2    R+      0:00 ps -ax
root@Bela:/tmp/pycore.43725/Bela.conf#

```

Figura 28 - Comando `ps -ax` no pc Bela

- Comando `show ip route` na *shell window* do router RA



```

vcmd
Hello, this is Quagga (version 0.99.21mr2.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

RA# show 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 [110/1] is directly connected, eth0, 00:37:40
O  10.0.0.0/24 [110/10] is directly connected, eth0, 00:37:40
C>* 10.0.0.0/24 is directly connected, eth0
o  10.0.1.0/24 [110/1] is directly connected, eth1, 00:37:40
O  10.0.1.0/24 [110/10] is directly connected, eth1, 00:37:40
C>* 10.0.1.0/24 is directly connected, eth1
o>* 10.0.2.0/24 [110/2] via 10.0.1.2, eth1, 00:36:55
O  10.0.2.0/24 [110/20] via 10.0.1.2, eth1, 00:37:34
o>* 10.0.3.0/24 [110/2] via 10.0.0.2, eth0, 00:36:54
O  10.0.3.0/24 [110/20] via 10.0.0.2, eth0, 00:37:34
o  10.0.4.0/24 [110/1] is directly connected, eth2, 00:37:40
O  10.0.4.0/24 [110/10] is directly connected, eth2, 00:37:40
C>* 10.0.4.0/24 is directly connected, eth2
o>* 10.0.5.0/24 [110/2] via 10.0.0.2, eth0, 00:36:59
O  10.0.5.0/24 [110/20] via 10.0.0.2, eth0, 00:37:34

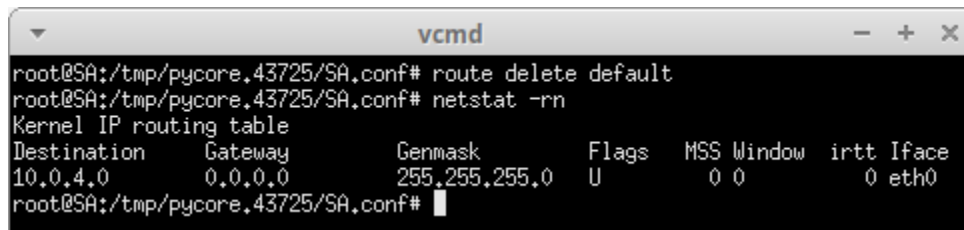
```

Figura 29 - Comando `show ip route` na *shell window* do router RA

Como podemos observar nas figuras 28 e 29, no router RA está a ser utilizado um encaminhamento dinâmico (OSPF - *Open Shortest Path First*). No pc Bela, não existe nenhuma referência acerca de encaminhamentos, pelo que podemos concluir que o pc está a utilizar encaminhamento estático.

### 2.2.c Alínea c)

Recorrendo ao comando `route delete default` no servidor SA, retirámos a rota por defeito e obtemos a seguinte tabela de encaminhamento:



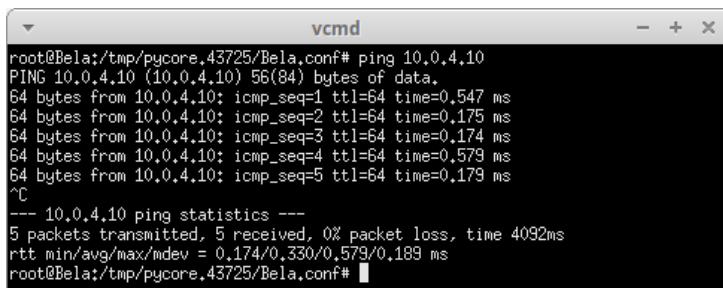
```

vcmd
root@SA:/tmp/pycore.43725/SA.conf# route delete default
root@SA:/tmp/pycore.43725/SA.conf# netstat -rn
Kernel IP routing table
Destination        Gateway           Genmask          Flags   MSS Window  irtt Iface
10.0.4.0            0.0.0.0          255.255.255.0    U        0  0      0  eth0
root@SA:/tmp/pycore.43725/SA.conf#

```

**Figura 30** - Tabela de encaminhamento após remover a rota por defeito do servidor SA.

Esta atualização à tabela não afeta os utilizadores que estão na mesma subrede do servidor SA. No entanto, todos os utilizadores fora da subrede de SA irão perder a comunicação com este mesmo servidor como podemos verificar nos exemplos das figuras que se seguem.

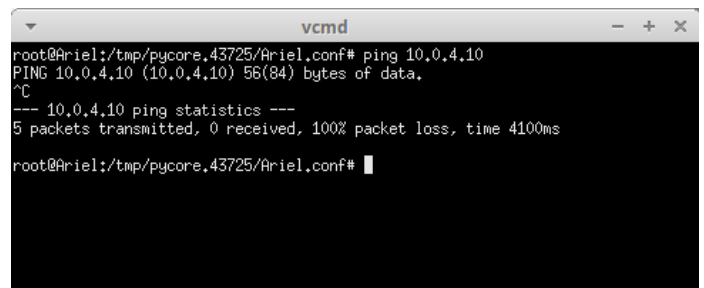


```

vcmd
root@Bela:/tmp/pycore.43725/Bela.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=64 time=0.547 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.175 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=0.174 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=64 time=0.579 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=64 time=0.179 ms
^C
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4092ms
rtt min/avg/max/mdev = 0.174/0.330/0.579/0.189 ms
root@Bela:/tmp/pycore.43725/Bela.conf#

```

**Figura 31** - ping 10.0.4.10 no pc Bela



```

vcmd
root@Ariel:/tmp/pycore.43725/Ariel.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
^C
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4100ms
root@Ariel:/tmp/pycore.43725/Ariel.conf#

```

**Figura 32** - ping 10.0.4.10 no pc Ariel

Como podemos verificar, o pc Bela (que se encontra na mesma rede que o servidor SA) ainda consegue comunicar com o servidor SA mas o pc Ariel (fora da subrede de SA) não, confirmando assim a afirmação anterior.

Isto acontece uma vez que o servidor SA só “sabe” enviar os pacotes para destinos referenciados na sua tabela de encaminhamento. Ora, na tabela de encaminhamento do servidor SA (figura 30) a única rede de destino referenciada é a rede que lhe está diretamente ligada (a sua subrede) pelo que nunca conseguirá responder a pedidos que venham de fora.

**2.2.d Alínea d)**

Para o servidor SA ficar novamente acessível, teremos de adicionar à sua tabela de encaminhamento as seguintes entradas:

#L	Destination	Gateway	Netmask	Interface
1	10.0.0.0	10.0.4.1	255.255.255.0	eth0
2	10.0.1.0	10.0.4.1	255.255.255.0	eth0
3	10.0.2.0	10.0.4.1	255.255.255.0	eth0
4	10.0.3.0	10.0.4.1	255.255.255.0	eth0
5	10.0.5.0	10.0.4.1	255.255.255.0	eth0
6	10.0.6.0	10.0.4.1	255.255.255.0	eth0
7	10.0.7.0	10.0.4.1	255.255.255.0	eth0
8	200.0.0.0	10.0.4.1	255.255.255.0	eth0

**Tabela 1-** Entradas necessárias para repor a comunicação com o servidor SA

Para adicionar as entradas da Tabela 1 na tabela de encaminhamento do servidor SA, utilizaram-se os seguintes comandos:

- `route add -net 10.0.0.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`
- `route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`
- `route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`
- `route add -net 10.0.3.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`
- `route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`
- `route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`
- `route add -net 10.0.7.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`
- `route add -net 200.0.0.0 netmask 255.255.255.0 gw 10.0.4.1 dev eth0`

Por forma a melhorar a solução anterior e reduzir o tamanho da nova tabela de encaminhamento do servidor SA, podemos recorrer ao *supernetting* e, com apenas duas novas entradas, podemos repor a comunicação do servidor com todas as outras subredes. Para isso, utilizou-se este comando: `route add -net 10.0.0.0 netmask 255.255.0.0 gw 10.0.4.1 dev eth0`.

Desta maneira, em vez de adicionar oito entradas na tabela de encaminhamento (Tabela 1), apenas teríamos de adicionar as seguintes entradas:

#L	Destination	Gateway	Netmask	Interface
1	10.0.0.0	10.0.4.1	255.255.0.0	eth0

2	200.0.0.0	10.0.4.1	255.255.255.0	eth0
---	-----------	----------	---------------	------

**Tabela 2** - Compactação das sete primeiras entradas da tabela 1 em apenas uma com *supernetting*

Esta optimização poderia levantar a seguinte questão: ‘Se adicionarmos a nova entrada com *supernetting*, não teremos um conflito devido a haver duas correspondências possíveis para o envio de pacotes para o destino 10.0.4.0?’. Ora, uma vez que o IP faz sempre o *longest prefix match*, ao enviar pacotes para o destino 10.0.4.0 será escolhida a entrada com o prefixo mais longo, i.e., [Dest: 10.0.4.0; Gateway: 0.0.0.0; Netmask: 255.255.255.0; Flags: U; ... ;Iface: eth0]. Logo, este problema não afeta o bom funcionamento da rede.

## 2.2.e Alínea e)

Testando a primeira solução da alínea anterior, temos o seguinte:

- Nova tabela de encaminhamento do servidor SA sem *supernetting*:

```

vcmd
root@SA:/tmp/pycore.39261/SA.conf# netstat -rn
Kernel IP routing table
Destination    Gateway         Genmask         Flags   MSS Window  irtt  Iface
10.0.0.0       10.0.4.1        255.255.255.0   UG        0  0          0 eth0
10.0.1.0       10.0.4.1        255.255.255.0   UG        0  0          0 eth0
10.0.2.0       10.0.4.1        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 eth0
10.0.4.0       0.0.0.0         255.255.255.0   U          0  0          0 eth0
10.0.5.0       10.0.4.1        255.255.255.0   UG        0  0          0 eth0
10.0.6.0       10.0.4.1        255.255.255.0   UG        0  0          0 eth0
10.0.7.0       10.0.4.1        255.255.255.0   UG        0  0          0 eth0
200.0.0.0      10.0.4.1        255.255.255.0   UG        0  0          0 eth0
root@SA:/tmp/pycore.39261/SA.conf#

```

**Figura 33** - Tabela de encaminhamento de SA com a comunicação reposta (sem *supernetting*)

- Comando `ping -c 5 10.0.4.10` no pc Jasmine (Departamento B).

```

vcmd
root@Jasmine:/tmp/pycore.39261/Jasmine.conf# ping -c 10.0.4.10
ping: invalid argument: '10.0.4.10'
root@Jasmine:/tmp/pycore.39261/Jasmine.conf# ping -c 5 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.585 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.157 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.222 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.117 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=62 time=0.322 ms
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4083ms
rtt min/avg/max/mdev = 0.117/0.280/0.585/0.167 ms

```

**Figura 34** - Comunicação restabelecida com o departamento B

- Comando `ping -c 5 10.0.4.10` no pc Ariel (Departamento C).

```

vcmd
root@Ariel:/tmp/pycore.39261/Ariel.conf# ping -c 5 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.585 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.261 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.251 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.321 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=62 time=0.266 ms
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4104ms
rtt min/avg/max/mdev = 0.251/0.336/0.585/0.126 ms

```

**Figura 35** - Comunicação restabelecida com o departamento C

- Comando `ping -c 5 10.0.4.10` no pc Simba (Departamento D).

```

vcmd
root@Simba:/tmp/pycore.39261/Simba.conf# ping -c 5 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=61 time=0.623 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=61 time=0.338 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=61 time=3.67 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=61 time=0.294 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=61 time=0.357 ms

--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4070ms
rtt min/avg/max/mdev = 0.294/1.056/3.670/1.311 ms

```

Figura 36 - Comunicação restabelecida com o departamento D

Testando a segunda solução da alínea anterior, temos o seguinte:

- Nova tabela de encaminhamento do servidor SA com *supernetting*:

```

vcmd
<A.conf# route add -net 10.0.0.0 netmask 255.255.0.0 gw 10.0.4.1 dev eth0
root@SA:/tmp/pycore.39261/SA.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt  Iface
10.0.0.0         10.0.4.1       255.255.0.0     UG        0 0          0     eth0
10.0.4.0         0.0.0.0        255.255.255.0   U          0 0          0     eth0
root@SA:/tmp/pycore.39261/SA.conf#

```

Figura 37 - Tabela de encaminhamento de SA com a comunicação reposta (com *supernetting*)

- Comando `ping -c 5 10.0.4.10` no pc Jasmine (Departamento B)

```

vcmd
root@Jasmine:/tmp/pycore.39261/Jasmine.conf# ping -c 5 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.654 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.285 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.259 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.125 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=62 time=0.240 ms

--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4087ms
rtt min/avg/max/mdev = 0.125/0.312/0.654/0.179 ms

```

Figura 38 - Comunicação restabelecida com o departamento B

- Comando `ping -c 5 10.0.4.10` no pc Ariel (Departamento C)

```

vcmd
root@Ariel:/tmp/pycore.39261/Ariel.conf# ping -c 5 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.672 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.265 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.209 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.283 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=62 time=0.254 ms

--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4097ms
rtt min/avg/max/mdev = 0.209/0.336/0.672/0.169 ms
root@Ariel:/tmp/pycore.39261/Ariel.conf#

```

Figura 39 - Comunicação restabelecida com o departamento C

- Comando `ping -c 5 10.0.4.10` no pc Simba (Departamento D)

```

root@Simba:/tmp/pycore.39261/Simba.conf# ping -c 5 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_seq=1 ttl=61 time=0.587 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=61 time=0.358 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=61 time=0.924 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=61 time=0.314 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=61 time=0.211 ms

--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.211/0.478/0.924/0.254 ms
root@Simba:/tmp/pycore.39261/Simba.conf#
  
```

Figura 40 - Comunicação restabelecida com o departamento D

Através das demonstrações práticas apresentadas neste exercício, conseguimos realmente verificar que as duas soluções (com e sem *supernetting*) foram bem sucedidas a restabelecer a comunicação do servidor A após retirar a rota *default*.

## 2.3 Exercício 3

### 2.3.1 Alínea 1)

A nossa rede irá agora dispor do seguinte endereço IP: 192.168.18.128/25.

O endereço expandido em binário fica da seguinte forma:

**11000000.10101000.00010010.1** | **0000000**

Neste esquema de endereçamento, temos 7 bits para construir os nossos endereços. Destes bits, alocámos 4 para a subrede e 3 para o *host* (prevendo que o número de departamentos possa vir a aumentar).

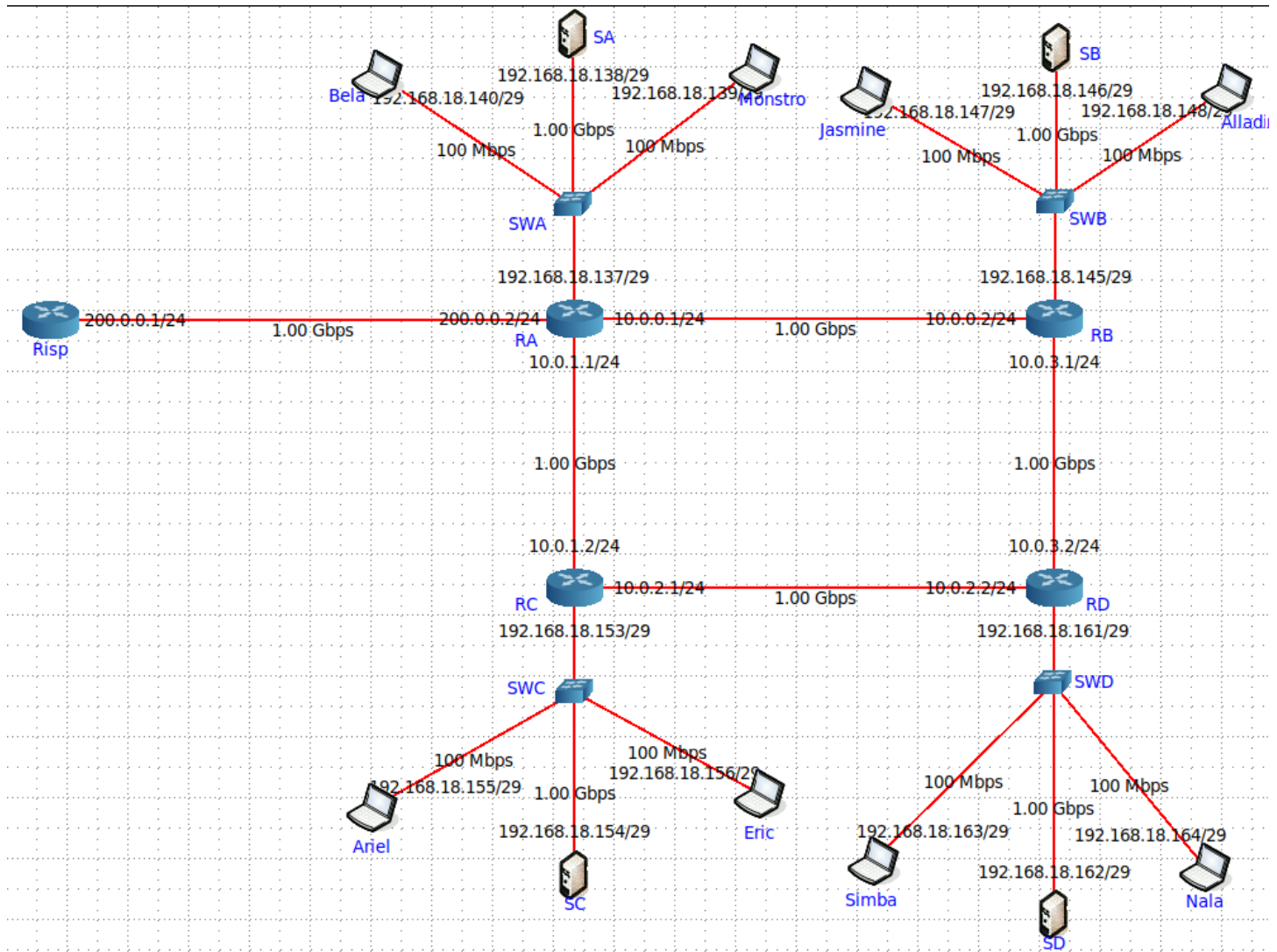
De acordo com as imposições do enunciado (não alterar a rede externa nem a rede *backbone*), devemos apenas atribuir endereços nas subredes dos departamentos. Assim, vamos designar estas redes por 'A', 'B', 'C' e 'D' (têm o mesmo nome do seu departamento).

Assim, temos o seguinte esquema de endereçamento:

Subrede	Endereço de subrede	Menor endereço (interfaces)	Maior endereço (interfaces)
A (0001)	192.168.18.10001000 /29 192.168.18.136 /29	192.168.18.10001001 /29 192.168.18.137 /29	192.168.18.10001110 /29 192.168.18.142 /29
B (0010)	192.168.18.10010000 /29 192.168.18.144 /29	192.168.18.10010001 /29 192.168.18.145 /29	192.168.18.10010110 /29 192.168.18.150 /29
C (0011)	192.168.18.10011000 /29 192.168.18.152 /29	192.168.18.10011001 /29 192.168.18.153 /29	192.168.18.10011110 /29 192.168.18.158 /29
D (0100)	192.168.18.10100000 /29 192.168.18.160 /29	192.168.18.10100001 /29 192.168.18.161 /29	192.168.18.10100110 /29 192.168.18.166 /29

Tabela 3 - Esquema de endereçamento para as subredes dos departamentos

Com o esquema de endereçamento da Tabela 3, podemos formar a seguinte topologia:



**Figura 41** - Topologia com o novo esquema de endereçamento

Decidimos usar este esquema uma vez que 3 bits são suficientes para endereçar as interfaces presentes em cada departamento e deixámos 4 bits para a subrede uma vez que o número de departamentos poderá vir a aumentar (conforme dito no enunciado).



### 2.3.2 Alínea 2)

Como utilizámos apenas 3 bits para cada *host* (isto é, os endereços são /29 na notação CIDR), então a máscara de rede utilizada é 255.255.255.248.

Ao utilizar 3 bits para o *host*, o número de *hosts* que podemos interligar em cada departamento é  $2^3 - 3 = 5$ , em que os três endereços que retirámos são os correspondentes a 000 (subrede), 111 (*broadcast*) e ao endereço atribuído à interface do *router* de acesso.

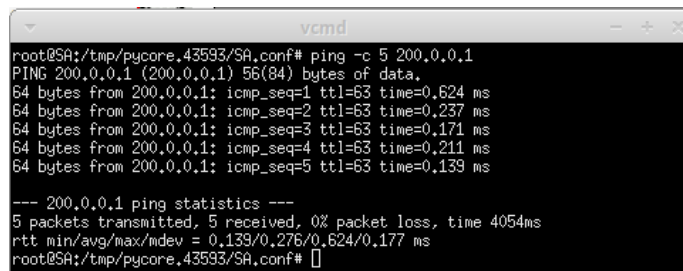
Ao utilizar 4 bits para prefixo de subrede, temos  $2^4 = 16$  prefixos disponíveis. Como já utilizámos quatro na alínea anterior, sobram 12 para uso futuro.

### 2.3.3 Alínea 3)

Fizemos as seguintes verificações para confirmar que a conectividade da rede local LEI-RC é mantida:

1. Verificar a conectividade do servidor A com o router Risp e os servidores B,C e D.
2. Verificar a conectividade do servidor B com o router Risp e os servidores C e D.
3. Verificar a conectividade do servidor C com o router Risp e o servidor D.
4. Verificar a conectividade do servidor D com o router Risp.

- Servidor A

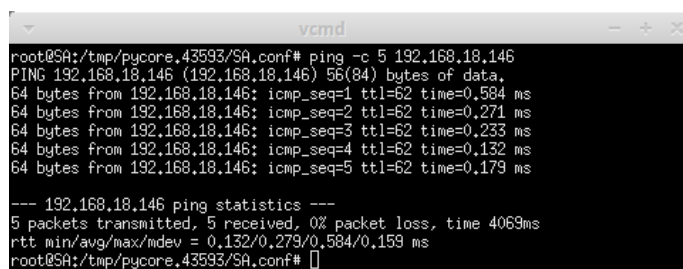


```

vcmd
root@SA:/tmp/pycore.43593/SA.conf# ping -c 5 200.0.0.1
PING 200.0.0.1 (200.0.0.1) 56(84) bytes of data:
64 bytes from 200.0.0.1: icmp_seq=1 ttl=63 time=0.624 ms
64 bytes from 200.0.0.1: icmp_seq=2 ttl=63 time=0.237 ms
64 bytes from 200.0.0.1: icmp_seq=3 ttl=63 time=0.171 ms
64 bytes from 200.0.0.1: icmp_seq=4 ttl=63 time=0.211 ms
64 bytes from 200.0.0.1: icmp_seq=5 ttl=63 time=0.139 ms

--- 200.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4054ms
rtt min/avg/max/mdev = 0.139/0.276/0.624/0.177 ms
root@SA:/tmp/pycore.43593/SA.conf#
    
```

Figura 42 - Comunicação SA com Risp



```

vcmd
root@SA:/tmp/pycore.43593/SA.conf# ping -c 5 192.168.18.146
PING 192.168.18.146 (192.168.18.146) 56(84) bytes of data:
64 bytes from 192.168.18.146: icmp_seq=1 ttl=62 time=0.584 ms
64 bytes from 192.168.18.146: icmp_seq=2 ttl=62 time=0.271 ms
64 bytes from 192.168.18.146: icmp_seq=3 ttl=62 time=0.233 ms
64 bytes from 192.168.18.146: icmp_seq=4 ttl=62 time=0.132 ms
64 bytes from 192.168.18.146: icmp_seq=5 ttl=62 time=0.179 ms

--- 192.168.18.146 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4069ms
rtt min/avg/max/mdev = 0.132/0.279/0.584/0.159 ms
root@SA:/tmp/pycore.43593/SA.conf#
    
```

Figura 43 - Comunicação SA com SB

```

vcmd
root@SA:/tmp/pycore.43593/SA.conf# ping -c 192.168.18.154
ping: invalid argument: '192.168.18.154'
root@SA:/tmp/pycore.43593/SA.conf# ping -c 5 192.168.18.154
PING 192.168.18.154 (192.168.18.154) 56(84) bytes of data:
64 bytes from 192.168.18.154: icmp_seq=1 ttl=62 time=0.541 ms
64 bytes from 192.168.18.154: icmp_seq=2 ttl=62 time=0.128 ms
64 bytes from 192.168.18.154: icmp_seq=3 ttl=62 time=0.296 ms
64 bytes from 192.168.18.154: icmp_seq=4 ttl=62 time=0.115 ms
64 bytes from 192.168.18.154: icmp_seq=5 ttl=62 time=0.164 ms

--- 192.168.18.154 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4083ms
rtt min/avg/max/mdev = 0.115/0.248/0.541/0.159 ms
root@SA:/tmp/pycore.43593/SA.conf#

```

Figura 44 - Comunicação SA com SC

```

vcmd
root@SA:/tmp/pycore.43593/SA.conf# ping -c 5 192.168.18.162
PING 192.168.18.162 (192.168.18.162) 56(84) bytes of data:
64 bytes from 192.168.18.162: icmp_seq=1 ttl=61 time=0.605 ms
64 bytes from 192.168.18.162: icmp_seq=2 ttl=61 time=0.328 ms
64 bytes from 192.168.18.162: icmp_seq=3 ttl=61 time=0.395 ms
64 bytes from 192.168.18.162: icmp_seq=4 ttl=61 time=0.377 ms
64 bytes from 192.168.18.162: icmp_seq=5 ttl=61 time=0.507 ms

--- 192.168.18.162 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4055ms
rtt min/avg/max/mdev = 0.328/0.442/0.605/0.100 ms
root@SA:/tmp/pycore.43593/SA.conf#

```

Figura 45 - Comunicação SA com SD

- Servidor B

```

vcmd
root@SB:/tmp/pycore.43593/SB.conf# ping -c 5 200.0.0.1
PING 200.0.0.1 (200.0.0.1) 56(84) bytes of data:
64 bytes from 200.0.0.1: icmp_seq=1 ttl=62 time=0.656 ms
64 bytes from 200.0.0.1: icmp_seq=2 ttl=62 time=0.257 ms
64 bytes from 200.0.0.1: icmp_seq=3 ttl=62 time=0.204 ms
64 bytes from 200.0.0.1: icmp_seq=4 ttl=62 time=0.155 ms
64 bytes from 200.0.0.1: icmp_seq=5 ttl=62 time=0.338 ms

--- 200.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4057ms
rtt min/avg/max/mdev = 0.155/0.322/0.656/0.177 ms
root@SB:/tmp/pycore.43593/SB.conf#

```

Figura 46 - Comunicação SB com Risp

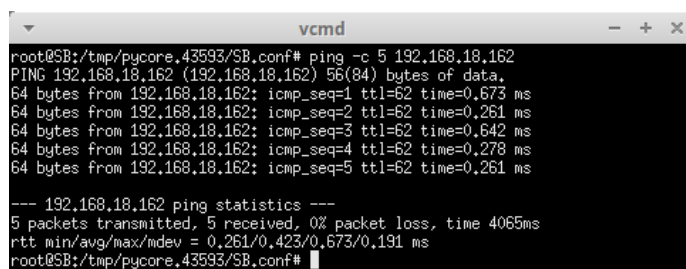
```

vcmd
root@SB:/tmp/pycore.43593/SB.conf# ping -c 5 192.168.18.154
PING 192.168.18.154 (192.168.18.154) 56(84) bytes of data:
64 bytes from 192.168.18.154: icmp_seq=1 ttl=61 time=0.474 ms
64 bytes from 192.168.18.154: icmp_seq=2 ttl=61 time=0.361 ms
64 bytes from 192.168.18.154: icmp_seq=3 ttl=61 time=0.402 ms
64 bytes from 192.168.18.154: icmp_seq=4 ttl=61 time=0.423 ms
64 bytes from 192.168.18.154: icmp_seq=5 ttl=61 time=0.631 ms

--- 192.168.18.154 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4074ms
rtt min/avg/max/mdev = 0.361/0.458/0.631/0.093 ms
root@SB:/tmp/pycore.43593/SB.conf#

```

Figura 47 - Comunicação SB com SC



```

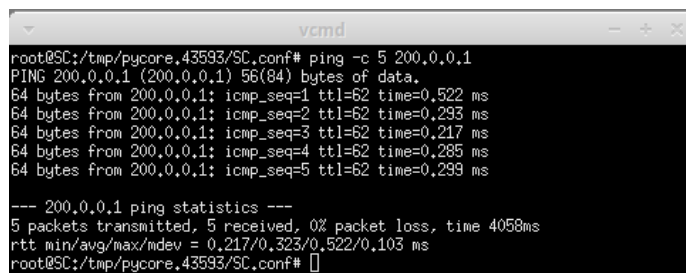
vcmd
root@SB:/tmp/pycore.43593/SB.conf# ping -c 5 192.168.18.162
PING 192.168.18.162 (192.168.18.162) 56(84) bytes of data:
64 bytes from 192.168.18.162: icmp_seq=1 ttl=62 time=0.673 ms
64 bytes from 192.168.18.162: icmp_seq=2 ttl=62 time=0.261 ms
64 bytes from 192.168.18.162: icmp_seq=3 ttl=62 time=0.642 ms
64 bytes from 192.168.18.162: icmp_seq=4 ttl=62 time=0.278 ms
64 bytes from 192.168.18.162: icmp_seq=5 ttl=62 time=0.261 ms

--- 192.168.18.162 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4065ms
rtt min/avg/max/mdev = 0.261/0.423/0.673/0.191 ms
root@SB:/tmp/pycore.43593/SB.conf#

```

Figura 48 - Comunicação SB com SD

- Servidor C



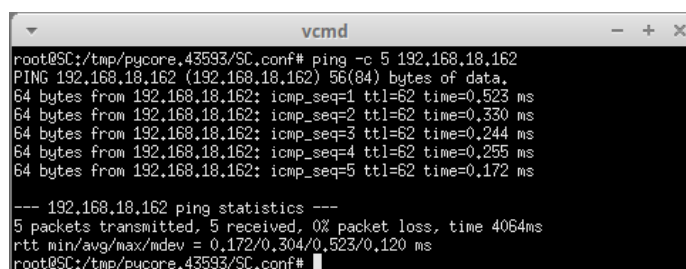
```

vcmd
root@SC:/tmp/pycore.43593/SC.conf# ping -c 5 200.0.0.1
PING 200.0.0.1 (200.0.0.1) 56(84) bytes of data:
64 bytes from 200.0.0.1: icmp_seq=1 ttl=62 time=0.522 ms
64 bytes from 200.0.0.1: icmp_seq=2 ttl=62 time=0.293 ms
64 bytes from 200.0.0.1: icmp_seq=3 ttl=62 time=0.217 ms
64 bytes from 200.0.0.1: icmp_seq=4 ttl=62 time=0.285 ms
64 bytes from 200.0.0.1: icmp_seq=5 ttl=62 time=0.293 ms

--- 200.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4058ms
rtt min/avg/max/mdev = 0.217/0.323/0.522/0.103 ms
root@SC:/tmp/pycore.43593/SC.conf#

```

Figura 49 - Comunicação SC com Risp



```

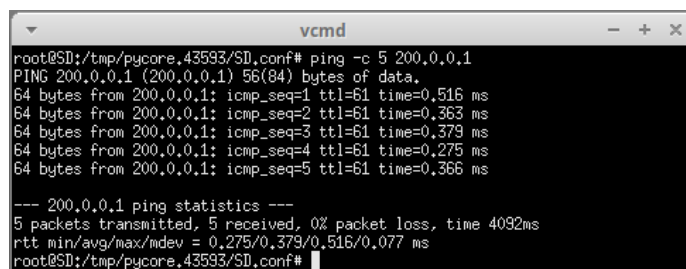
vcmd
root@SC:/tmp/pycore.43593/SC.conf# ping -c 5 192.168.18.162
PING 192.168.18.162 (192.168.18.162) 56(84) bytes of data:
64 bytes from 192.168.18.162: icmp_seq=1 ttl=62 time=0.523 ms
64 bytes from 192.168.18.162: icmp_seq=2 ttl=62 time=0.330 ms
64 bytes from 192.168.18.162: icmp_seq=3 ttl=62 time=0.244 ms
64 bytes from 192.168.18.162: icmp_seq=4 ttl=62 time=0.255 ms
64 bytes from 192.168.18.162: icmp_seq=5 ttl=62 time=0.172 ms

--- 192.168.18.162 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4064ms
rtt min/avg/max/mdev = 0.172/0.304/0.523/0.120 ms
root@SC:/tmp/pycore.43593/SC.conf#

```

Figura 50 - Comunicação SC com SD

- Servidor D



```

vcmd
root@SD:/tmp/pycore.43593/SD.conf# ping -c 5 200.0.0.1
PING 200.0.0.1 (200.0.0.1) 56(84) bytes of data:
64 bytes from 200.0.0.1: icmp_seq=1 ttl=61 time=0.516 ms
64 bytes from 200.0.0.1: icmp_seq=2 ttl=61 time=0.363 ms
64 bytes from 200.0.0.1: icmp_seq=3 ttl=61 time=0.379 ms
64 bytes from 200.0.0.1: icmp_seq=4 ttl=61 time=0.275 ms
64 bytes from 200.0.0.1: icmp_seq=5 ttl=61 time=0.366 ms

--- 200.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4092ms
rtt min/avg/max/mdev = 0.275/0.379/0.516/0.077 ms
root@SD:/tmp/pycore.43593/SD.conf#

```

Figura 51 - Comunicação SD com Risp

Como se pôde verificar, todos os elementos da rede LEI-RC conseguem comunicar entre si pelo que a conectividade é mantida.

## 3 - Conclusões

A realização deste trabalho permitiu aprofundar bastante os conhecimentos adquiridos nas aulas teóricas, ou seja, aplicar na prática como funciona o comando *traceroute*, fazer análise de pacotes IP e como é feita a sua fragmentação. Permitiu também perceber quais os procedimentos a tomar para saber se uma rede local está a funcionar devidamente e como consultar tabelas de encaminhamento e quais as implicações de eliminar rotas (nomeadamente a rota *default*) e também como corrigir possíveis erros no encaminhamento.

Houve também a possibilidade de definir o nosso próprio esquema de endereçamento permitindo saber o funcionamento desta área na realidade.