



Redes de Computadores
(3^o ano LEI)

Trabalho Prático 2

Relatório de Desenvolvimento

Grupo 4

a93241 Francisco Reis Izquierdo
a89526 Duarte Augusto Rodrigues Lucas
a96277 Diogo Miguel Serra Silva

24 de Março de 2022

Conteúdo

1	Introdução	4
2	Parte 1	4
2.1	Perguntas e Respostas	4
2.1.1	Pergunta 1	4
2.1.2	Resposta 1	4
2.1.3	Pergunta 2	5
2.1.4	Resposta 2	5
2.1.5	Pergunta 3	5
2.1.6	Resposta 3	5
2.1.7	Pergunta 4	6
2.1.8	Resposta 4	6
2.1.9	Pergunta 5	7
2.1.10	Resposta 5	7
2.1.11	Pergunta 6	7
2.1.12	Resposta 6	7
2.1.13	Pergunta 7	7
2.1.14	Resposta 7	7
2.1.15	Pergunta 8	8
2.1.16	Resposta 8	8
2.1.17	Pergunta 9	8
2.1.18	Resposta 9	8
2.1.19	Pergunta 10	8
2.1.20	Resposta 10	9
2.1.21	Pergunta 11	9
2.1.22	Resposta 11	9
2.1.23	Pergunta 12	9
2.1.24	Resposta 12	9
2.1.25	Pergunta 13	9
2.1.26	Resposta 13	9
2.1.27	Pergunta 14	10
2.1.28	Resposta 14	10
2.1.29	Pergunta 15	11
2.1.30	Resposta 15	11
2.1.31	Pergunta 16	11
2.1.32	Resposta 16	11
2.1.33	Pergunta 17	11
2.1.34	Resposta 17	12
2.1.35	Pergunta 18	12
2.1.36	Resposta 18	12
2.2	Parte 2	13
2.2.1	Exercício 1	13
2.2.2	Pergunta 1	13
2.2.3	Resposta 1	13

2.2.4	Pergunta 2	13
2.2.5	Resposta 2	13
2.2.6	Pergunta 3	14
2.2.7	Resposta 3	14
2.2.8	Pergunta 4	14
2.2.9	Resposta 4	14
2.2.10	Pergunta 5	15
2.2.11	Resposta 5	15
2.2.12	Exercício 2	15
2.2.13	Pergunta 1	15
2.2.14	Resposta 1	16
2.2.15	Pergunta 2	17
2.2.16	Resposta 2	17
2.2.17	Pergunta 3	18
2.2.18	Resposta 3	18
2.2.19	Pergunta 4	19
2.2.20	Resposta 4	19
2.2.21	Pergunta 5	20
2.2.22	Resposta 5	21
2.2.23	Exercício 3	23
2.2.24	Pergunta 1	23
2.2.25	Resposta 1	23
2.2.26	Pergunta 2	24
2.2.27	Resposta 2	24
2.2.28	Pergunta 3	25
2.2.29	Resposta 3	25

3 Conclusão 27

Listings

Lista de Figuras

1	Topologia Core Parte 1	4
2	Tráfego ICMP enviado e recebido pelo sistema Bela	5
3	Resposta 2 - <i>TTL</i> mínimo de 4 (<i>wireshark</i>)	5
4	Resposta 3 - <i>RTT</i> (<i>terminal do host Bela</i>)	6
5	Cabeçalho ICMP - Resposta 5,6,7 e 8	8
6	Primeiro fragmento do datagrama IP segmentado	10
7	Segundo fragmento do datagrama IP segmentado	10
8	Topologia da Organização LEI-RC	13
9	Conectividade interna de cada Departamentos	14
10	Conectividade entre o <i>router</i> RISP e o portátil Bela	15
11	Tabela de Encaminhamento do router A	16

12	Tabela de Encaminhamento do Portátil Bela	17
13	Lista de processos do <i>router</i> RA	18
14	Rota eliminada por defeito no servidor SA	19
15	Rota eliminada por defeito no servidor SA	20
16	Ping entre SA e cada Departamento	21
17	Nova Tabela de Encaminhamento	22
18	Novo esquema de endereçamento	24
19	Conectividade interna (em cada departamento)	25
20	Conectividade externa (para router RISP)	26

1 Introdução

No âmbito da disciplina de Redes de Computadores, foi proposto ao grupo de trabalho a elaboração e exploração do segundo trabalho prático. Neste trabalho serão manipulados e trabalhados temas como a fragmentação de pacotes, endereçamento e encaminhamento em relação ao IPv4 - *Internet Protocol version 4*. O presente projeto divide-se em duas partes, a primeira é direcionada ao formato e fragmentação de pacotes, por outro lado, a segunda parte é relativa a endereçamento e encaminhamento.

2 Parte 1

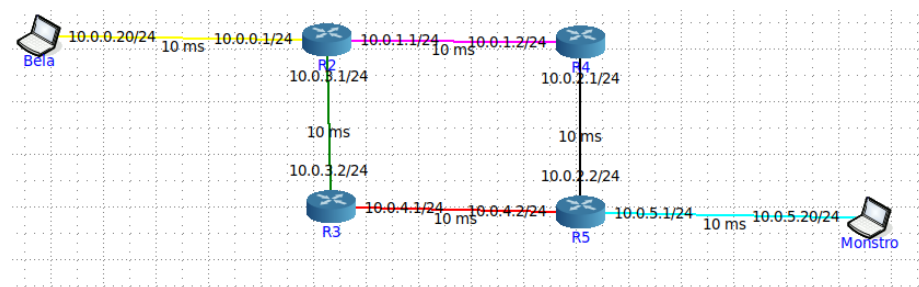


Figura 1: Topologia Core Parte 1

2.1 Perguntas e Respostas

2.1.1 Pergunta 1

”Registre e analise o tráfego ICMP enviado pelo sistema Bela e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.”

2.1.2 Resposta 1

Através do *wireshark* conseguimos analisar o tráfego ICMP enviado e recebido aquando da invocação do comando *traceroute -I* para o endereço IP do Monstro. Assim, ao abrir o *wireshark* no *host* Bela, vemos a presença de diversos pacotes ICMP, dos quais podemos identificar pacotes de *request* e pacotes de *reply*. Além disso, conseguimos realçar um campo do cabeçalho dos ICMP de *request*, sendo o *TTL (Time To Live)*. Por outro lado, os pacotes ICMP de *reply* apresentam um *TTL* ”constante”, com o valor de 61. Assim, podemos afirmar que os resultados obtidos correspondem ao esperado face ao que conhecemos do protocolo ICMP.

```

vcmd
root@Bella:/tmp/pycore.37987/Bella.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) 40.615 ms 40.542 ms 40.539 ms
 2 10.0.1.2 (10.0.1.2) 60.821 ms 60.821 ms 60.820 ms
 3 10.0.3.2 (10.0.3.2) 101.315 ms 101.314 ms 101.313 ms
 4 10.0.5.10 (10.0.5.10) 141.871 ms 141.871 ms 141.869 ms
root@Bella:/tmp/pycore.37987/Bella.conf#

```

Figura 2: Tráfego ICMP enviado e recebido pelo sistema Bela

2.1.3 Pergunta 2

”Qual deve ser o valor inicial mínimo do campo *TTL* para alcançar o servidor Monstro ? Verifique na prática que a sua resposta está correta.”

2.1.4 Resposta 2

Dada a topologia proposta para a realização deste trabalho prático e que se encontra acima apresentada, conseguimos prever que o *host* Monstro irá responder, quando o campo *TTL* for no mínimo 4, face à disposição dos *links* de encaminhamento que ligam os *routers* presentes na topologia. Podemos comprovar este valor mínimo de *TTL*, observando o *wireshark*, no qual percebemos que obtemos resposta, quando este valor é de facto 4.

No.	Time	Source	Destination	Protocol	Length	Info
36	46.782090192	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=1/256, ttl=1 (no response..
37	46.782090586	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=2/512, ttl=1 (no response..
38	46.782090951	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=3/768, ttl=1 (no response..
39	46.782091150	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=4/1024, ttl=2 (no respons..
40	46.782091380	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=5/1280, ttl=2 (no respons..
41	46.782091630	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=6/1536, ttl=2 (no respons..
42	46.782091887	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=7/1792, ttl=3 (no respons..
43	46.782092143	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=8/2048, ttl=3 (no respons..
44	46.782092479	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=9/2304, ttl=3 (no respons..
45	46.782092673	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=10/2560, ttl=4 (reply in ..
46	46.782092915	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=11/2816, ttl=4 (reply in ..
47	46.782093312	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=12/3072, ttl=4 (reply in ..
48	46.782093650	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=13/3328, ttl=5 (reply in ..
49	46.782093907	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=14/3584, ttl=5 (reply in ..
50	46.782094127	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=15/3840, ttl=5 (reply in ..
51	46.782094377	10.0.0.20	10.0.5.20	ICMP	74	Echo (ping) request id=0x001b, seq=16/4096, ttl=6 (reply in ..

Figura 3: Resposta 2 - *TTL* mínimo de 4 (*wireshark*)

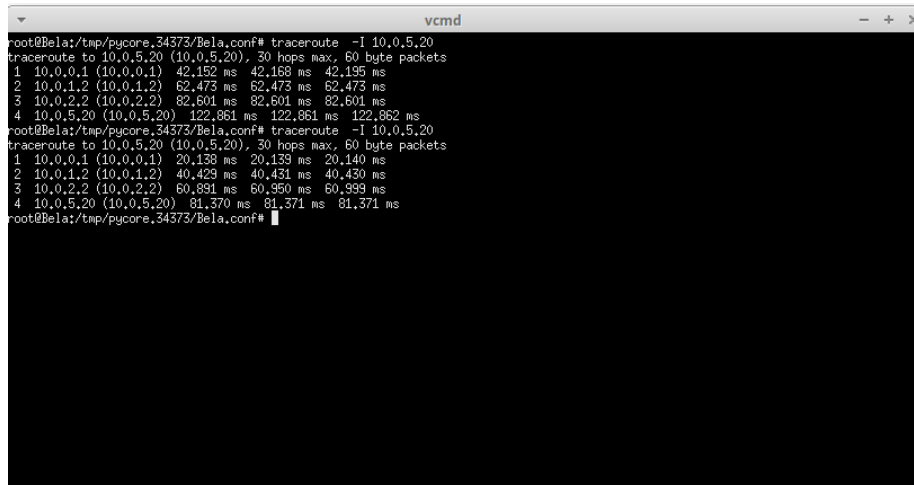
2.1.5 Pergunta 3

”Calcule o valor médio do tempo de ida-e-volta (*RTT* - *Round-Trip Time*) obtido no acesso ao servidor. Para melhorar a média, poderá alterar o número pacotes de prova com a opção -q.”

2.1.6 Resposta 3

Ao invocar o comando *traceroute -I* proposto no enunciado, o mesmo dá informações providenciais no resultado, das quais podemos realçar a resposta à

pergunta. Assim, através da informação prestada pelo comando anteriormente referido, conseguimos concluir que o RTT é de 81.271 ms , podendo comprovar este valor com a imagem abaixo.



```
vcmd
root@Bela:/tmp/pycore.34373/Bela.conf# traceroute -I 10.0.5.20
traceroute to 10.0.5.20 (10.0.5.20), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  42.152 ms  42.158 ms  42.195 ms
 2  10.0.1.2 (10.0.1.2)  62.473 ms  62.473 ms  62.473 ms
 3  10.0.2.2 (10.0.2.2)  82.601 ms  82.601 ms  82.601 ms
 4  10.0.5.20 (10.0.5.20) 122.861 ms 122.861 ms 122.862 ms
root@Bela:/tmp/pycore.34373/Bela.conf# traceroute -I 10.0.5.20
traceroute to 10.0.5.20 (10.0.5.20), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  20.138 ms  20.139 ms  20.140 ms
 2  10.0.1.2 (10.0.1.2)  40.429 ms  40.431 ms  40.430 ms
 3  10.0.2.2 (10.0.2.2)  60.891 ms  60.950 ms  60.999 ms
 4  10.0.5.20 (10.0.5.20) 81.370 ms 81.371 ms 81.371 ms
root@Bela:/tmp/pycore.34373/Bela.conf#
```

Figura 4: Resposta 3 - RTT (terminal do host Bela)

2.1.7 Pergunta 4

”O valor médio do atraso num sentido (One-Way Delay) poderia ser calculado com precisão dividindo o RTT por dois? O que torna difícil o cálculo desta métrica?”

2.1.8 Resposta 4

Convém mencionar aspetos relevantes que justifiquem esta resposta. Assim, para causa efeito, respondendo à consequente questão, ao dividirmos o RTT por dois não iríamos obter uma métrica com precisão, uma vez que nas redes físicas e reais existem ”perturbações” nas mesmas que afetam a métrica em estudo, podendo afetar em apenas um dos sentidos. Além disso, de forma a obtermos um cálculo com precisão desta métrica seriam necessárias diversas informações tais como, tempos de processamento em cada nodo, filas ou tempos de propagação entre links, sendo estas informações difíceis de obter.

Nota: De forma a mitigar a redundância de informação, apenas ilustramos uma imagem que permitem completar as respostas das questões 5,6,7 e 8.

2.1.9 Pergunta 5

”Qual é o endereço IP da interface ativa do seu computador?”

2.1.10 Resposta 5

Através do *wireshark* conseguimos analisar o teor dos vários pacotes ICMP, dos quais podemos identificar os pacotes que nós recebemos na nossa máquina e os que enviámos através das ”*labels*”, *source* e *destination*. Assim, podemos identificar os pacotes ICMP cujo *TTL* não foi suficiente para obter resposta, como sendo os pacotes que enviámos, no qual identificamos o endereço IP da interface ativa do nosso computador através da *label source* previamente mencionada, no qual o mesmo é *192.168.1.3*.

2.1.11 Pergunta 6

”Qual é o valor do campo protocolo? O que permite identificar?”

2.1.12 Resposta 6

Com o auxílio do *wireshark* conseguimos identificar o valor do campo do protocolo, algo notório na imagem abaixo, sendo este valor ICMP(1). Além disto, este valor permite identificar que tipo de protocolo está em causa nos pacotes, permitindo distinguir os protocolos usados, sem ter de analisar toda a estrutura do cabeçalho dos pacotes.

2.1.13 Pergunta 7

”Quanto bytes tem o cabeçalho IPv4? Quanto bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?”

2.1.14 Resposta 7

Com a imagem abaixo, conseguimos analisar o campo relativo ao tamanho do cabeçalho, nomeadamente *Header Length*, tendo este *20 bytes* de tamanho. Além disso, conseguimos perceber o tamanho do pacote através do campo *Total Length*, tendo este *60 bytes* de tamanho. Com isto, conseguimos calcular o *payload* fazendo a subtração do tamanho total pelo tamanho do cabeçalho, sendo o *payload* de *40 bytes* de tamanho.

2.1.15 Pergunta 8

”O datagrama IP foi fragmentado? Justifique.”

2.1.16 Resposta 8

Uma vez analisado a imagem abaixo, uma vez que não há a presença da flag *More Fragments*, conclui-se que o datagrama não foi fragmentado.

```

> Frame 730: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlo1, id 0
> Ethernet II, Src: 18:cc:18:28:54:c9 (18:cc:18:28:54:c9), Dst: ARRISGro_37:3c:4a (44:34:a7:37:3c:4a)
> Internet Protocol Version 4, Src: 192.168.1.3, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0xf5b5 (62901)
  > Flags: 0x0000
  Fragment offset: 0
  > Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x36e8 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.1.3
  Destination: 193.136.9.240
```

Figura 5: Cabeçalho ICMP - Resposta 5,6,7 e 8

2.1.17 Pergunta 9

”Ordene os pacotes capturados de acordo com o endereço IP fonte (e.g., selecionando o cabeçalho da coluna Source), e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à interface da sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.”

2.1.18 Resposta 9

Após observar os vários campos do cabeçalho de cada pacote ICMP, conseguimos perceber o aspeto que varia entre as mensagens protocolares, sendo estes aspetos os campos *Identification* e *TTL*, no qual estes valores são graduais de pacote para pacote.

2.1.19 Pergunta 10

”Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL?”

2.1.20 Resposta 10

Tal como mencionado anteriormente, conseguimos perceber que os campos cujos valores alteram de pacote para pacote, são *Identification* e *TTL*, no qual é visível um padrão crescente após a disposição da ordem crescente das mensagens enviadas.

2.1.21 Pergunta 11

”Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP *TTL exceeded* enviadas ao seu computador. Qual é o valor do campo *TTL*? Esse valor permanece constante para todas as mensagens de resposta ICMP *TTL exceeded* enviados ao seu host? Porquê?”

2.1.22 Resposta 11

O valor do campo *TTL*, das mensagens protocolares ICMP cujo *TTL* foi excedido, varia de facto de pacote para pacote, sendo que os valores variam de 253 a 255, sendo que por este motivo, este valor não é constante. A razão disto é o facto que quando as mensagens protocolares cujo *TTL* foi excedido admitem o valor de 256, pelo que ao serem redireccionados a nós, este irá ser diminuído por causa do número de saltos efetuados até chegar a nós.

2.1.23 Pergunta 12

”Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?”

2.1.24 Resposta 12

Ao analisar o cabeçalho da primeira mensagem protocolar ICMP, percebemos que o valor que indica o tamanho total da mesma, é de 1044 *bytes*. Uma vez que definimos um tamanho das mensagens protocolar de 4004 *bytes*, a mesma mensagem teve de ser fragmentada.

2.1.25 Pergunta 13

”Imprima o primeiro fragmento do datagrama IP segmentado. Que informação no cabeçalho indica que o datagrama foi fragmentado? Que informação no cabeçalho IP indica que se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP?”

2.1.26 Resposta 13

A informação providencial do cabeçalho que nos indica que o pacote foi fragmentado, é o facto do campo *Fragment offset* não ser nulo e além disso a presença

da flag *More Fragments*. A informação que permite perceber que se trata do primeiro fragmento é o valor do campo *Fragment offset* estar a 0, indicando início do datagrama. Ao analisar os consequentes fragmentos pertencentes ao mesmo datagrama (através do campo *Identification*, ao somar o valor do campo *Total Length* de cada fragmento, o tamanho deste datagrama é de 1500 *bytes*.

```

> Frame 90: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp2s0, id 0
> Ethernet II, Src: Chongqin_83:a9:a9 (28:cd:c4:83:a9:a9), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.100.163, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x3ca0 (15520)
    > Flags: 0x2000, More fragments
      Fragment offset: 0
    > Time to live: 1
      Protocol: ICMP (1)
      Header checksum: 0x7b4b [validation disabled]
      [Header checksum status: Unverified]
      Source: 172.26.100.163
      Destination: 193.136.9.240
      Reassembled IPv4 in frame: 92
  > Data (1480 bytes)

```

Figura 6: Primeiro fragmento do datagrama IP segmentado

2.1.27 Pergunta 14

”Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do 1º fragmento? Há mais fragmentos? O que nos permite afirmar isso?”

2.1.28 Resposta 14

A informação do cabeçalho que permite identificar que não se trata do 1º fragmento é o valor do *offset* que se encontra com um valor não nulo. Através da flag *More Fragments* conseguimos perceber que há mais fragmentos.

```

> Frame 91: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface wlp2s0, id 0
> Ethernet II, Src: Chongqin_83:a9:a9 (28:cd:c4:83:a9:a9), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.100.163, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x3ca0 (15520)
    > Flags: 0x20b9, More fragments
      Fragment offset: 1480
    > Time to live: 1
      Protocol: ICMP (1)
      Header checksum: 0x7a92 [validation disabled]
      [Header checksum status: Unverified]
      Source: 172.26.100.163
      Destination: 193.136.9.240
      Reassembled IPv4 in frame: 92
  > Data (1480 bytes)

```

Figura 7: Segundo fragmento do datagrama IP segmentado

2.1.29 Pergunta 15

”Quantos fragmentos foram criados a partir do datagrama original? ”

2.1.30 Resposta 15

Ao analisar a informação prestada pelo *wireshark*, vemos que o número de fragmentos corresponde a 3, através do campo *Identification* ter o mesmo valor em cada três pacotes.

2.1.31 Pergunta 16

”Indique, resumindo, os campos que mudam no cabeçalho IP entre os diferentes fragmentos, e explique a forma como essa informação permite reconstruir o datagrama original.”

2.1.32 Resposta 16

Após uma breve análise entre fragmentos de datagramas em comum, os campos que mudam no cabeçalho IP são:

- *offset* do fragmento, no qual o valor do mesmo permite perceber qual o *offset* em que deve estar o fragmento correspondente no datagrama original.
- tamanho total do fragmento, no qual nos permite saber quantos *bytes* cada fragmento ocupa/tem de tamanho.
- *flag More Fragments*, no qual este campo permite-nos perceber qual é o último fragmento, distinguindo este dos restantes.
- *Identification*, no qual este campo permite-nos perceber quais os fragmentos que pertencem ao mesmo datagrama.

2.1.33 Pergunta 17

”Verifique o processo de fragmentação através de um processo de cálculo.”

2.1.34 Resposta 17

Precisamos do valor do *offset* do fragmento de modo a saber onde começa cada fragmento do datagrama original. Posto isto, através do tamanho de cada fragmento conseguimos construir o datagrama original que fora fragmentado, com ajuda da flag *More Fragments*, bem como do campo *Identification* relativo ao datagrama. Assim temos, para o datagrama temos o seguinte cálculo:

- tamanho do pacote ICMP / tamanho máximo de *payload* por IP > 1

Uma vez que temos um datagrama cujo tamanho pré-definido foi 4004 *bytes*, temos o seguinte resultado:

- $4004 / 1480 \approx 2,70 > 1$

Pelo que percebemos que temos que fragmentar o datagrama original em 3 fragmentos.

2.1.35 Pergunta 18

”Escreva uma expressão lógica que permita detetar o último fragmento correspondente ao datagrama original.”

2.1.36 Resposta 18

Para a expressão lógica em questão, iremos assumir o valor do campo *Identification* do datagrama original como sendo *ID*. Assim, temos a seguinte expressão lógica:

```
MOREFRAGMENTS == false && IDENTIFICATION == ID && OFFSET != 0
```

2.2 Parte 2

2.2.1 Exercício 1

2.2.2 Pergunta 1

”Indique que endereços IP e máscaras de rede foram atribuídos pelo CORE a cada equipamento. Para simplificar, pode incluir uma imagem que ilustre de forma clara a topologia definida e o endereçamento usado.”

2.2.3 Resposta 1

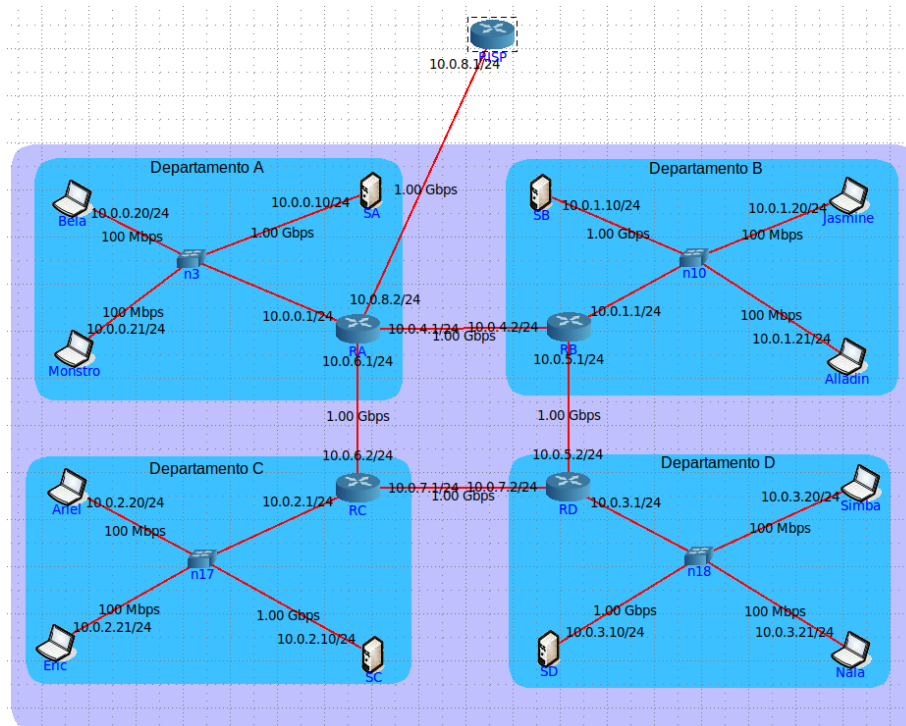


Figura 8: Topologia da Organização LEI-RC

2.2.4 Pergunta 2

”Tratam-se de endereços públicos ou privados? Porquê?”

2.2.5 Resposta 2

Ao analisar todos os endereços atribuídos pelo *Core*, uma vez que, além de serem todos classificados como classe A, como se encontram no intervalo de 10.0.0.0 a

10.255.255.255, correspondem a endereços privados.

2.2.6 Pergunta 3

”Porque razão não é atribuído um endereço IP aos switches?”

2.2.7 Resposta 3

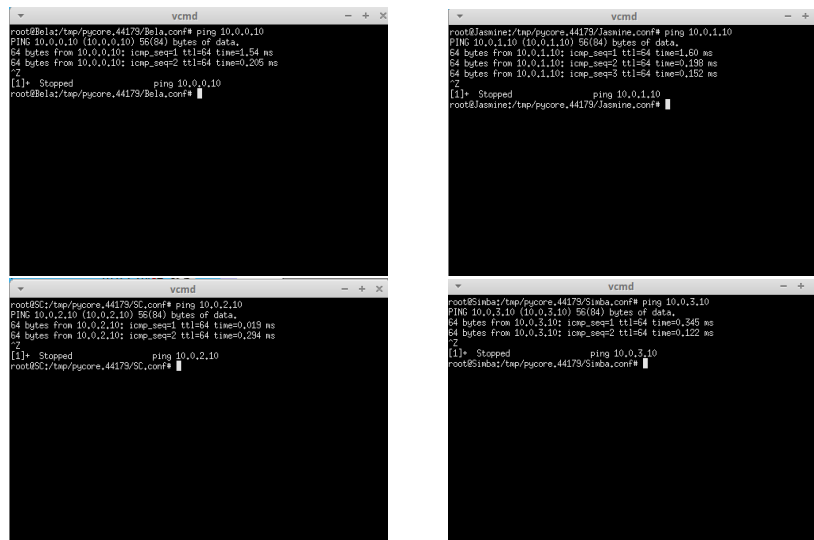
É de ressaltar que os endereços IP são endereços lógicos, pelo que os *switches* funcionam com endereços físicos. Além disto, os *switches* servem para a conexão entre entidades na vertente dos dados, não pertencendo à camada de rede.

2.2.8 Pergunta 4

”Usando o comando ping certifique-se que existe conectividade IP interna a cada departamento (e.g. entre um laptop e o servidor respetivo).”

2.2.9 Resposta 4

De modo a analisar a conectividade de cada departamento fizemos uso do comando *ping* num dos portáteis de cada departamento para cada servidor do respetivo departamento.



```
root@bela:/tmp/pycore.44179/bela.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=1.54 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.205 ms
^C
[1]- Stopped ping 10.0.0.10
root@bela:/tmp/pycore.44179/bela.conf#

root@jasmine:/tmp/pycore.44179/jasmine.conf# ping 10.0.1.10
PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data:
64 bytes from 10.0.1.10: icmp_seq=1 ttl=64 time=1.50 ms
64 bytes from 10.0.1.10: icmp_seq=2 ttl=64 time=0.188 ms
64 bytes from 10.0.1.10: icmp_seq=3 ttl=64 time=0.152 ms
^C
[1]- Stopped ping 10.0.1.10
root@jasmine:/tmp/pycore.44179/jasmine.conf#

root@SC:/tmp/pycore.44179/SC.conf# ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data:
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=0.619 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=64 time=0.294 ms
^C
[1]- Stopped ping 10.0.2.10
root@SC:/tmp/pycore.44179/SC.conf#

root@Sinba2:/tmp/pycore.44179/Sinba2.conf# ping 10.0.3.10
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data:
64 bytes from 10.0.3.10: icmp_seq=1 ttl=64 time=0.345 ms
64 bytes from 10.0.3.10: icmp_seq=2 ttl=64 time=0.122 ms
^C
[1]- Stopped ping 10.0.3.10
root@Sinba2:/tmp/pycore.44179/Sinba2.conf#
```

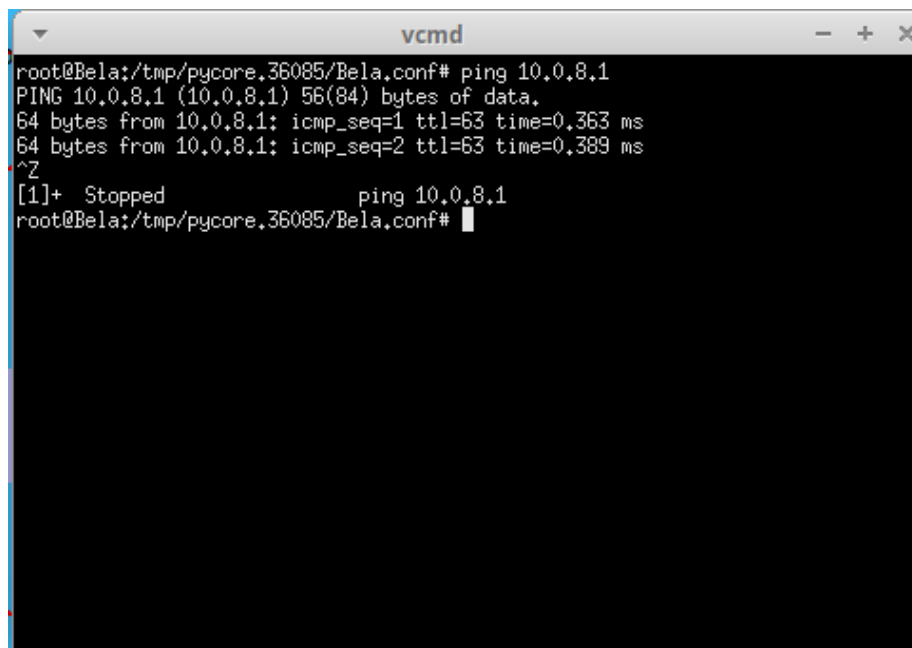
Figura 9: Conectividade interna de cada Departamentos

2.2.10 Pergunta 5

”Verifique se existe conectividade IP do portátil Bela para o router de acesso RISP.”

2.2.11 Resposta 5

Recorrendo, novamente, ao comando *ping* conseguimos verificar a conectividade entre o router RISP e o portátil Bela.



```
vcmd
root@Bela:/tmp/pycore.36085/Bela.conf# ping 10.0.8.1
PING 10.0.8.1 (10.0.8.1) 56(84) bytes of data:
64 bytes from 10.0.8.1: icmp_seq=1 ttl=63 time=0.363 ms
64 bytes from 10.0.8.1: icmp_seq=2 ttl=63 time=0.389 ms
^Z
[1]+  Stopped                  ping 10.0.8.1
root@Bela:/tmp/pycore.36085/Bela.conf#
```

Figura 10: Conectividade entre o *router* RISP e o portátil Bela

2.2.12 Exercício 2

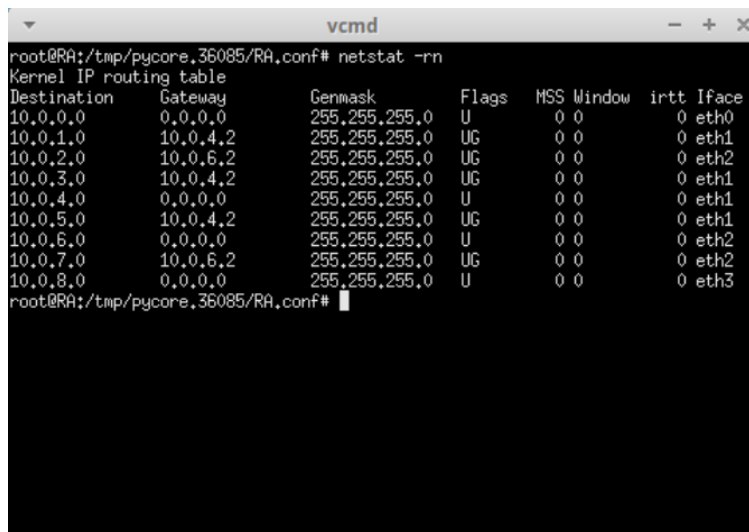
”Para o router RA e o portátil Bela:”

2.2.13 Pergunta 1

”Execute o comando *netstat -rn* por forma a poder consultar a tabela de encaminhamento unicast (IPv4). Inclua no seu relatório as tabelas de encaminhamento obtidas; interprete as várias entradas de cada tabela. Se necessário, consulte o manual respetivo (*man netstat*).”

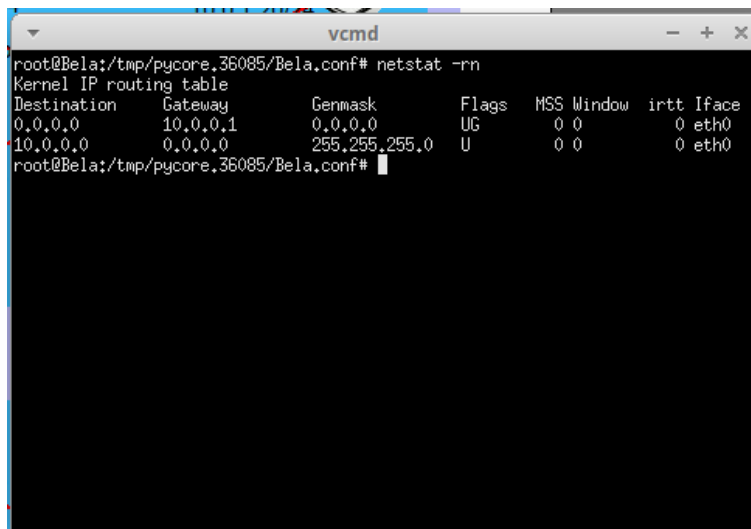
2.2.14 Resposta 1

A coluna de endereços IP destino indica para cada IP destino existe uma saída do router (*gateway*). A coluna Genmask refere-se à máscara utilizada, no caso presente verificamos que a máscara é 255.255.255.0. As flags por sua vez identificam se a rede é global ou local. Por fim, a ultima coluna permite identificar a interface de saída da máquina local.



```
vcmd
root@RA:/tmp/pycore.36085/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 10.0.4.2 255.255.255.0 UG 0 0 0 eth1
10.0.2.0 10.0.6.2 255.255.255.0 UG 0 0 0 eth2
10.0.3.0 10.0.4.2 255.255.255.0 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 10.0.4.2 255.255.255.0 UG 0 0 0 eth1
10.0.6.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.7.0 10.0.6.2 255.255.255.0 UG 0 0 0 eth2
10.0.8.0 0.0.0.0 255.255.255.0 U 0 0 0 eth3
root@RA:/tmp/pycore.36085/RA.conf#
```

Figura 11: Tabela de Encaminhamento do router A



```
root@Bela:/tmp/pycore.36085/Bela.conf# netstat -rn
Kernel IP routing table
Destination    Gateway        Genmask         Flags   MSS Window  irtt Iface
0.0.0.0        10.0.0.1       0.0.0.0         UG      0  0        0 eth0
10.0.0.0       0.0.0.0        255.255.255.0   U       0  0        0 eth0
root@Bela:/tmp/pycore.36085/Bela.conf#
```

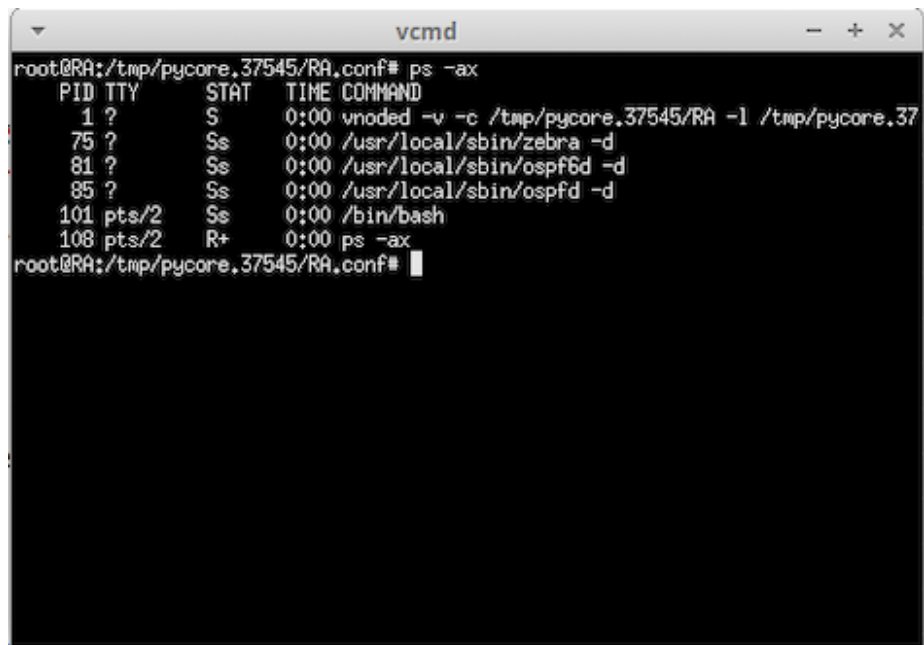
Figura 12: Tabela de Encaminhamento do Portátil Bela

2.2.15 Pergunta 2

”Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correr em cada sistema, por exemplo, ps -ax ou equivalente).”

2.2.16 Resposta 2

Ao executar o comando ps -ax é possível verificar que está a ser executado o protocolo OSPF6, logo conseguimos concluir que se trata de um encaminhamento dinâmico.



```
vcmd
root@RA:/tmp/pycore.37545/RA.conf# ps -ax
  PID TTY          STAT       TIME COMMAND
    1 ?           S            0:00 vnc -c /tmp/pycore.37545/RA -l /tmp/pycore.37
   75 ?          Ss           0:00 /usr/local/sbin/zebra -d
   81 ?          Ss           0:00 /usr/local/sbin/ospf6d -d
   85 ?          Ss           0:00 /usr/local/sbin/ospf6d -d
  101 pts/2      Ss           0:00 /bin/bash
  108 pts/2      R+           0:00 ps -ax
root@RA:/tmp/pycore.37545/RA.conf#
```

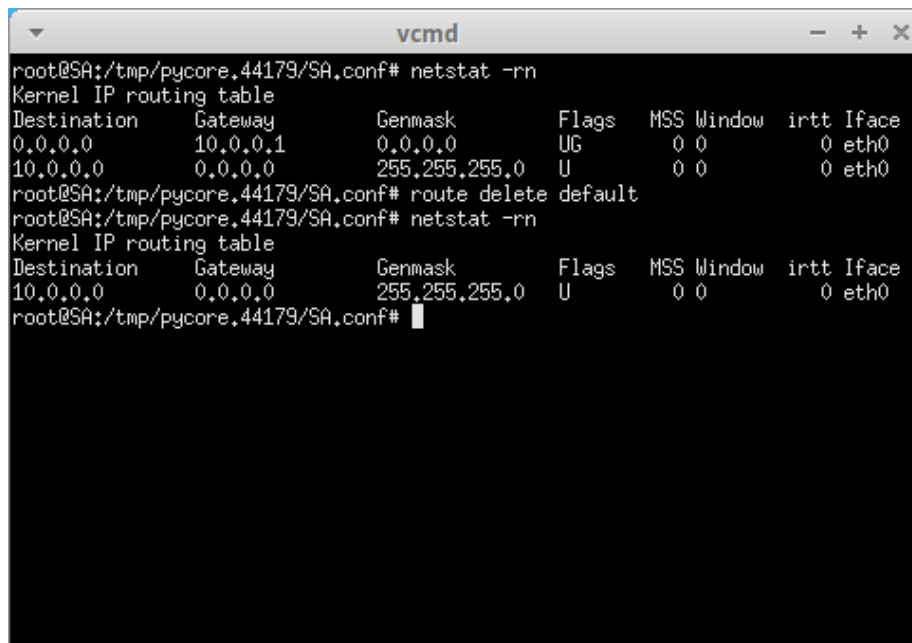
Figura 13: Lista de processos do *router* RA

2.2.17 Pergunta 3

”Admita que, por questões administrativas, a rota por defeito (0.0.0.0 ou default) deve ser retirada definitivamente da tabela de encaminhamento do servidor SA. Use o comando `route delete` para o efeito. Que implicações tem esta medida para os utilizadores da LEI-RC que acedem ao servidor. Justifique.”

2.2.18 Resposta 3

O servidor SA, após a eliminação da rota *default*, irá encaminhar para o endereço IP destino 10.0.0.0, ou seja, para a sua rede local. Com isto o servidor SA não poderá encaminhar tráfego de pacotes para outras entidades de outras redes.



```
root@SA:/tmp/pycore.44179/SA.conf# netstat -rn
Kernel IP routing table
Destination      Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0          10.0.0.1        0.0.0.0         UG      0 0        0 eth0
10.0.0.0         0.0.0.0         255.255.255.0   U       0 0        0 eth0
root@SA:/tmp/pycore.44179/SA.conf# route delete default
root@SA:/tmp/pycore.44179/SA.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
root@SA:/tmp/pycore.44179/SA.conf#
```

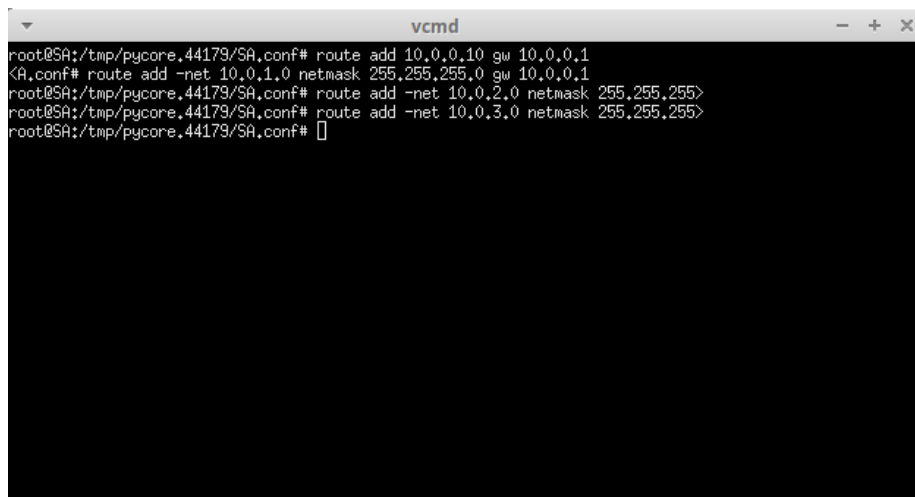
Figura 14: Rota eliminada por defeito no servidor SA

2.2.19 Pergunta 4

”Não volte a repor a rota por defeito. Adicione todas as rotas estáticas necessárias para restaurar a conectividade para o servidor SA, por forma a contornar a restrição imposta na alínea c). Utilize para o efeito o comando route add e registe os comandos que usou.”

2.2.20 Resposta 4

Uma vez imposta a restrição supramencionada, teremos agora de restaurar todas as conectividades necessárias para repor o estado de ligação anterior.



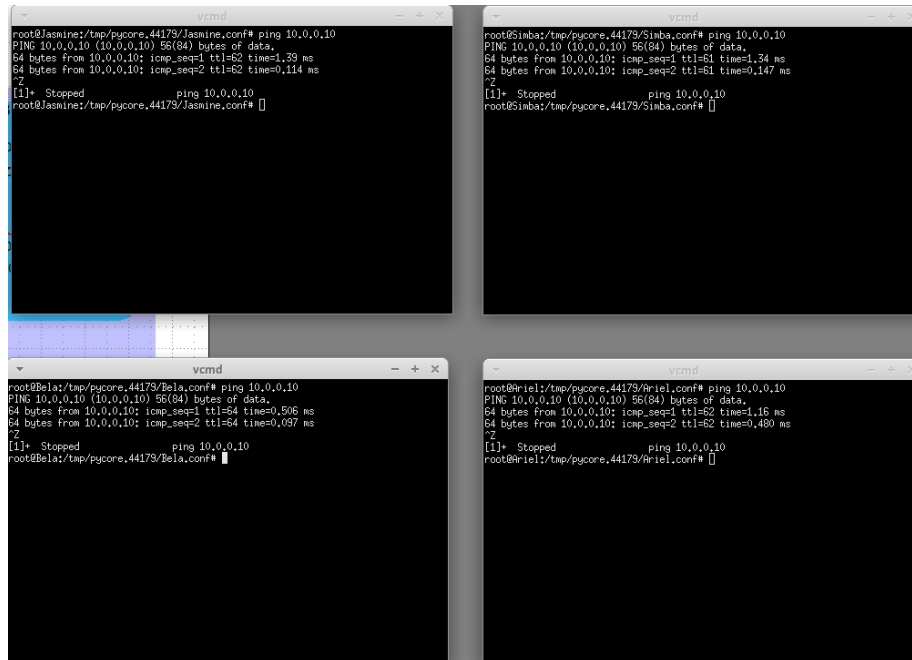
```
vcmd
root@SA:/tmp/pycore,44179/SA.conf# route add 10.0.0.10 gw 10.0.0.1
<A.conf# route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.0.1
root@SA:/tmp/pycore,44179/SA.conf# route add -net 10.0.2.0 netmask 255.255.255.0
root@SA:/tmp/pycore,44179/SA.conf# route add -net 10.0.3.0 netmask 255.255.255.0
root@SA:/tmp/pycore,44179/SA.conf#
```

Figura 15: Rota eliminada por defeito no servidor SA

2.2.21 Pergunta 5

”Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível, utilizando para o efeito o comando ping. Registe a nova tabela de encaminhamento do servidor.”

2.2.22 Resposta 5



The figure displays four terminal windows, each representing a different department (Jasmine, Sinba, Bela, and Ariel) performing a ping test to the IP address 10.0.0.10. Each window shows the command being executed, the results of the ping (including TTL and time), and the final status of the test.

```
root@Jasmine:/tmp/pycore.44179/Jasmine.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=62 time=1.39 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=62 time=0.114 ms
^C
[1]+  Stopped                  ping 10.0.0.10
root@Jasmine:/tmp/pycore.44179/Jasmine.conf#
```

```
root@Sinba:/tmp/pycore.44179/Sinba.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=61 time=1.34 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=61 time=0.147 ms
^C
[1]+  Stopped                  ping 10.0.0.10
root@Sinba:/tmp/pycore.44179/Sinba.conf#
```

```
root@Bela:/tmp/pycore.44179/Bela.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.506 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.097 ms
^C
[1]+  Stopped                  ping 10.0.0.10
root@Bela:/tmp/pycore.44179/Bela.conf#
```

```
root@Ariel:/tmp/pycore.44179/Ariel.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=62 time=1.16 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=62 time=0.480 ms
^C
[1]+  Stopped                  ping 10.0.0.10
root@Ariel:/tmp/pycore.44179/Ariel.conf#
```

Figura 16: Ping entre SA e cada Departamento

```
vcmd
root@SA:/tmp/pycore.44179/SA.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
10.0.0.10 10.0.0.1 255.255.255.255 UGH 0 0 0 eth0
10.0.1.0 10.0.0.1 255.255.255.0 UG 0 0 0 eth0
10.0.2.0 10.0.0.1 255.255.255.0 UG 0 0 0 eth0
10.0.3.0 10.0.0.1 255.255.255.0 UG 0 0 0 eth0
root@SA:/tmp/pycore.44179/SA.conf#
```

Figura 17: Nova Tabela de Encaminhamento

2.2.23 Exercício 3

2.2.24 Pergunta 1

”Considere que dispõe apenas do endereço de rede IP 192.168.XXX.128/25, em que XXX é o decimal correspondendo ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo as redes de acesso externo e backbone inalteradas), sabendo que o número de departamentos pode vir a aumentar no curto prazo. Atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis. Justifique as opções tomadas no planeamento.”

2.2.25 Resposta 1

É de notar que estamos a trabalhar com endereços IPv4 no qual temos um endereço de rede que possui 25 *bits* para rede, pelo que nos sobram 7 *bits* para manusear e aplicar o novo esquema de endereçamento. Atualmente temos um total de 4 departamentos, o que nos permitiria usar um menor número de bits do que o realmente é usado para representar os departamentos, mas tal como referido no enunciado, dentro de um curto espaço de tempo serão introduzidos novos departamentos, o que iria criar um conflito devido à falta de bits para os representar. Deste modo, o grupo de trabalho definiu que um total de 4 bits para representar os departamentos seria o mais adequado devido ao número de departamentos que será possível adicionar à rede com esta representação, isto é, será possível representar um total de 16 departamentos, sendo que 4 já estão na rede, logo permitirá adicionar 12 departamentos. Por outro lado, esta configuração faz com que usemos 3 bits para a representação dos hosts, o que no ponto de vista do grupo de trabalho torna a nossa decisão melhorada devido à possibilidade de aumentar o número de hosts por cada departamento, sendo possível representar 6 hosts por cada departamento. Por fim, obtemos uma máscara de 29 bits, dos quais 4 representam a sub-rede.

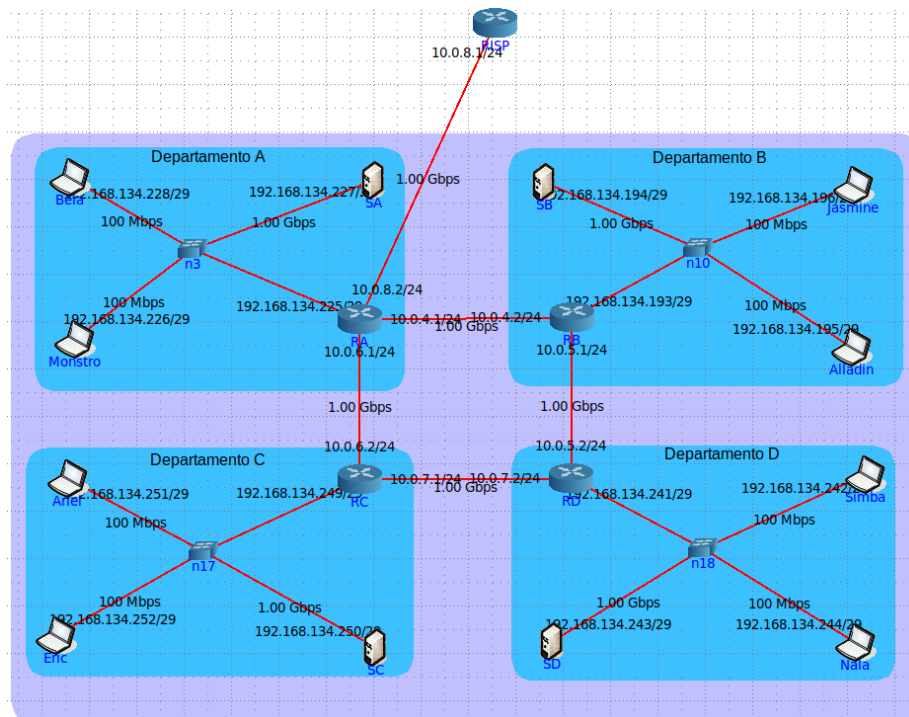


Figura 18: Novo esquema de endereçamento

2.2.26 Pergunta 2

”Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Quantos prefixos de sub-rede ficam disponíveis para uso futuro? Justifique.”

2.2.27 Resposta 2

Para a identificação da máscara concluímos que como seria necessário 25 bits para a rede, mais 4 bits para sub redes e 3 bits para hosts, teríamos uma disposição final de os 3 primeiros octetos serem 11111111 e o ultimo octeto ser 11111000:

- Máscara de rede (binário): 11111111.11111111.11111111.11111000.
- Máscara de rede (decimal em notação CIDR) 255.255.255.248/29.

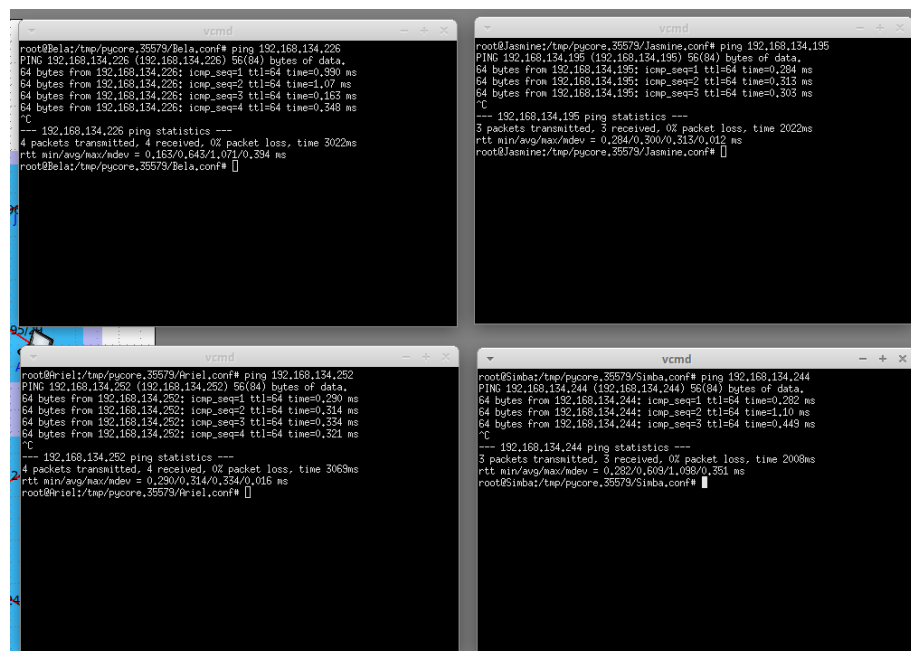
Temos um total de 6 hosts para cada sub rede, visto termos definido 3 bits para representar os hosts, e não podemos representar os endereços da rede e do broadcast sendo então necessário subtrair 2, então o número de hosts é $2^3 - 2 = 6$. Por fim, estão disponíveis 16 sub redes, pois definimos um total de 4 bits para as mesmas, 2^4 .

2.2.28 Pergunta 3

”Verifique e garanta que a conectividade IP interna na rede local LEI-RC é mantida. No caso de não existência de conectividade, reveja a atribuição de endereços efetuada e eventuais erros de encaminhamento por forma a realizar as correções necessárias. Explique como procedeu.”

2.2.29 Resposta 3

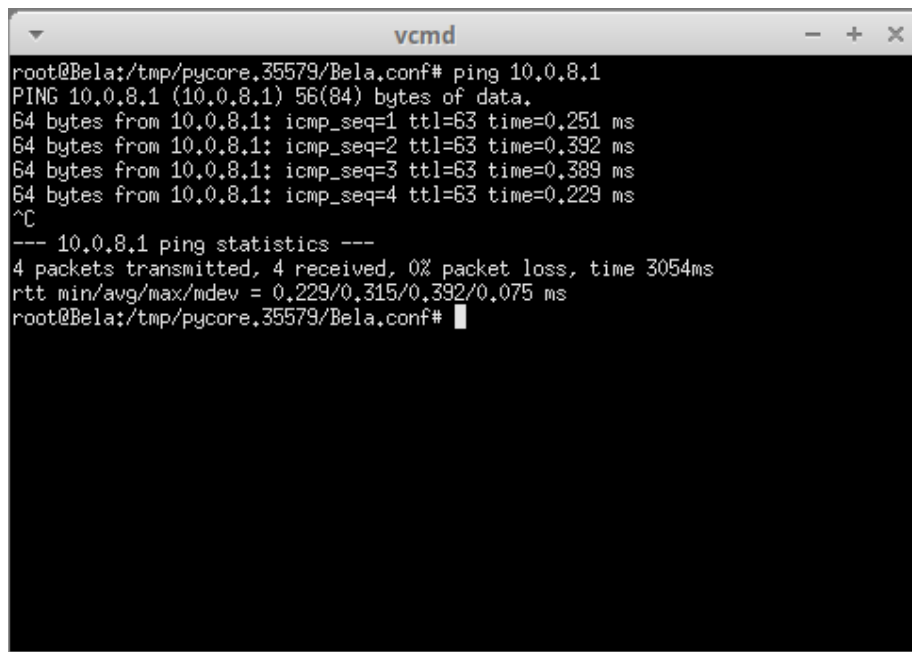
Procedemos ao teste de conexão IP interna na rede local LEI-RC, através da execução do comando *ping* internamente a cada subrede já definida, bem como entre as subredes.



The figure displays four terminal windows, each representing a different department in a network. Each window shows the execution of a `ping` command to a specific IP address, followed by the resulting output showing packet transmission statistics and round-trip times.

- Top Left Window (Bela):** The user is at `root@Bela:/tmp/pycore.35579/Bela.conf#` and pings `192.168.134.226`. The output shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3022ms.
- Top Right Window (Jasmine):** The user is at `root@Jasmine:/tmp/pycore.35579/Jasmine.conf#` and pings `192.168.134.195`. The output shows 3 packets transmitted, 3 received, 0% packet loss, and a time of 2022ms.
- Bottom Left Window (Iriel):** The user is at `root@Iriel:/tmp/pycore.35579/Iriel.conf#` and pings `192.168.134.252`. The output shows 4 packets transmitted, 4 received, 0% packet loss, and a time of 3063ms.
- Bottom Right Window (Simba):** The user is at `root@Simba:/tmp/pycore.35579/Simba.conf#` and pings `192.168.134.244`. The output shows 3 packets transmitted, 3 received, 0% packet loss, and a time of 2008ms.

Figura 19: Conectividade interna (em cada departamento)



```
root@Bela:/tmp/pycore.35579/Bela.conf# ping 10.0.8.1
PING 10.0.8.1 (10.0.8.1) 56(84) bytes of data:
64 bytes from 10.0.8.1: icmp_seq=1 ttl=63 time=0.251 ms
64 bytes from 10.0.8.1: icmp_seq=2 ttl=63 time=0.392 ms
64 bytes from 10.0.8.1: icmp_seq=3 ttl=63 time=0.389 ms
64 bytes from 10.0.8.1: icmp_seq=4 ttl=63 time=0.229 ms
^C
--- 10.0.8.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.229/0.315/0.392/0.075 ms
root@Bela:/tmp/pycore.35579/Bela.conf#
```

Figura 20: Conectividade externa (para router RISP)

3 Conclusão

Com a realização deste trabalho foi possível consolidar toda a matéria lecionada nas aulas teóricas e práticas, uma vez que nos permitiu estudar e praticar temas como endereçamento, encaminhamento e datagramas IP. Foi possível consolidar o uso e manipulação de um software de gestão e manipulação de redes, nomeadamente o CORE e o wireshark. Em suma, concluímos que este trabalho prático constitui uma mais valia para o nosso conhecimento visto que permitiu-nos adquirir boas bases para trabalhos futuros, quer académicos ou profissionais. Consideramos também que em relação ao pretendido pelos docentes da cadeira para este segundo trabalho, conseguimos uma boa resolução e uma boa interpretação do enunciado.