

# RC TP2 PL54

Artur Luís

Carlos Pina

Luís Ferreira

March 2023

# Índice

<b>1</b>	<b>Primeira Parte - Resolução</b>	<b>3</b>
1.1	Questão 1 . . . . .	3
1.1.1	Alínea A . . . . .	4
1.1.2	Alínea B . . . . .	5
1.1.3	Alínea C . . . . .	5
1.1.4	Alínea D . . . . .	5
1.2	Questão 2 . . . . .	6
1.2.1	Alínea A . . . . .	6
1.2.2	Alínea B . . . . .	6
1.2.3	Alínea C . . . . .	7
1.2.4	Alínea D . . . . .	7
1.2.5	Alínea E . . . . .	8
1.2.6	Alínea F . . . . .	10
1.2.7	Alínea G . . . . .	10
1.2.8	Alínea H . . . . .	12
1.3	Questão 3 . . . . .	13
1.3.1	Alínea A . . . . .	13
1.3.2	Alínea B . . . . .	14
1.3.3	Alínea C . . . . .	15
1.3.4	Alínea D . . . . .	15
1.3.5	Alínea E . . . . .	15
1.3.6	Alínea F . . . . .	15
1.3.7	Alínea G . . . . .	16
1.3.8	Alínea H . . . . .	16
1.3.9	Alínea I . . . . .	16
1.3.10	Alínea J . . . . .	17
<b>2</b>	<b>Segunda Parte - Resolução</b>	<b>18</b>
2.1	Questão 1 . . . . .	19
2.1.1	Alínea A . . . . .	19
2.1.2	Alínea B . . . . .	20
2.1.3	Alínea C . . . . .	21
2.1.4	Alínea D . . . . .	26
2.1.5	Alínea E . . . . .	27
2.1.6	Alínea F . . . . .	27
2.1.7	Alínea G . . . . .	27
2.2	Questão 2 . . . . .	28
2.2.1	Alínea A . . . . .	28
2.2.2	Alínea B . . . . .	28
2.2.3	Alínea C . . . . .	29
2.3	Questão 3 . . . . .	29
2.3.1	Alínea A . . . . .	29
2.3.2	Alínea B . . . . .	29
2.3.3	Alínea C . . . . .	29

# 1 Primeira Parte - Resolução

## 1.1 Questão 1

“Prepare uma topologia CORE para verificar o comportamento do traceroute. Na topologia deve existir: um host (pc) cliente designado Lost, cujo router de acesso é RA1; o router RA1 está simultaneamente ligado a dois routers no core da rede RC1 e RC2; estes estão conectados a um router de acesso RA2, que por sua vez, se liga a um host (servidor) designado Found. Ajuste o nome dos equipamentos atribuídos por defeito para o enunciado. Apenas nas ligações (links) da rede de core, estabeleça um tempo de propagação de 15 ms. Após ativar a topologia, note que pode não existir conectividade IP imediata entre Lost e Found até que o anúncio de rotas entre routers estabilize.”

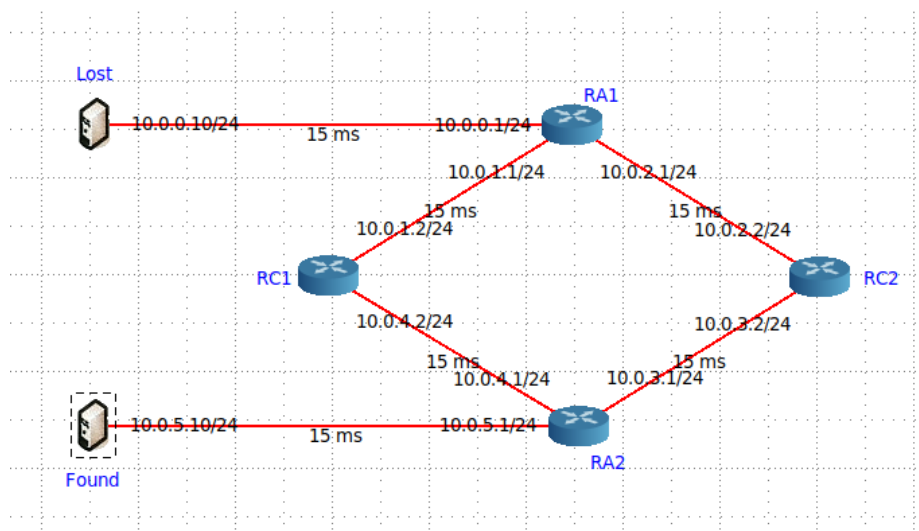


Figure 1: Modelo

### 1.1.1 Alínea A

“Ative o Wireshark no host Lost. Numa shell de Lost execute o comando traceroute -I para o endereço IP do Found. Registe e analise o tráfego ICMP enviado pelo sistema Lost e o tráfego ICMP recebido como resposta.

Explique os resultados obtidos tendo em conta o princípio de funcionamento do traceroute.”

```
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) 0.183 ms 0.024 ms 0.015 ms
 2 10.0.1.2 (10.0.1.2) 30.908 ms 30.892 ms 30.883 ms
 3 10.0.3.2 (10.0.3.2) 62.395 ms 62.388 ms 62.379 ms
 4 10.0.5.10 (10.0.5.10) 62.370 ms 62.361 ms 62.353 ms
```

Figure 2: Resultado da execução do comando

```
48 17.399489619 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=1/256, ttl=1 (no response found!)
49 17.399524771 10.0.0.1 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
50 17.399552929 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=2/512, ttl=1 (no response found!)
51 17.399594142 10.0.0.1 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
52 17.399577592 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=3/768, ttl=1 (no response found!)
53 17.399588021 10.0.0.1 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
54 17.399598790 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=4/1024, ttl=2 (no response found!)
55 17.399619980 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=5/1280, ttl=2 (no response found!)
56 17.399631332 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=6/1536, ttl=2 (no response found!)
57 17.399643149 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=7/1792, ttl=3 (no response found!)
58 17.399653917 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=8/2048, ttl=3 (no response found!)
59 17.399663857 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=9/2304, ttl=3 (no response found!)
60 17.399675823 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=10/2560, ttl=4 (reply in 79)
61 17.399686282 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=11/2816, ttl=4 (reply in 80)
62 17.399701421 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=12/3072, ttl=4 (reply in 81)
63 17.399713130 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=13/3328, ttl=5 (reply in 82)
64 17.399723155 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=14/3584, ttl=5 (reply in 83)
65 17.399732982 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=15/3840, ttl=5 (reply in 84)
66 17.399744491 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=16/4096, ttl=6 (reply in 85)
67 17.401079537 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=17/4352, ttl=6 (reply in 86)
68 17.401096603 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=18/4608, ttl=6 (reply in 87)
69 17.401109765 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=19/4864, ttl=7 (reply in 88)
70 17.439407422 10.0.1.2 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
71 17.439506877 10.0.1.2 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
72 17.439509086 10.0.1.2 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
73 17.431161804 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=20/5120, ttl=7 (reply in 89)
74 17.431193192 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=21/5376, ttl=7 (reply in 90)
75 17.431208252 10.0.0.20 10.0.5.10 ICMP 74 Echo (ping) request id=0x001b, seq=22/5632, ttl=8 (reply in 91)
76 17.462038945 10.0.3.2 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
77 17.462038886 10.0.3.2 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
78 17.462038928 10.0.3.2 10.0.0.20 ICMP 102 Time-to-live exceeded (Time to live exceeded in transit)
79 17.462040723 10.0.5.10 10.0.0.20 ICMP 74 Echo (ping) reply id=0x001b, seq=10/2560, ttl=61 (request in 60)
80 17.462042494 10.0.5.10 10.0.0.20 ICMP 74 Echo (ping) reply id=0x001b, seq=11/2816, ttl=61 (request in 61)
81 17.462044914 10.0.5.10 10.0.0.20 ICMP 74 Echo (ping) reply id=0x001b, seq=12/3072, ttl=61 (request in 62)
```

Figure 3:

- O traceroute é um comando utilizado para descobrir quantos dispositivos estão entre o destino e a origem do comando. Inicialmente, a origem envia um pacote com TTL (time to live) de 1. Este TTL é decrementado a cada vez que atinge um novo dispositivo, e quando chega a 0 esse dispositivo emite uma mensagem ICMP para a origem, que então volta a enviar um pacote com um TTL maior (se enviou previamente com TTL = 1, agora vai enviar com TTL = 2). Quando voltar a 0, o mesmo processo ocorre, o local do último salto envia uma mensagem para a origem e esta volta a enviar um pacote com TTL maior. Este processo ocorre até ter sido atingido o destino. Como podemos observar pela figura 3, vemos que as mensagens ICMP recebidas têm como objetivo avisar que os pacotes não conseguiram ser entregues(TTL chegou a 0).

### 1.1.2 Alínea B

“Qual deve ser o valor inicial mínimo do campo TTL para alcançar o servidor Found?”

- O valor inicial mínimo para alcançar o servidor é 4. Podemos observar na Figura 3 que os pacotes enviados com TTL = 1, TTL = 2 e TTL = 3 nunca chegam a devolver uma resposta, esta aparece apenas quando o TTL é igual ou superior a 4, concluindo assim que esse é o TTL mínimo para alcançar o servidor.

### 1.1.3 Alínea C

“Calcule o valor médio do tempo de ida-e-volta (RTT - Round-Trip Time) obtido no acesso ao servidor. Por modo a obter uma média mais confiável, poderá alterar o número pacotes de prova com a opção -q.”

```
tracert -q 10 to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.078 ms 0.021 ms 0.015 ms 0.014 ms 0.015 ms 0.015 ms * * * *
 2 10.0.1.2 (10.0.1.2) 30.363 ms 30.344 ms 30.332 ms 30.322 ms 30.313 ms 30.303 ms * * * *
 3 10.0.3.2 (10.0.3.2) 61.303 ms 61.294 ms 60.394 ms 60.350 ms 60.337 ms 60.328 ms * * * *
 4 10.0.5.10 (10.0.5.10) 66.038 ms 66.028 ms 60.922 ms 60.888 ms 60.504 ms 60.483 ms 60.472 ms 60.462 ms 61.267 ms 61.234 ms
```

Figure 4: Tempo de ida e volta

- Como é possível observar na figura 4, foram enviados 10 pacotes, e na última linha podemos verificar o tempo entre o envio e a resposta do servidor. Assim sendo, basta fazer uma média dos 10 tempos medidos para obter o valor médio do tempo de ida-e-volta.

$$- (66.038 + 66.028 + 60.922 + 60.888 + 60.504 + 60.483 + 60.472 + 60.462 + 61.267 + 61.234) / 10 = \mathbf{61.8298 \text{ ms}}$$

### 1.1.4 Alínea D

“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 numa rede real?”

- O valor médio de atraso não podia ser calculado dividindo o RTT por dois pois para isso estávamos a assumir que a distância percorrida era a mesma nos dois sentidos, o que não acontece. Um pacote pode percorrer um determinado caminho em direção ao destino e voltar à sua origem por um caminho completamente diferente.

## 1.2 Questão 2

“Pretende-se agora usar o traceroute na sua máquina nativa e gerar datagramas IP de diferentes tamanhos.

*Procedimento a seguir:*

Usando o wireshark capture o tráfego gerado pelo traceroute sem especificar o tamanho do pacote, i.e., quando é usado o tamanho do pacote de prova por defeito. Utilize como máquina destino o host marco.uminho.pt. Pare a captura. Com base no tráfego capturado, identifique os pedidos ICMP Echo Request e o conjunto de mensagens devolvidas como resposta.

Selecione a primeira mensagem ICMP capturada e centre a análise no nível protocolar IP e, em particular, do cabeçalho IP (expandir o tab correspondente na janela de detalhe do wireshark).”

### 1.2.1 Alínea A

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.26.30.116	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=409/39169, ttl=255 (reply in 2)
2	0.002588	193.136.9.240	172.26.30.116	ICMP	70	Echo (ping) reply id=0x0001, seq=409/39169, ttl=61 (request in 1)
3	0.050860	172.26.30.116	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=410/39425, ttl=1 (no response found!)

Figure 5: IP da interface ativa

- O endereço IP da interface ativa do computador é **172.26.30.116**.  
É possível retirar esta informação a partir da figura 5.

### 1.2.2 Alínea B

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

> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{B0FCE783-65BC-4EA3-8946-7FE5008780A9}, 1	
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)	
▼ Internet Protocol Version 4, Src: 172.26.30.116, 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: 56	
Identification: 0x7a7b (31355)	
> 000. .... = Flags: 0x0	
...0 0000 0000 0000 = Fragment Offset: 0	
Time to Live: 255	
Protocol: ICMP (1)	
Header Checksum: 0x0000 [validation disabled]	
[Header checksum status: Unverified]	
Source Address: 172.26.30.116	
Destination Address: 193.136.9.240	
> Internet Control Message Protocol	

Figure 6: Campos de um Pacote - Protocol

- O campo protocol tem valor **1**. Permite identificar o **ICMP**.

### 1.2.3 Alínea C

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

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE500B780A9}, 1
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.30.116, 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: 56
    Identification: 0x7a7b (31355)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.30.116
    Destination Address: 193.136.9.240
> Internet Control Message Protocol
```

Figure 7: Campos de um Pacote - Header Length e Total Length

- O cabeçalho IPv4 contém **20 bytes**. Para calcularmos o tamanho do payload necessitamos apenas de fazer a diferença entre o tamanho do pacote (Total Length) e tamanho do cabeçalho, ou seja,  $56 - 20 = 36$  bytes. Concluimos então o payload tem **36 bytes**.

### 1.2.4 Alínea D

“O datagrama IP foi fragmentado? Justifique.”

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE500B780A9}, 1
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.30.116, 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: 56
    Identification: 0x7a7b (31355)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.30.116
    Destination Address: 193.136.9.240
> Internet Control Message Protocol
```

Figure 8: Campos de um Pacote - Fragment Offset

- Analisando o Fragment Offset, que tem valor 0, podemos concluir que o datagrama IP não foi fragmentado.

### 1.2.5 Alínea E

“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. ”

1 0.000000	172.26.30.116	193.136.9.240	ICMP	70 Echo (ping) request	Id=0x0001, seq=400/39329, ttl=32 (reply in 2)
1 0.000010	172.26.30.116	193.136.9.240	ICMP	70 Echo (ping) request	Id=0x0001, seq=410/39429, ttl=1 (no response found)
5 0.000040	172.26.30.116	193.136.9.240	ICMP	70 Echo (ping) request	Id=0x0001, seq=412/39481, ttl=2 (no response found)
7 0.151139	172.26.30.116	193.136.9.240	ICMP	70 Echo (ping) request	Id=0x0001, seq=412/39537, ttl=3 (no response found)
9 0.261795	172.26.30.116	193.136.9.240	ICMP	70 Echo (ping) request	Id=0x0001, seq=412/39593, ttl=4 (reply in 10)

Figure 9: Pacotes ordenados

- Os campos do cabeçalho IP que variam de pacote para pacote são o do *TTL* e o de *Identificação*.

Tal como podemos verificar nas seguintes imagens:

```
> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.30.116, 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: 56
    Identification: 0x7a7b (31355)
  000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 255
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.30.116
  Destination Address: 193.136.9.240
> Internet Control Message Protocol
```

Figure 10: Pacote 1

```
> Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.30.116, 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: 56
    Identification: 0x7a7c (31356)
  000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.30.116
  Destination Address: 193.136.9.240
> Internet Control Message Protocol
```

Figure 11: Pacote 2



```

> Frame 5: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.30.116, 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: 56
    Identification: 0x7a7d (31357)
    000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 2
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.30.116
    Destination Address: 193.136.9.240
> Internet Control Message Protocol

```

Figure 12: Pacote 3

```

> Frame 7: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.30.116, 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: 56
    Identification: 0x7a7e (31358)
    000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 3
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.30.116
    Destination Address: 193.136.9.240
> Internet Control Message Protocol

```

Figure 13: Pacote 4

```

> Frame 9: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.30.116, 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: 56
    Identification: 0x7a7f (31359)
    000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 4
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.26.30.116
    Destination Address: 193.136.9.240
> Internet Control Message Protocol

```

Figure 14: Pacote 5

### 1.2.6 Alínea F

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

- Analisando a forma como esses campos variam de pacote para pacote, podemos concluir que há um aumento no valor de 1.

Pode-se observar esse aumento progressivo de 1 no valor do campo Identification do pacote, como demonstrado pela variação dos valores de 0x7a7b a 0x7a7f.

No campo do TTL também há esse aumento, no entanto apenas nos últimos 4 pacotes, visto que o primeiro apresenta o valor de 255 (provavelmente o ping-plotter envia um ping para saber se o destino está ”vivo” e só depois inicia o traceroute).

### 1.2.7 Alínea G

“Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL Exceeded enviadas ao seu computador.”

4 0.055088	172.26.254.254	172.26.30.116	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
6 0.109472	172.16.2.1	172.26.30.116	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
8 0.165074	172.16.115.252	172.26.30.116	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

Figure 15:

```
> Frame 4: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{88FCE783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0)
  > Internet Protocol Version 4, Src: 172.26.254.254, Dst: 172.26.30.116
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
      Total Length: 56
      Identification: 0x4ddf (19935)
    > 0000 .... = Flags: 0x0
      ...0 0000 0000 0000 = Fragment Offset: 0
      Time to Live: 255
      Protocol: ICMP (1)
      Header Checksum: 0xf77d [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 172.26.254.254
      Destination Address: 172.26.30.116
    > Internet Control Message Protocol
```

Figure 16:

```

> Frame 6: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{8BFC783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0)
  Internet Protocol Version 4, Src: 172.16.2.1, Dst: 172.26.30.116
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0xc3a5 (50085)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 254
    Protocol: ICMP (1)
    Header Checksum: 0x807f [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.16.2.1
    Destination Address: 172.26.30.116
  > Internet Control Message Protocol

```

Figure 17:

```

> Frame 8: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{8BFC783-65BC-4EA3-8946-7FE500B780A9}, i
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0)
  Internet Protocol Version 4, Src: 172.16.115.252, Dst: 172.26.30.116
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x3926 (14630)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 253
    Protocol: ICMP (1)
    Header Checksum: 0x9a03 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.16.115.252
    Destination Address: 172.26.30.116
  > Internet Control Message Protocol

```

Figure 18:

“i. Qual é o valor do campo TTL recebido no seu computador? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL Exceeded recebidas no seu computador? Porquê?”

- O TTL diminui ao longo do trajeto, devido à passagem da mensagem por diversos routers. Como as mensagens de erro são enviadas pelos routers mais distantes, a mensagem passa por mais routers intermédios, o que contribui para o decremento do TTL.

“ii. Porque razão as mensagens de resposta ICMP TTL Exceeded são sempre enviadas na origem com um valor TTL relativamente alto?”

- Como dito anteriormente, as mensagens são enviadas de routers distantes, por isso o router arranca com um valor TTL relativamente alto para tentar garantir que o pacote chegue ao destino.

### 1.2.8 Alínea H

“Sabendo que o ICMP é um protocolo pertencente ao nível de rede, discuta se a informação contida no cabeçalho ICMP poderia ser incluída no cabeçalho IPv4? Quais seriam as vantagens/desvantagens resultantes dessa hipotética inclusão?”

- As vantagens e desvantagens de incluirmos as informações do cabeçalho ICMP no cabeçalho IPv4 seriam as seguintes:

#### *VANTAGENS:*

- reduzir o número de pacotes na rede o que poderia trazer benefícios substanciais em termos de eficiência de rede.

#### *DESVANTAGENS:*

- aumento do tamanho dos cabeçalhos, o que pode resultar num aumento da fragmentação o que leva à diminuição do desempenho da rede.

### 1.3 Questão 3

“Pretende-se agora analisar a fragmentação de pacotes IP. Usando o wireshark, capture e observe o tráfego gerado depois do tamanho de pacote ter sido definido para  $(3500 + 54=3554)$  bytes, em que 54 é o número do grupo de trabalho (PL54). De modo a poder visualizar os fragmentos, aceda a Edit -> Prefer-ences -> Protocols e em IPv4 desative a opção “Reassemble fragmented IPv4 datagrams”.”

1156	118.397752	172.26.30.116	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=8f0c)
1157	118.397752	172.26.30.116	193.136.9.240	IPv4	608 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=8f0c)
1158	118.434618	172.26.254.254	172.26.30.116	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
1159	118.448939	172.26.30.116	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0001, seq=322/16897, ttl=2 (no response)
1160	118.448939	172.26.30.116	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=8f0d)
1161	118.448939	172.26.30.116	193.136.9.240	IPv4	608 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=8f0d)
1162	118.451036	172.16.2.1	172.26.30.116	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
1163	118.499464	172.26.30.116	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0001, seq=323/17153, ttl=3 (no response)
1164	118.499464	172.26.30.116	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=8f0e)
1165	118.499464	172.26.30.116	193.136.9.240	IPv4	608 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=8f0e)
1166	118.501023	172.16.115.252	172.26.30.116	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
1167	118.549846	172.26.30.116	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0001, seq=324/17400, ttl=4 (reply in progress)
1168	118.549846	172.26.30.116	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=8f0f)
1169	118.549846	172.26.30.116	193.136.9.240	IPv4	608 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=8f0f)

Figure 19: Pacotes ordenados

#### 1.3.1 Alínea A

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

```
> Frame 1155: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{BBFCE783-65BC-4EA3-8946-7FE50}
> Ethernet II, Src: IntelCor_d6:f2:e0 (c4:bd:e5:d6:f2:e0), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
  > Internet Protocol Version 4, Src: 172.26.30.116, 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: 0x8f0c (36620)
    > 001. .... = Flags: 0x1, More fragments
      ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 1
      Protocol: ICMP (1)
      Header Checksum: 0x0000 [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 172.26.30.116
      Destination Address: 193.136.9.240
    > Internet Control Message Protocol
```

Figure 20: Fragmentação do pacote inicial



### 1.3.3 Alínea C

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

- Tal como foi referido anteriormente, os campos a analisar são os valores do offset do fragmento e do more fragments, que devem ser diferente de 0 e 1, respetivamente, tal como é possível verificar na figura 21.

### 1.3.4 Alínea D

“Estime teoricamente o número de fragmentos gerados a partir do datagrama IP original e o número de bytes transportados no último fragmento desse datagrama. Compare os dois valores estimados com os obtidos através do Wireshark.”

- Com base numa estimativa teórica, é esperado que o datagrama IP original seja fragmentado em três partes, sendo os dois primeiros fragmentos de 1500 bytes cada e o último fragmento de 554 bytes. No entanto, ao comparar esses valores estimados com os dados obtidos através do Wireshark, foi observado que o tamanho do último fragmento é 594 bytes, o que difere da estimativa teórica anteriormente feita.

### 1.3.5 Alínea E

“Como se deteta o último fragmento correspondente ao datagrama original? Estabeleça um filtro no Wireshark que permita listar o último fragmento do primeiro datagrama IP segmentado.”

- O filtro usado é o `ip.flags.mf == 0 && ip.frag_offset != 0`. A 1ª parte (`ip.flags.mf == 0`) permite filtrar apenas os segmentos que não possuem mais fragmentos, enquanto que a 2ª parte filtra apenas os segmentos cujo offset é diferente de 0 (quando ocorreu fragmentação).

### 1.3.6 Alínea F

“Identifique o equipamento onde o datagrama IP original é reconstruído a partir dos fragmentos. A reconstrução poderia ter ocorrido noutro equipamento diferente do identificado? Porquê?”

- O datagrama é reconstruído no destino pois, caso fosse reconstruído noutro ponto, teria de ser desconstruído novamente para ser transportado até ao destino.

### 1.3.7 Alínea G

“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.”

- Os campos que mudam são: fragment offset e More fragments. O fragment offset é inicialmente 0 e é incrementado a cada fragmento subsequente. O More Fragments é inicialmente 1 e permanece assim até ao último fragmento, onde o campo toma o valor de 0.

### 1.3.8 Alínea H

“Por que razão apenas o primeiro fragmento de cada pacote é identificado como sendo um pacote ICMP?”

- O primeiro pacote é aquele que tem o header original, e por isso é um pacote ICMP. Os restantes possuem um header com informações sobre a fragmentação, logo não são enviados como pacotes ICMP.

### 1.3.9 Alínea I

“Com que valor é o tamanho do datagrama comparado a fim de se determinar se este deve ser fragmentado? Quais seriam os efeitos na rede ao aumentar/diminuir este valor?”

- O tamanho do datagrama é comparado com o tamanho máximo que um pacote pode ter para ser transmitido na rede. Neste caso é de 1500 bytes. Se este valor fosse diminuído, os datagramas poderiam ter de ser divididos num número maior de fragmentos. Caso o valor fosse maior, seriam necessários menos fragmentos para transmitir o pacote.



### 1.3.10 Alínea J

“Sabendo que no comando ping a opção -f (Windows), -M do (Linux) ou -D (Mac) ativa a flag “Don’t Fragment” (DF) no cabeçalho do IPv4, usando ping <opção DF> <opção pkt\_size> SIZE marco.uminho.pt, (opção pkt\_size = -l (Windows) ou -s (Linux, Mac), determine o valor máximo de SIZE sem que ocorra fragmentação do pacote? Justifique o valor obtido.”

```
C:\Users\lusmi>ping -f -l 1472 marco.uminho.pt

Pinging marco.uminho.pt [193.136.9.240] with 1472 bytes of data:
Reply from 193.136.9.240: bytes=1472 time=4ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=2ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=4ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=14ms TTL=61

Ping statistics for 193.136.9.240:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 14ms, Average = 6ms

C:\Users\lusmi>ping -f -l 1473 marco.uminho.pt

Pinging marco.uminho.pt [193.136.9.240] with 1473 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.

Ping statistics for 193.136.9.240:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Figure 23:

- Como sabemos que o limite de tamanho que um pacote pode ter sem ser fragmentado é 1500 bytes, e sabemos que o cabeçalho ocupa 20 bytes, acreditamos que existe algum tipo de informação que ocupe 8 bytes, fazendo assim com que não seja possível enviar um pacote com 1473 bytes.

## 2 Segunda Parte - Resolução

“Considere a topologia da Figura 24, disponível no formato .imn via plataforma de e-learning em Conteúdo -> Material de Apoio PL.”

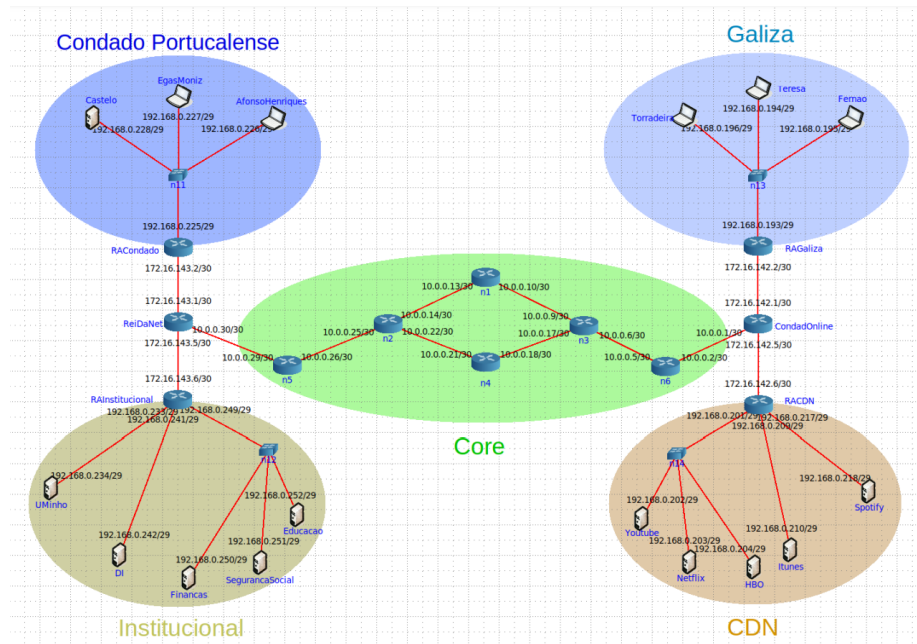


Figure 24: Topologia fornecida

“Ao longo do guião apenas irá fazer uso do endereçamento IPv4. Para facilitar a visualização, pode ocultar o endereçamento IPv6 em View -> Show -> IPv6 Addresses.

A topologia é constituída por quatro polos (Condado Portucalense, Galiza, Institucional e CDN (Content Delivery Network)).

Nos polos Condado e Galiza existe um router de acesso (que permite comunicação com o exterior) ligado a um comutador (switch), por modo a que todos os dispositivos partilhem a mesma rede local. Em cada um dos polos Institucional e CDN encontram-se três sub-redes distintas, criando-se uma segmentação dos dispositivos existentes.

Os polos Condado Portucalense e Institucional estão ligados ao router do ISP ReiDaNet, enquanto Galiza e CDN estão ligados ao ISP CondadoOnline. Interligando os dois ISPs existe um ISP de trânsito cuja rede Core é constituída pelos dispositivos n1 a n6.”

## 2.1 Questão 1

“D.Afonso Henriques afirma ter problemas de comunicação com a sua mãe, D.Teresa. Este alega que o problema deverá estar no dispositivo de D.Teresa, uma vez que no dia anterior conseguiu enviar a sua declaração do IRS para o portal das finanças, e não tem qualquer problema em ver as suas séries favoritas disponíveis na rede de conteúdos.”

### 2.1.1 Alínea A

“Averigue, através do comando ping, que AfonsoHenriques tem efetivamente conectividade com o servidor Financas e com os servidores da CDN.”

```
<ycore.33427/AfonsoHenriques.conf# ping -c 3 192.168.0.250
PING 192.168.0.250 (192.168.0.250) 56(84) bytes of data.
64 bytes from 192.168.0.250: icmp_seq=1 ttl=61 time=0.071 ms
64 bytes from 192.168.0.250: icmp_seq=2 ttl=61 time=0.116 ms
64 bytes from 192.168.0.250: icmp_seq=3 ttl=61 time=0.079 ms

--- 192.168.0.250 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2030ms
rtt min/avg/max/mdev = 0.071/0.088/0.116/0.019 ms
```

Figure 25: Servidor Financas

```
<ycore.33427/AfonsoHenriques.conf# ping -c 3 192.168.0.210
PING 192.168.0.210 (192.168.0.210) 56(84) bytes of data.
64 bytes from 192.168.0.210: icmp_seq=1 ttl=55 time=0.159 ms
64 bytes from 192.168.0.210: icmp_seq=2 ttl=55 time=0.178 ms
64 bytes from 192.168.0.210: icmp_seq=3 ttl=55 time=0.122 ms

--- 192.168.0.210 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.122/0.153/0.178/0.023 ms
```

Figure 26: Servidores CDN

- Com base nas figuras 25 e 26, podemos concluir que AfonsoHenriques tem conectividade com o servidor Financas e com os servidores da CDN.

### 2.1.2 Alínea B

“Recorrendo ao comando `netstat -rn`, analise as tabelas de encaminhamento dos dispositivos AfonsoHenriques e Teresa. Existe algum problema com as suas entradas? Identifique e descreva a utilidade de cada uma das entradas destes dois hosts.”

```
root@AfonsoHenriques:/tmp/pycore.33427/AfonsoHenriques.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 192.168.0.225 0.0.0.0 UG 0 0 0 eth0
192.168.0.224 0.0.0.0 255.255.255.248 U 0 0 0 eth0
```

Figure 27: Tabela de Encaminhamento de AfonsoHenriques

```
root@Teresa:/tmp/pycore.33427/Teresa.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 192.168.0.193 0.0.0.0 UG 0 0 0 eth0
192.168.0.192 0.0.0.0 255.255.255.248 U 0 0 0 eth0
```

Figure 28: Tabela de Encaminhamento de Teresa

- Ao analisar as Tabelas de Encaminhamento dos dispositivos AfonsoHenriques e Teresa utilizando o comando `netstat -rn`, observa-se o seguinte:

No dispositivo AfonsoHenriques:

O gateway sendo 0.0.0.0, indica que os pacotes têm como destino dispositivos dentro da mesma rede que AfonsoHenriques.

O destination sendo 0.0.0.0, indica que os pacotes são enviados para o RACondado para encaminhamento fora da rede.

No dispositivo Teresa:

O gateway sendo 0.0.0.0, indica que os pacotes têm como destino dispositivos dentro da mesma rede que Teresa.

O destination sendo 0.0.0.0, indica que os pacotes são enviados para o RA-Galiza para encaminhamento fora da rede.

### 2.1.3 Alínea C

“Utilize o Wireshark para investigar o comportamento dos routers do core da rede (n1 a n6) quando tenta estabelecer comunicação entre os hosts AfonsoHenriques e Teresa. Indique que dispositivo(s) não permite(m) o encaminhamento correto dos pacotes. Seguidamente, avalie e explique a(s) causa(s) do funcionamento incorreto do dispositivo.

Utilize o comando `ip route add/del` para adicionar as rotas necessárias ou remover rotas incorretas. Verifique a sintaxe completa do comando a usar com `man ip-route` ou `man route`. Poderá também utilizar o comando `traceroute` para se certificar do caminho nó a nó. Considere a alínea resolvida assim que houver tráfego a chegar ao ISP CondadOnline.”

```
<ycore.46215/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225) 0.054 ms 0.006 ms 0.004 ms
 2 172.16.143.1 (172.16.143.1) 0.025 ms 0.008 ms 0.007 ms
 3 10.0.0.29 (10.0.0.29) 0.016 ms !N 0.010 ms !N *
```

Figure 29: Traceroute AfonsoHenriques-Teresa Inicial

Ao analisar o traceroute para *Teresa* vemos que os pacotes chegam até 10.0.0.29 (n5) e não avançam mais. Assim, percebemos que não existia um reencaminhamento em n5 para permitir que os pacotes atravessassem a rede em direção ao destino.

Através do comando `ip route add` adicionamos o endereço de n2 à tabela de encaminhamento de n5, permitindo assim a continuidade do tráfego dos pacotes.

```
<5/AfonsoHenriques.conf# traceroute -I 192.168.0.194
traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225) 0.027 ms 0.005 ms 0.005 ms
 2 172.16.143.1 (172.16.143.1) 0.014 ms 0.007 ms 0.007 ms
 3 10.0.0.29 (10.0.0.29) 0.017 ms 0.010 ms 0.010 ms
 4 10.0.0.25 (10.0.0.25) 0.040 ms 0.013 ms 0.012 ms
 5 10.0.0.25 (10.0.0.25) 3066.887 ms !H 3066.877 ms !H 3066.872 ms !H
```

Figure 30: Traceroute a passar pelo n5 e n2

Destination	Gateway	Genmask	Flags	MSS Window	irtt	Iface
10.0.0.0	10.0.0.13	255.255.255.252	UG	0 0	0	eth1
10.0.0.4	10.0.0.21	255.255.255.252	UG	0 0	0	eth0
10.0.0.8	10.0.0.13	255.255.255.252	UG	0 0	0	eth1
10.0.0.12	0.0.0.0	255.255.255.252	U	0 0	0	eth1
10.0.0.16	10.0.0.13	255.255.255.252	UG	0 0	0	eth1
10.0.0.20	0.0.0.0	255.255.255.252	U	0 0	0	eth0
10.0.0.24	0.0.0.0	255.255.255.252	U	0 0	0	eth2
10.0.0.28	10.0.0.26	255.255.255.252	UG	0 0	0	eth2
172.0.0.0	10.0.0.26	255.0.0.0	UG	0 0	0	eth2
172.16.142.0	10.0.0.13	255.255.255.252	UG	0 0	0	eth1
172.16.142.4	10.0.0.21	255.255.255.252	UG	0 0	0	eth0
172.16.143.0	10.0.0.26	255.255.255.252	UG	0 0	0	eth2
172.16.143.4	10.0.0.26	255.255.255.252	UG	0 0	0	eth2
192.168.0.192	10.0.0.13	255.255.255.248	UG	0 0	0	eth1
192.168.0.194	10.0.0.25	255.255.255.254	UG	0 0	0	eth2
192.168.0.200	10.0.0.21	255.255.255.248	UG	0 0	0	eth0
192.168.0.208	10.0.0.21	255.255.255.248	UG	0 0	0	eth0
192.168.0.216	10.0.0.21	255.255.255.248	UG	0 0	0	eth0
192.168.0.224	10.0.0.26	255.255.255.248	UG	0 0	0	eth2
192.168.0.232	10.0.0.26	255.255.255.248	UG	0 0	0	eth2
192.168.0.240	10.0.0.26	255.255.255.248	UG	0 0	0	eth2
192.168.0.248	10.0.0.26	255.255.255.248	UG	0 0	0	eth2

Figure 31: Tabela de Encaminhamento de n2

Como podemos observar na tabela de Encaminhamento, quando o endereço de Destino é *192.168.0.194* os pacotes são encaminhados para *10.0.0.25* que se observarmos a imagem da topologia, é o endereço do próprio n2. Para resolver este conflito utilizamos o comando *ip route del* para remover esta entrada, e de seguida o *ip route add* para que o tráfego fosse enviado para o endereço de n1.

```

traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225)  0.027 ms  0.006 ms  0.005 ms
 2 172.16.143.1 (172.16.143.1)  0.015 ms  0.008 ms  0.008 ms
 3 10.0.0.29 (10.0.0.29)  0.015 ms  0.009 ms  0.010 ms
 4 10.0.0.25 (10.0.0.25)  0.018 ms  0.013 ms  0.012 ms
 5 10.0.0.13 (10.0.0.13)  0.045 ms  0.016 ms  0.015 ms
 6 10.0.0.25 (10.0.0.25)  0.014 ms  0.022 ms  0.015 ms
 7 10.0.0.13 (10.0.0.13)  0.017 ms  0.017 ms  0.016 ms

```

Figure 32: Traceroute a passar pelo n1

Como podemos observar aqui o tráfego volta a passar por n2 depois de n1, o que gera um loop e não um caminho eficiente, pelo que removemos a entrada com destino *192.168.0.192* da tabela de Encaminhamento seguinte, e adicionando em seguida uma entrada com o mesmo destino mas com Gateway para n3.

Destination	Gateway	Genmask	Flags	MSS Window	irrtt	Iface
10.0.0.0	10.0.0.9	255.255.255.252	UG	0 0	0	eth0
10.0.0.4	10.0.0.9	255.255.255.252	UG	0 0	0	eth0
10.0.0.8	0.0.0.0	255.255.255.252	U	0 0	0	eth0
10.0.0.12	0.0.0.0	255.255.255.252	U	0 0	0	eth1
10.0.0.16	10.0.0.9	255.255.255.252	UG	0 0	0	eth0
10.0.0.20	10.0.0.14	255.255.255.252	UG	0 0	0	eth1
10.0.0.24	10.0.0.14	255.255.255.252	UG	0 0	0	eth1
10.0.0.28	10.0.0.14	255.255.255.252	UG	0 0	0	eth1
172.0.0.0	10.0.0.14	255.0.0.0	UG	0 0	0	eth1
172.16.142.0	10.0.0.9	255.255.255.252	UG	0 0	0	eth0
172.16.142.4	10.0.0.9	255.255.255.252	UG	0 0	0	eth0
172.16.143.0	10.0.0.14	255.255.255.252	UG	0 0	0	eth1
172.16.143.4	10.0.0.14	255.255.255.252	UG	0 0	0	eth1
192.168.0.192	10.0.0.14	255.255.255.248	UG	0 0	0	eth1
192.168.0.200	10.0.0.9	255.255.255.248	UG	0 0	0	eth0
192.168.0.208	10.0.0.9	255.255.255.248	UG	0 0	0	eth0
192.168.0.216	10.0.0.9	255.255.255.248	UG	0 0	0	eth0
192.168.0.224	10.0.0.14	255.255.255.248	UG	0 0	0	eth1
192.168.0.232	10.0.0.14	255.255.255.248	UG	0 0	0	eth1
192.168.0.240	10.0.0.14	255.255.255.248	UG	0 0	0	eth1
192.168.0.248	10.0.0.14	255.255.255.248	UG	0 0	0	eth1

Figure 33: Tabela de Encaminhamento de n1

```

traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225) 0.026 ms 0.005 ms 0.005 ms
 2 172.16.143.1 (172.16.143.1) 0.014 ms 0.008 ms 0.007 ms
 3 10.0.0.29 (10.0.0.29) 0.016 ms 0.010 ms 0.009 ms
 4 10.0.0.25 (10.0.0.25) 0.018 ms 0.012 ms 0.012 ms
 5 10.0.0.13 (10.0.0.13) 0.022 ms 0.016 ms 0.015 ms
 6 10.0.0.17 (10.0.0.17) 0.030 ms !N 0.028 ms !N *
```

Figure 34: Traceroute até n3

Mais uma vez, podemos observar pelo Traceroute que os pacotes são enviados para n3, porém ao observar a sua Tabela de Encaminhamento percebemos que existe uma entrada para *192.168.0.192/28* que envia o tráfego para *10.0.0.18* (n4), o que possibilitaria a criação de um loop, pelo que o removemos com o comando *ip route del*.

Destination	Gateway	Genmask	Flags	MSS	Window	irrt	Iface
10.0.0.0	10.0.0.5	255.255.255.252	UG	0	0	0	eth2
10.0.0.4	0.0.0.0	255.255.255.252	U	0	0	0	eth2
10.0.0.8	0.0.0.0	255.255.255.252	U	0	0	0	eth0
10.0.0.12	10.0.0.10	255.255.255.252	UG	0	0	0	eth0
10.0.0.16	0.0.0.0	255.255.255.252	U	0	0	0	eth1
10.0.0.20	10.0.0.18	255.255.255.252	UG	0	0	0	eth1
10.0.0.24	10.0.0.18	255.255.255.252	UG	0	0	0	eth1
10.0.0.28	10.0.0.10	255.255.255.252	UG	0	0	0	eth0
172.0.0.0	10.0.0.10	255.0.0.0	UG	0	0	0	eth0
172.16.142.0	10.0.0.5	255.255.255.252	UG	0	0	0	eth2
172.16.142.4	10.0.0.5	255.255.255.252	UG	0	0	0	eth2
172.16.143.0	10.0.0.18	255.255.255.252	UG	0	0	0	eth1
172.16.143.4	10.0.0.10	255.255.255.252	UG	0	0	0	eth0
192.168.0.192	10.0.0.5	255.255.255.255	UGH	0	0	0	eth2
192.168.0.192	10.0.0.18	255.255.255.240	UG	0	0	0	eth1
192.168.0.200	10.0.0.5	255.255.255.248	UG	0	0	0	eth2
192.168.0.208	10.0.0.5	255.255.255.248	UG	0	0	0	eth2
192.168.0.216	10.0.0.5	255.255.255.248	UG	0	0	0	eth2
192.168.0.224	10.0.0.18	255.255.255.248	UG	0	0	0	eth1
192.168.0.232	10.0.0.10	255.255.255.248	UG	0	0	0	eth0
192.168.0.240	10.0.0.10	255.255.255.248	UG	0	0	0	eth0
192.168.0.248	10.0.0.10	255.255.255.248	UG	0	0	0	eth0

Figure 35: Tabela de Encaminhamento de n3

Destination	Gateway	Genmask	Flags	MSS	Window	irrt	Iface
10.0.0.0	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
10.0.0.4	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
10.0.0.8	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
10.0.0.12	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
10.0.0.16	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
10.0.0.20	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
10.0.0.24	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
10.0.0.28	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
172.0.0.0	172.16.142.1	255.0.0.0	UG	0	0	0	eth0
172.16.142.0	0.0.0.0	255.255.255.252	U	0	0	0	eth0
172.16.142.4	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
172.16.143.0	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
172.16.143.4	172.16.142.1	255.255.255.252	UG	0	0	0	eth0
192.168.0.192	0.0.0.0	255.255.255.248	U	0	0	0	eth1
192.168.0.200	172.16.142.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.208	172.16.142.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.216	172.16.142.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.232	172.16.142.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.240	172.16.142.1	255.255.255.248	UG	0	0	0	eth0
192.168.0.248	172.16.142.1	255.255.255.248	UG	0	0	0	eth0

Figure 36: Tabela de Encaminhamento RAGaliza



```

traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.032 ms  0.006 ms  0.005 ms
 2  172.16.143.1 (172.16.143.1)  0.014 ms  0.007 ms  0.007 ms
 3  10.0.0.29 (10.0.0.29)  0.016 ms  0.010 ms  0.009 ms
 4  10.0.0.25 (10.0.0.25)  0.021 ms  0.013 ms  0.012 ms
 5  10.0.0.13 (10.0.0.13)  0.024 ms  0.014 ms  0.014 ms
 6  10.0.0.17 (10.0.0.17)  0.038 ms  0.026 ms  0.018 ms
 7  10.0.0.5 (10.0.0.5)  0.031 ms  0.021 ms  0.021 ms
 8  10.0.0.1 (10.0.0.1)  0.030 ms  0.024 ms  0.023 ms
 9  172.16.142.2 (172.16.142.2)  0.068 ms  0.033 ms  0.027 ms
10  192.168.0.194 (192.168.0.194)  0.041 ms  0.029 ms  0.029 ms

```

Figure 37: Traceroute Completo

Depois de estabelecer as rotas entre todos os nós é possível verificar que o traceroute permite chegar a *Teresa* a partir de *AfonsoHenriques*.

#### 2.1.4 Alínea D

“Uma vez que o core da rede esteja a encaminhar corretamente os pacotes enviados por AfonsoHenriques, confira com o Wireshark se estes são recebidos por Teresa. ”

“i. Em caso afirmativo, porque é que continua a não existir conectividade entre D.Teresa e D.Afonso Henriques? Efetue as alterações necessárias para garantir que a conectividade é restabelecida e o confronto entre os dois é evitado.”

```
root@AfonsoHenriques:/tmp/pycore.46215/AfonsoHenriques.conf# ping 192.168.0.194
PING 192.168.0.194 (192.168.0.194) 56(84) bytes of data:
64 bytes from 192.168.0.194: icmp_seq=1 ttl=55 time=0.129 ms
64 bytes from 192.168.0.194: icmp_seq=2 ttl=55 time=0.220 ms
64 bytes from 192.168.0.194: icmp_seq=3 ttl=55 time=0.213 ms
64 bytes from 192.168.0.194: icmp_seq=4 ttl=55 time=0.306 ms
64 bytes from 192.168.0.194: icmp_seq=5 ttl=55 time=0.141 ms
64 bytes from 192.168.0.194: icmp_seq=6 ttl=55 time=0.498 ms
64 bytes from 192.168.0.194: icmp_seq=7 ttl=55 time=0.089 ms
64 bytes from 192.168.0.194: icmp_seq=8 ttl=55 time=0.292 ms
64 bytes from 192.168.0.194: icmp_seq=9 ttl=55 time=0.204 ms
64 bytes from 192.168.0.194: icmp_seq=10 ttl=55 time=0.215 ms
64 bytes from 192.168.0.194: icmp_seq=11 ttl=55 time=0.206 ms
64 bytes from 192.168.0.194: icmp_seq=12 ttl=55 time=0.210 ms
64 bytes from 192.168.0.194: icmp_seq=13 ttl=55 time=0.197 ms
```

Figure 38:

- Como obtemos um reply do ping entao os dispositivos estao conectados.

“ii. As rotas dos pacotes ICMP echo reply são as mesmas, mas em sentido inverso, que as rotas dos pacotes ICMP echo request enviados entre AfonsoHenriques e Teresa?”

```
tracert to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.029 ms  0.006 ms  0.005 ms
 2  172.16.142.1 (172.16.142.1)  0.016 ms  0.007 ms  0.007 ms
 3  10.0.0.2 (10.0.0.2)  0.018 ms  0.052 ms  0.015 ms
 4  10.0.0.6 (10.0.0.6)  0.020 ms  0.012 ms  0.012 ms
 5  10.0.0.18 (10.0.0.18)  0.023 ms  0.016 ms  0.016 ms
 6  10.0.0.14 (10.0.0.14)  0.027 ms  0.036 ms  0.020 ms
 7  10.0.0.26 (10.0.0.26)  0.029 ms  0.022 ms  0.022 ms
 8  10.0.0.30 (10.0.0.30)  0.029 ms  0.025 ms  0.024 ms
 9  172.16.143.2 (172.16.143.2)  0.033 ms  0.028 ms  0.027 ms
10  192.168.0.226 (192.168.0.226)  0.035 ms  0.030 ms  0.029 ms
```

Figure 39: Traceroute Teresa-AfonsoHenriques

```

traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
 1 192.168.0.225 (192.168.0.225)  0.034 ms  0.015 ms  0.005 ms
 2 172.16.143.1 (172.16.143.1)  0.014 ms  0.007 ms  0.007 ms
 3 10.0.0.29 (10.0.0.29)  0.017 ms  0.010 ms  0.009 ms
 4 10.0.0.25 (10.0.0.25)  0.022 ms  0.013 ms  0.012 ms
 5 10.0.0.13 (10.0.0.13)  0.025 ms  0.015 ms  0.016 ms
 6 10.0.0.17 (10.0.0.17)  0.033 ms  0.028 ms  0.020 ms
 7 10.0.0.5 (10.0.0.5)  0.033 ms  0.021 ms  0.021 ms
 8 10.0.0.1 (10.0.0.1)  0.031 ms  0.024 ms  0.023 ms
 9 172.16.142.2 (172.16.142.2)  0.033 ms  0.028 ms  0.028 ms
10 192.168.0.194 (192.168.0.194)  0.045 ms  0.032 ms  0.031 ms

```

Figure 40: Traceroute AfonsoHenriques-Teresa

-

#### 2.1.5 Alínea E

“Estando restabelecida a conectividade entre os dois hosts, obtenha a tabela de encaminhamento de n3 e foque-se na seguinte entrada:

```

192.168.0.192 20.0.0.18 255.255.255.240 UG 0 0 0 eth1

```

Existe uma correspondência (match) nesta entrada para pacotes enviados para o polo Galiza? E para CDN?

Caso seja essa a entrada utilizada para o encaminhamento, permitirá o funcionamento esperado do dispositivo?

Ofereça uma explicação pela qual essa entrada é ou não utilizada.”

-

#### 2.1.6 Alínea F

“Os endereços utilizados pelos quatro polos são endereços públicos ou privados? E os utilizados no core da rede/ISPs? Justifique convenientemente.”

-

#### 2.1.7 Alínea G

“Os switches localizados em cada um dos polos têm um endereço IP atribuído? Porquê?”

-

## 2.2 Questão 2

“Tendo feito as pazes com a mãe, D. Afonso Henriques vê-se com algum tempo livre e decide fazer remodelações no condado:”

### 2.2.1 Alínea A

“Não estando satisfeito com a decoração do Castelo, opta por eliminar a sua rota default.

Adicione as rotas necessárias para que o Castelo continue a ter acesso a cada um dos três polos. Mostre que a conectividade é restabelecida, assim como a tabela de encaminhamento resultante. Explícite ainda a utilidade de uma rota default.”

```
root@Castelo:/tmp/pycore.46215/Castelo.conf# ip route del default
root@Castelo:/tmp/pycore.46215/Castelo.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
192.168.0.224 0.0.0.0 255.255.255.248 U 0 0 0 eth0
```

Figure 41: Tabela de Encaminhamento após eliminar a rota default

```
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
192.168.0.192 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.200 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.208 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.216 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.224 0.0.0.0 255.255.255.248 U 0 0 0 eth0
192.168.0.232 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.240 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
192.168.0.248 192.168.0.225 255.255.255.248 UG 0 0 0 eth0
```

Figure 42: Tabela de Encaminhamento Final

-

### 2.2.2 Alínea B

“Por modo a garantir uma posição estrategicamente mais vantajosa e ter casa de férias para relaxar entre batalhas, ordena também a construção de um segundo Castelo, em Braga. Não tendo qualquer queixa do serviço prestado, recorre novamente aos serviços do ISP ReiDaNet para ter acesso à rede no segundo Castelo. O ISP atribuiu-lhe o endereço de rede IP 172.16.XX.128/26 em que XX corresponde ao seu número de grupo (PLXX). Defina um esquema de endereçamento que permita o estabelecimento de pelo menos 3 redes e que garanta que cada uma destas possa ter 10 ou mais hosts. Assuma que todos os endereços de sub-redes são utilizáveis.”

-

### **2.2.3 Alínea C**

“Ligue um novo host diretamente ao router ReiDaNet. Associe-lhe um endereço, à sua escolha, pertencente a uma sub-rede disponível das criadas na alínea anterior (garanta que a interface do router ReiDaNet utiliza o primeiro endereço da sub-rede escolhida). Verifique que tem conectividade com os diferentes polos.

Existe algum host com o qual não seja possível comunicar? Porquê?”

-

## **2.3 Questão 3**

“Ao planear um novo ataque, D. Afonso Henriques constata que o seu exército não só perde bastante tempo a decidir que direção tomar a cada salto como, por vezes, inclusivamente se perde.”

### **2.3.1 Alínea A**

“De modo a facilitar a travessia, elimine as rotas referentes a Galiza e CDN no dispositivo n6 e defina um esquema de sumarização de rotas (Supernetting) que permita o uso de apenas uma rota para ambos os polos. Confirme que a conectividade é mantida.”

-

### **2.3.2 Alínea B**

“Repita o processo descrito na alínea anterior para CondadoPortucalense e Institucional, também no dispositivo n6.”

-

### **2.3.3 Alínea C**

“Comente os aspetos positivos e negativos do uso do Supernetting.”

-