

TP2 - Redes de Computadores

Francisco Ferreira - a100660

Júlio Pinto - a100742

Rui Lopes - a100643

16 de maio de 2023

Conteúdo

1	Parte I - Exercício 1	3
1.1	Alínea a)	3
1.1.1	Resposta alínea a)	3
1.2	Alínea b)	4
1.2.1	Resposta alínea b)	4
1.3	Alínea c)	5
1.3.1	Resposta alínea c)	5
1.4	Alínea d)	5
1.4.1	Resposta alínea d)	5
2	Parte I - Exercício 2	6
2.1	Alínea a)	6
2.1.1	Resposta alínea a)	6
2.2	Alínea b)	6
2.2.1	Resposta alínea b)	6
2.3	Alínea c)	6
2.3.1	Resposta alínea c)	6
2.4	Alínea d)	6
2.4.1	Resposta alínea d)	6
2.5	Alínea e)	7
2.5.1	Resposta alínea e)	7
2.6	Alínea f)	7
2.6.1	Resposta alínea f)	7
2.7	Alínea g)	7
2.7.1	Resposta alínea g.i)	7
2.7.2	Resposta alínea g.ii)	8
2.8	Alínea h)	8
2.8.1	Resposta alínea h)	8
3	Parte I - Exercício 3	9
3.1	Alínea a)	9
3.1.1	Resposta alínea a)	9
3.2	Alínea b)	9
3.2.1	Resposta alínea b)	9
3.3	Alínea c)	9
3.3.1	Resposta alínea c)	9
3.4	Alínea d)	9
3.4.1	Resposta alínea d)	10
3.5	Alínea e)	10
3.5.1	Resposta alínea e)	10
3.6	Alínea f)	10

3.6.1	Resposta alínea f)	10
3.7	Alínea g)	10
3.7.1	Resposta alínea g)	10
3.8	Alínea h)	10
3.8.1	Resposta alínea h)	11
3.9	Alínea i)	11
3.9.1	Resposta alínea i)	11
3.10	Alínea j)	11
3.10.1	Resposta alínea j)	11
4	Parte II - Exercício 1	12
4.1	Alínea a)	12
4.1.1	Resposta alínea a)	12
4.2	Alínea b)	13
4.2.1	Resposta alínea b)	13
4.3	Alínea c)	14
4.3.1	Resposta alínea c)	14
4.4	Alínea d)	15
4.4.1	Resposta alínea d.i)	15
4.4.2	Resposta alínea d.ii)	16
4.5	Alínea e)	17
4.5.1	Resposta alínea e)	17
4.6	Alínea f)	17
4.6.1	Resposta alínea f)	17
4.7	Alínea g)	17
4.7.1	Resposta alínea g)	17
5	Parte II - Exercício 2	18
5.1	Alínea a)	18
5.1.1	Resposta alínea a)	18
5.2	Alínea b)	19
5.2.1	Resposta alínea b)	19
5.3	Alínea c)	20
5.3.1	Resposta alínea c)	20
6	Parte II - Exercício 3	21
6.1	Alínea a)	21
6.1.1	Resposta alínea a)	21
6.2	Alínea b)	23
6.2.1	Resposta alínea b)	23
6.3	Alínea c)	24
6.3.1	Resposta alínea c)	24

1 Parte I - Exercício 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.

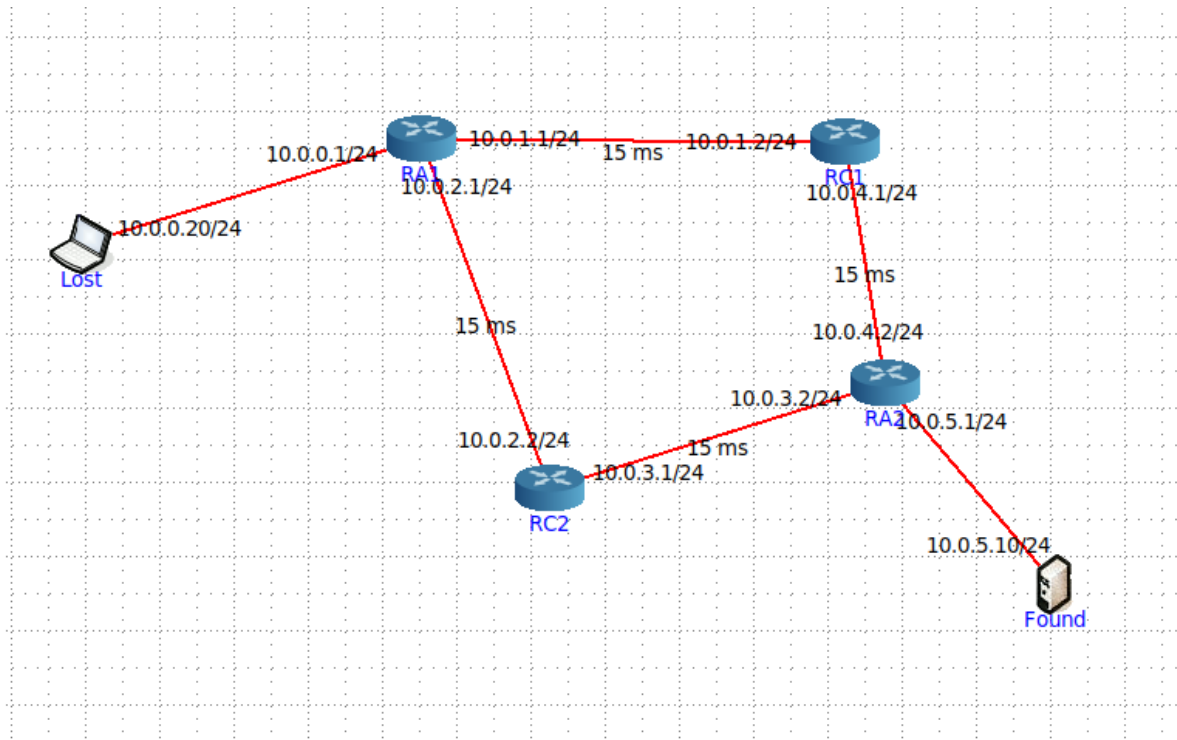


Figura 1: Topologia do exercício 1

1.1 Alínea a)

Active 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.

1.1.1 Resposta alínea a)

O comando *traceroute* é usado para rastrear o caminho que um pacote de rede leva de um host de origem para uma host destino. Ele faz isso enviando pacotes ICMP com diferentes valores TTL para o host destino (neste caso, Found). O TTL é um valor numérico no header do pacote IP que indica quantos routers o pacote pode passar antes de ser descartado. Quando um pacote alcança um router, o router decrementa o valor TTL e encaminha o pacote para o próximo router no caminho para o Found.

O primeiro pacote enviado tem um TTL inicial de 1. O primeiro router no caminho para o Found recebe o pacote e descarta-o, e envia de volta uma mensagem ICMP de "time exceeded", indicando que o TTL do pacote expirou antes que o pacote pudesse alcançar o Found. O *traceroute* regista o endereço IP do primeiro router e o tempo que levou para receber a mensagem ICMP de "time exceeded". Em seguida, o traceroute envia outro pacote ICMP com um TTL de 2, que fará com que ele passe por dois

routers antes de expirar e receber outra mensagem ICMP de "time exceeded". O processo continua com incrementos no valor do TTL até que o Found seja alcançado e responda com uma mensagem ICMP de "echo reply".

Ao observar o tráfego gerado pelo traceroute com o Wireshark, é possível visualizar as mensagens ICMP de "time exceeded" enviadas pelos routers ao longo do caminho e as mensagens ICMP de "echo reply" enviadas pelo Found. Isso permite que se tenha uma ideia do caminho que os pacotes estão a tomar para alcançar o Found e o tempo que leva para cada pacote alcançar o próximo router.

```

1 > traceroute -I 10.0.5.10
2 traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
3 1 10.0.0.1 (10.0.0.1) 0.175 ms 0.039 ms 0.031 ms
4 2 10.0.1.2 (10.0.1.2) 30.832 ms 30.713 ms 30.689 ms
5 3 10.0.3.2 (10.0.3.2) 94.447 ms 94.435 ms 94.423 ms
6 4 10.0.5.10 (10.0.5.10) 94.408 ms 94.395 ms 94.383 ms

```

1.2 Alínea b)

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

1.2.1 Resposta alínea b)

O valor mínimo do TTL deve ser 4, isto pois, é a essa distância que se encontra o host *Found*. Como se pode comprovar através da análise da imagem anexada.

icmp						
No.	Time	Source	Destination	Protocol	Length	Info
4	1.367302524	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=2/512, ttl=1 (no response...
5	1.367305023	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
6	1.367309284	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=3/768, ttl=1 (no response...
7	1.367311174	10.0.0.1	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
8	1.367315671	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=4/1024, ttl=1 (no respons...
9	1.367323549	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=5/1280, ttl=2 (no respons...
10	1.367327387	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=6/1536, ttl=2 (no respons...
11	1.367331819	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=7/1792, ttl=3 (no respons...
12	1.367335467	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=8/2048, ttl=3 (no respons...
13	1.367339194	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=9/2304, ttl=3 (no respons...
14	1.367343555	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=10/2560, ttl=4 (reply in ...
15	1.367347269	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=11/2816, ttl=4 (reply in ...
16	1.367350982	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=12/3072, ttl=4 (reply in ...
17	1.367355295	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=13/3328, ttl=5 (reply in ...
18	1.367359384	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=14/3584, ttl=5 (reply in ...
19	1.367363179	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=15/3840, ttl=5 (reply in ...
20	1.367368645	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=16/4096, ttl=6 (reply in ...
21	1.367720563	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=17/4352, ttl=6 (reply in ...
22	1.367726202	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=18/4608, ttl=6 (reply in ...
23	1.367730608	10.0.0.20	10.0.5.10	ICMP	74	Echo (ping) request id=0x001d, seq=19/4864, ttl=7 (reply in ...
24	1.461934202	10.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

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.

1.3.1 Resposta alínea c)

Calculando a média a partir da última linha do *traceroute*, deduzimos que o *Round-Trip Time* tem o valor de:

$$RTT = (62.691 + 61.266 + 61.238 + 61.224 + 61.212)/5 = 61.5262ms$$

```
1 > traceroute -I 10.0.5.10 -q 5
2 traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
3 1 10.0.0.1 (10.0.0.1) 0.066 ms 0.021 ms 0.017 ms 0.017 ms 0.019 ms
4 2 10.0.1.2 (10.0.1.2) 30.268 ms 30.250 ms 30.237 ms 31.318 ms 31.306 ms
5 3 10.0.3.2 (10.0.3.2) 62.758 ms 62.745 ms 62.732 ms 62.720 ms 62.707 ms
6 4 10.0.5.10 (10.0.5.10) 62.691 ms 61.266 ms 61.238 ms 61.224 ms 61.212 ms
```

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?

1.4.1 Resposta alínea d)

Não, o valor médio do atraso num sentido (One-Way Delay) não pode ser calculado simplesmente dividindo o RTT por dois. Isto acontece porque o RTT inclui o tempo de ida e volta, enquanto o One-Way Delay mede apenas o tempo de atraso num único sentido.

Para medir o One-Way Delay, é necessário ter uma fonte confiável de tempo para sincronizar os relógios de todos os dispositivos envolvidos na comunicação. Além disso, é preciso ter um método para registar a hora em que um pacote é enviado e recebido em cada dispositivo. A partir dessas informações, é possível calcular o atraso em cada sentido.

No entanto, o cálculo do One-Way Delay numa rede real pode ser difícil porque a rede pode ter um comportamento imprevisível e variável. Por exemplo, o atraso pode variar devido a congestionamento da rede, problemas de encaminhamento, problemas de configuração de rede, entre outros fatores. Além disso, pode haver dispositivos intermédios, como switches e routers, que introduzem atrasos adicionais na comunicação. Portanto, é importante usar técnicas avançadas de medição e monitorização da rede para medir o One-Way Delay com precisão numa rede real.

2 Parte I - Exercício 2

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).

2.1 Alínea a)

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

2.1.1 Resposta alínea a)

O endereço IP da interface ativa do computador é 172.26.44.37

2.2 Alínea b)

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

2.2.1 Resposta alínea b)

O valor do campo protocol é ICMP(1). Este campo permite identificar que o datagrama IP contém uma mensagem ICMP.

2.3 Alínea c)

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

2.3.1 Resposta alínea c)

O cabeçalho IPv4 tem 20 bytes. Enquanto que o *Payload* do datagrama tem 40 bytes.

$\text{Payload} = \text{total length} - \text{header length}$

$\text{Payload} = 60 - 20 = 40 \text{ bytes}$

2.4 Alínea d)

O datagrama IP foi fragmentado? Justifique.

2.4.1 Resposta alínea d)

Não. Concluímos que o datagrama não foi fragmentado isto pois a flag "*More Fragments*" não contém nenhum valor e o "*Fragment Offset*" é igual a 0.

```
Internet Protocol Version 4, Src: 172.26.44.37, 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: 0x001a (26)
  000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    0..... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 1
  Protocol: ICMP (1)
  Header Checksum: 0x15f0 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.44.37
  Destination Address: 193.136.9.240
  Internet Control Message Protocol
```

Figura 2: Base da resposta da alínea d)

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.

2.5.1 Resposta alínea e)

Os campos que sofrem alterações são: o campo de identificação, o TTL (varia apenas de três em três pacotes) e o "Header Checksum".

2.6 Alínea f)

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

2.6.1 Resposta alínea f)

O valor do campo de identificação sobre sempre uma unidade em hexadecimal. Já o TTL sobe em uma unidade a cada três pacotes enviados pela minha máquina.

2.7 Alínea g)

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 recebido no seu computador? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL Exceeded recebidas no seu computador? Porquê?
- Porque razão as mensagens de resposta ICMP TTL Exceeded são sempre enviadas na origem com um valor TTL relativamente alto?

2.7.1 Resposta alínea g.i)

O valor recebido varia entre 253 e 255, portanto, nem sempre é constante. De forma a garantir que o pacote é reencaminhado com sucesso, usa-se sempre o maior valor possível associado ao TTL. Uma vez que só temos 8 bits disponíveis, o valor máximo é 256 (2^8), mas como já foi efetuado um salto o valor passa a ser 255. Além disso, os valores variarem entre 253 e 255 deve-se ao facto do TTL ser decrementado no caminho de volta e os nós estarem a distâncias diferentes do destino.

2.7.2 Resposta alínea g.ii)

Como dito anteriormente, para garantir que o pacote é reencaminhado com sucesso.

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?

2.8.1 Resposta alínea h)

Sim, teoricamente, a informação contida no cabeçalho ICMP poderia ser incluída no cabeçalho IPv4. Uma das principais vantagens de incluir as informações do ICMP no cabeçalho IPv4 é que isso pode reduzir o tamanho dos pacotes enviados pela rede, já que a sobrecarga adicional de um cabeçalho ICMP separado seria eliminada. Isso pode ser especialmente benéfico em redes onde a largura de banda é limitada e cada byte de dados é valioso. Embora seja possível incluir as informações do ICMP no cabeçalho IPv4, essa abordagem pode ter algumas desvantagens significativas, incluindo aumento da complexidade, problemas de compatibilidade e sobrecarga desnecessária do cabeçalho IPv4.

3 Parte I - Exercício 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 $(3500 + X)$ bytes, em que X é o número do grupo de trabalho (e.g., $X=22$ para o grupo PL22).

3.1 Alínea a)

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

3.1.1 Resposta alínea a)

O valor máximo que um datagrama ICMP pode ter é de 1500 bytes, onde 1480 bytes são para os dados e 20 bytes são para o cabeçalho. Uma vez que os pacotes enviados têm um tamanho igual a 3512 bytes, foi necessário recorrer a fragmentação.

3.2 Alínea b)

Imprima o primeiro fragmento do datagrama IP original. 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?

3.2.1 Resposta alínea b)

Uma vez que a flag "More fragments" tem o valor 1, é possível afirmar que o datagrama foi fragmentado. Além disso, como o "Fragment offset" tem o valor 0, isso significa que este fragmento se trata do primeiro. O tamanho do datagrama é de 1480 bytes, valor coincidente com a capacidade máxima de dados num datagrama ICMP.

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?

3.3.1 Resposta alínea c)

É possível afirmar que este fragmento não se trata do primeiro, pois o "Fragment offset" é diferente de 0. Neste caso, o "Fragment offset" é de 1480. Isto pois, o fragmento anterior tinha tamanho igual a 1480 bytes (como anteriormente foi referido). No entanto, existem mais fragmentos do datagrama IP original, pois a flag "More fragments" tem o valor 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.

3.4.1 Resposta alínea d)

Tendo por base um datagrama IP com 3512 bytes de dados e a MTU (maximum transmission unit) igual a 1500, irão ser gerados 3 fragmentos. Os dois primeiros fragmentos terão 1500 bytes (1480 bytes de dados cada) e o último 572 (552 bytes de dados), totalizando então 3512 bytes de dados. No entanto, em cada um dos dois primeiros pacotes, apenas 1480 bytes irão ser de dados. Isto deve-se ao facto de 20 bytes estarem reservados para o cabeçalho IP. No último caso, aplica-se também o raciocínio de 20 bytes estarem reservados para o cabeçalho IP, mas apenas 552 bytes irão ser de dados. Segundo o Wireshark, foram também gerados 3 fragmentos, o que bate certo com o estimado teoricamente. O tamanho de cada fragmento, ao nível dos dados, foi também o esperado: $1480 + 1480 + 552$

15	-6071.035391..	192.168.1.35	193.136.9.240	ICMP	1514 Echo (ping) request id=0x0003, seq=1/256, ttl=64 (reply in 18)
16	-6071.035368..	192.168.1.35	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=3886)
17	-6071.035363..	192.168.1.35	193.136.9.240	IPv4	594 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=3886)

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.

3.5.1 Resposta alínea e)

O último fragmento tem sempre a flag "More fragments" igual a 0 e o "Fragment offset" diferente de 0. Filtro para o Wireshark:

```
1 > ip.dst == 193.136.9.240 && ip.flags.mf == 0 && ip.frag_offset > 0 && ip.id == 0x3886
```

ip.dst == 193.136.9.240 && ip.flags.mf == 0 && ip.frag_offset > 0 && ip.id == 0x3886					
No.	Time	Source	Destination	Protocol	Length Info
17	0.785638484	192.168.1.35	193.136.9.240	IPv4	594 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=3886)

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?

3.6.1 Resposta alínea f)

O pacote é refragmentado no equipamanto destino (identificado pelo ip 193.136.9.240). Sim, a reconstrução poderia ter ocorrido noutro equipamento - desde que para isso, esse equipamento tivesse acesso a todos os fragmentos. No entanto, os equipamentos intermédios apenas reencaminham tráfego (sem olhar para os seus conteúdos).

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.

3.7.1 Resposta alínea g)

Os campos que sofrem alteração são: a flag "More fragments", que tem o valor 1 em todos os fragmentos exceto o último; o "Fragment offset", que indica a posição inicial dos dados no fragmento atual, em relação à posição inicial de dados no pacote original. Este "offset" é utilizado, depois, para reagrupar os fragmentos. Além disso, o "Header checksum" também sofre alterações, já que cada fragmento possui o seu próprio cabeçalho e é necessário garantir a integridade de cada um deles.

3.8 Alínea h)

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

3.8.1 Resposta alínea h)

Ao transmitir um pacote ICMP, esta conta com um cabeçalho ICMP, que se encontra no início da pacote. No entanto, quando o pacote é fragmentado, apenas o primeiro é identificado como ICMP, pois esse é que o "leva" o cabeçalho ICMP consigo (todos os outros "levam" cabeçalhos IP).

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?

3.9.1 Resposta alínea i)

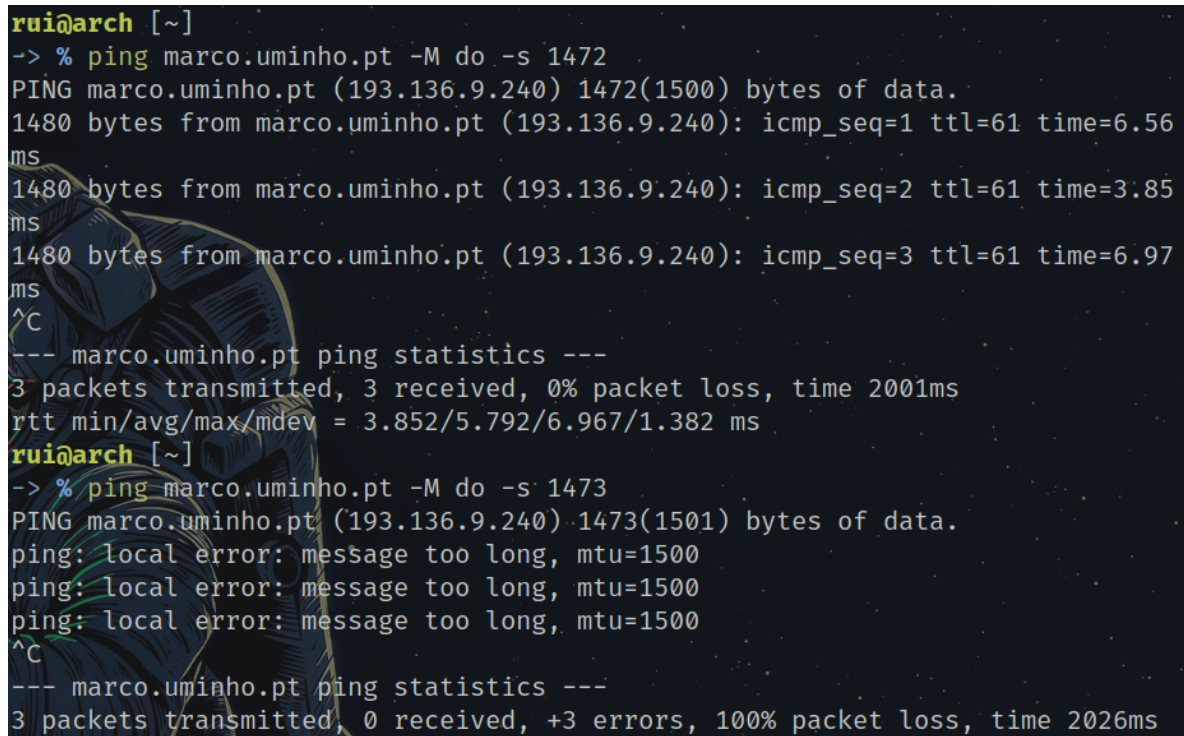
O datagrama é comparado com o valor 1500, o MTU (Maximum Transmission Unit, neste caso). A variação deste valor influencia a quantidade de fragmentos em que o pacote tem de ser transmitido. Quanto maior este valor, menos fragmentos serão necessários.

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.

3.10.1 Resposta alínea j)

O tamanho máximo de `SIZE` é de 1472. Após este valor (1473 ou mais), ocorre fragmentação. Isto deve-se ao facto do MTU ser 1500, mas existirem 20 bytes alocados para o cabeçalho IP e 8 bytes alocados para o cabeçalho ICMP. É possível demonstrar isso através da imagem abaixo:



```
rui@arch [~]
-> % ping marco.uminho.pt -M do -s 1472
PING marco.uminho.pt (193.136.9.240) 1472(1500) bytes of data.
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=1 ttl=61 time=6.56
ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=2 ttl=61 time=3.85
ms
1480 bytes from marco.uminho.pt (193.136.9.240): icmp_seq=3 ttl=61 time=6.97
ms
^C
--- marco.uminho.pt ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 3.852/5.792/6.967/1.382 ms
rui@arch [~]
-> % ping marco.uminho.pt -M do -s 1473
PING marco.uminho.pt (193.136.9.240) 1473(1501) bytes of data.
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
ping: local error: message too long, mtu=1500
^C
--- marco.uminho.pt ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2026ms
```

4 Parte II - Exercício 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

4.1 Alínea a)

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

4.1.1 Resposta alínea a)

De maneira a concluir que, efetivamente, AfonsoHenriques tem conectividade com o servidor Finanças e os servidores da CDN recolhemos as seguintes imagens:

```
<ycore.33539/AfonsoHenriques.conf# ping 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=0.295 ms
64 bytes from 192.168.0.250: icmp_seq=2 ttl=61 time=0.126 ms
^C
--- 192.168.0.250 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.126/0.210/0.295/0.084 ms
root@AfonsoHenriques:/tmp/pycore.33539/AfonsoHenriques.conf#
```

Figura 3: Conectividade ao Servidor Finanças.

```

<ycore.33539/AfonsoHenriques.conf# ping 192.168.0.202
PING 192.168.0.202 (192.168.0.202) 56(84) bytes of data.
64 bytes from 192.168.0.202: icmp_seq=1 ttl=55 time=0.258 ms
^C
--- 192.168.0.202 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.258/0.258/0.258/0.000 ms
root@AfonsoHenriques:/tmp/pycore.33539/AfonsoHenriques.conf#

<ycore.33539/AfonsoHenriques.conf# ping 192.168.0.203
PING 192.168.0.203 (192.168.0.203) 56(84) bytes of data.
64 bytes from 192.168.0.203: icmp_seq=1 ttl=55 time=0.209 ms
64 bytes from 192.168.0.203: icmp_seq=2 ttl=55 time=0.116 ms
^C
--- 192.168.0.203 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.116/0.162/0.209/0.046 ms
root@AfonsoHenriques:/tmp/pycore.33539/AfonsoHenriques.conf#

<core.33539/AfonsoHenriques.conf# ping 192.168.0.204
PING 192.168.0.204 (192.168.0.204) 56(84) bytes of data.
64 bytes from 192.168.0.204: icmp_seq=1 ttl=55 time=0.370 ms
64 bytes from 192.168.0.204: icmp_seq=2 ttl=55 time=0.214 ms
^C
--- 192.168.0.204 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.214/0.292/0.370/0.078 ms
root@AfonsoHenriques:/tmp/pycore.33539/AfonsoHenriques.conf#

<core.33539/AfonsoHenriques.conf# ping 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=0.348 ms
64 bytes from 192.168.0.210: icmp_seq=2 ttl=55 time=0.286 ms
^C
--- 192.168.0.210 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.286/0.317/0.348/0.031 ms
root@AfonsoHenriques:/tmp/pycore.33539/AfonsoHenriques.conf#

<core.33539/AfonsoHenriques.conf# ping 192.168.0.218
PING 192.168.0.218 (192.168.0.218) 56(84) bytes of data.
64 bytes from 192.168.0.218: icmp_seq=1 ttl=55 time=0.261 ms
64 bytes from 192.168.0.218: icmp_seq=2 ttl=55 time=0.225 ms
64 bytes from 192.168.0.218: icmp_seq=3 ttl=55 time=0.210 ms
^C
--- 192.168.0.218 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 0.210/0.232/0.261/0.021 ms
root@AfonsoHenriques:/tmp/pycore.33539/AfonsoHenriques.conf#

```

Figura 4: Conectividade aos hosts do CDN.

4.2 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

4.2.1 Resposta alínea b)

Tabela de reencaminhamento de AfonsoHenriques:

```

1 > netstat -rn
2 Kernel IP routing table
3 Destination      Gateway            Genmask           Flags     MSS Window  irtt Iface
4 0.0.0.0           192.168.0.225     0.0.0.0           UG        0 0       0 eth0
5 192.168.0.224     0.0.0.0           255.255.255.248   U         0 0       0 eth0

```

Esta tabela não apresenta nenhum problema com as entradas. A primeira entrada é a "default", ou seja, todo o tráfego cujo IP destino não esteja discriminado na tabela é reencaminhado por aí. A segunda entrada refere-se a todos os IPs que se encontrem na mesma subrede que o equipamento atual. Nesses casos, o tráfego é reencaminhado pelo próprio equipamento até ao host destino.

Tabela de reencaminhamento de Teresa:

```
1 > netstat -rn
2 Kernel IP routing table
3 Destination      Gateway           Genmask          Flags   MSS Window  irtt Iface
4 0.0.0.0           192.168.0.193    0.0.0.0          UG      0 0      0 eth0
5 192.168.0.192     0.0.0.0          255.255.255.248 U        0 0      0 eth0
```

Conseguimos deduzir que o foi dito sobre a tabela anterior também poderá ser dito relativamente a esta tabela. A única diferença seriam os seus endereços IP.

4.3 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.

4.3.1 Resposta alínea c)

```
1 > traceroute 192.168.0.194
2 traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
3 1 192.168.0.225 (192.168.0.225) 0.175 ms 0.028 ms 0.035 ms
4 2 172.16.143.1 (172.16.143.1) 0.090 ms 0.030 ms 0.028 ms
5 3 10.0.0.29 (10.0.0.29) 0.086 ms !N 0.038 ms !N *
```

Executando o comando `traceroute 192.168.0.194` é possível perceber que o problema está no "n5", identificado pelo IP 10.0.0.29.

Utilizando o comando `netstat -rn` nesse mesmo host, é possível perceber que não existe uma rota para encaminhar tráfego com destino a IPs 192.168.0.192/29:

```
1 > netstat -rn
2 Kernel IP routing table
3 Destination      Gateway           Genmask          Flags   MSS Window  irtt Iface
4 10.0.0.0           10.0.0.25        255.255.255.252 UG      0 0      0 eth1
5 10.0.0.4           10.0.0.25        255.255.255.252 UG      0 0      0 eth1
6 10.0.0.8           10.0.0.25        255.255.255.252 UG      0 0      0 eth1
7 10.0.0.12          10.0.0.25        255.255.255.252 UG      0 0      0 eth1
8 10.0.0.16          10.0.0.25        255.255.255.252 UG      0 0      0 eth1
9 10.0.0.20          10.0.0.25        255.255.255.252 UG      0 0      0 eth1
10 10.0.0.24          0.0.0.0          255.255.255.252 U        0 0      0 eth1
11 10.0.0.28          0.0.0.0          255.255.255.252 U        0 0      0 eth0
12 172.0.0.0          10.0.0.30        255.0.0.0        UG      0 0      0 eth0
13 172.16.142.0       10.0.0.25        255.255.255.248 UG      0 0      0 eth1
14 172.16.143.0       10.0.0.30        255.255.255.252 UG      0 0      0 eth0
15 172.16.143.0       10.0.0.30        255.255.255.248 UG      0 0      0 eth0
16 172.16.143.4       10.0.0.30        255.255.255.252 UG      0 0      0 eth0
17 192.142.0.4        10.0.0.25        255.255.255.252 UG      0 0      0 eth1
18 192.168.0.200      10.0.0.25        255.255.255.248 UG      0 0      0 eth1
19 192.168.0.208      10.0.0.25        255.255.255.248 UG      0 0      0 eth1
20 192.168.0.216      10.0.0.25        255.255.255.248 UG      0 0      0 eth1
21 192.168.0.224      10.0.0.30        255.255.255.248 UG      0 0      0 eth0
22 192.168.0.232      10.0.0.30        255.255.255.248 UG      0 0      0 eth0
23 192.168.0.240      10.0.0.30        255.255.255.248 UG      0 0      0 eth0
24 192.168.0.248      10.0.0.30        255.255.255.248 UG      0 0      0 eth0
```

Devemos, assim, adicionar essa rota com o comando: `ip route add 192.168.0.192/29 via 10.0.0.25`

No entanto, o problema de comunicação entre AfonsoHenriques e Teresa mantém-se, tal como é possível inferir através do seguinte comando:

```
1 > traceroute 192.168.0.194
2 traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
3 1 192.168.0.225 (192.168.0.225) 0.061 ms 0.037 ms 0.015 ms
4 2 172.16.143.1 (172.16.143.1) 0.035 ms 0.024 ms 0.023 ms
5 3 10.0.0.29 (10.0.0.29) 0.045 ms 0.032 ms 0.032 ms
```

```

6 4 10.0.0.25 (10.0.0.25) 0.052 ms 0.041 ms 0.040 ms
7 5 10.0.0.25 (10.0.0.25) 3055.726 ms !H 3055.655 ms !H 3055.602 ms !H

```

Agora o problema encontra-se no "n2". Acontece que a rota 192.168.0.194/31 dá match em vez da rota 192.168.0.192/29. Isto deve-se à forma como o match se dá, usando "Longest prefix match". Para resolver o problema é só remover a entrada 192.168.0.194/31 com o comando `ip route del 192.168.0.194/31`

No entanto, permanecemos ainda com outro problema. O tráfego encaminhado pelo "n2" para o "n1" é novamente reencaminhado para o "n2". Criando um loop de reencaminhamentos, que termina quando o TTL dos pacotes é igual a 0. Para isso, é necessário alterar a forma como o "n1" encaminha o tráfego com destino a Teresa. Em baixo, está a tabela de encaminhamento do "n1":

```

1 > netstat -rn
2 Kernel IP routing table
3 Destination      Gateway           Genmask           Flags        MSS Window  irtt  Iface
4 10.0.0.0          10.0.0.9         255.255.255.252  UG           0 0        0     eth0
5 10.0.0.4          10.0.0.9         255.255.255.252  UG           0 0        0     eth0
6 10.0.0.8          0.0.0.0          255.255.255.252  U            0 0        0     eth0
7 10.0.0.12         0.0.0.0          255.255.255.252  U            0 0        0     eth1
8 10.0.0.16         10.0.0.9         255.255.255.252  UG           0 0        0     eth0
9 10.0.0.20         10.0.0.14        255.255.255.252  UG           0 0        0     eth1
10 10.0.0.24         10.0.0.14        255.255.255.252  UG           0 0        0     eth1
11 10.0.0.28         10.0.0.14        255.255.255.252  UG           0 0        0     eth1
12 172.0.0.0         10.0.0.14        255.0.0.0        UG           0 0        0     eth1
13 172.16.142.0     10.0.0.9         255.255.255.252  UG           0 0        0     eth0
14 172.16.142.4     10.0.0.9         255.255.255.252  UG           0 0        0     eth0
15 172.16.143.0     10.0.0.14        255.255.255.252  UG           0 0        0     eth1
16 172.16.143.4     10.0.0.14        255.255.255.252  UG           0 0        0     eth1
17 192.168.0.192    10.0.0.14        255.255.255.248  UG           0 0        0     eth1
18 192.168.0.200    10.0.0.9         255.255.255.248  UG           0 0        0     eth0
19 192.168.0.208    10.0.0.9         255.255.255.248  UG           0 0        0     eth0
20 192.168.0.216    10.0.0.9         255.255.255.248  UG           0 0        0     eth0
21 192.168.0.224    10.0.0.14        255.255.255.248  UG           0 0        0     eth1
22 192.168.0.232    10.0.0.14        255.255.255.248  UG           0 0        0     eth1
23 192.168.0.240    10.0.0.14        255.255.255.248  UG           0 0        0     eth1
24 192.168.0.248    10.0.0.14        255.255.255.248  UG           0 0        0     eth1

```

É possível perceber através da linha 17, que o tráfego é reencaminhado novamente para o "n2"

Portanto, basta alterar esta entrada e definir o Gateway para o "n3". Executando os seguintes comandos:

```

1 > ip route del 192.168.0.192/29
2 > ip route add 192.168.0.192/29 via 10.0.0.9

```

Após isto, no caminho para a Teresa, o pacote já consegue chegar ao CondadOnline.

4.4 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.AfonsoHenriques? Efetue as alterações necessárias para garantir que a conectividade é restabelecida e o confronto entre os dois é evitado.

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? (Sugestão: analise as rotas nos dois sentidos com o `tracert`). Mostre graficamente a rota seguida nos dois sentidos por esses pacotes ICMP.

4.4.1 Resposta alínea d.i)

É possível constatar, executando o comando `tracert 192.168.0.194`, que, a partir, do CondadOnline não existiu uma resposta de volta.

```

1 > tracert 192.168.0.194
2 tracert to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
3 1 192.168.0.225 (192.168.0.225) 0.073 ms 0.021 ms 0.017 ms

```

```

4 2 172.16.143.1 (172.16.143.1) 0.039 ms 0.027 ms 0.026 ms
5 3 10.0.0.29 (10.0.0.29) 0.044 ms 0.034 ms 0.074 ms
6 4 10.0.0.25 (10.0.0.25) 0.067 ms 0.043 ms 0.074 ms
7 5 10.0.0.13 (10.0.0.13) 0.073 ms 0.050 ms 0.050 ms
8 6 10.0.0.17 (10.0.0.17) 0.086 ms 0.076 ms 0.059 ms
9 7 10.0.0.5 (10.0.0.5) 0.117 ms 0.071 ms 0.068 ms
10 8 10.0.0.1 (10.0.0.1) 0.140 ms 0.084 ms 0.075 ms
11 9 * * *
12 10 * * *
13 11 * * *
14 12 * * *
15 13 * * *

```

Se realizarmos o processo inverso, ou seja, traceroute de Teresa para AfonsoHenriques percebemos que os pacotes não passam do RAGaliza.

```

1 > traceroute 192.168.0.226
2 traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
3 1 192.168.0.193 (192.168.0.193) 0.076 ms !N 0.019 ms !N *

```

De facto, não existe nenhuma rota que encaminhe tráfego com destino aos IPs 192.168.0.224/29.

```

1 > netstat -rn
2 Kernel IP routing table
3 Destination      Gateway           Genmask           Flags      MSS Window  irtt Iface
4 10.0.0.0          172.16.142.1     255.255.255.252   UG          0 0        0 eth0
5 10.0.0.4          172.16.142.1     255.255.255.252   UG          0 0        0 eth0
6 10.0.0.8          172.16.142.1     255.255.255.252   UG          0 0        0 eth0
7 10.0.0.12         172.16.142.1     255.255.255.252   UG          0 0        0 eth0
8 10.0.0.16         172.16.142.1     255.255.255.252   UG          0 0        0 eth0
9 10.0.0.20         172.16.142.1     255.255.255.252   UG          0 0        0 eth0
10 10.0.0.24         172.16.142.1     255.255.255.252   UG          0 0        0 eth0
11 10.0.0.28         172.16.142.1     255.255.255.252   UG          0 0        0 eth0
12 172.0.0.0         172.16.142.1     255.0.0.0         UG          0 0        0 eth0
13 172.16.142.0      0.0.0.0          255.255.255.252   U           0 0        0 eth0
14 172.16.142.4      172.16.142.1     255.255.255.252   UG          0 0        0 eth0
15 172.16.143.0      172.16.142.1     255.255.255.252   UG          0 0        0 eth0
16 172.16.143.4      172.16.142.1     255.255.255.252   UG          0 0        0 eth0
17 192.168.0.192     0.0.0.0          255.255.255.248   U           0 0        0 eth1
18 192.168.0.200     172.16.142.1     255.255.255.248   UG          0 0        0 eth0
19 192.168.0.208     172.16.142.1     255.255.255.248   UG          0 0        0 eth0
20 192.168.0.216     172.16.142.1     255.255.255.248   UG          0 0        0 eth0
21 192.168.0.232     172.16.142.1     255.255.255.248   UG          0 0        0 eth0
22 192.168.0.240     172.16.142.1     255.255.255.248   UG          0 0        0 eth0
23 192.168.0.248     172.16.142.1     255.255.255.248   UG          0 0        0 eth0

```

Assim sendo, é necessário adicioná-la. Simplesmente executando o comando: `ip route add 192.168.0.224/29 via 172.16.142.1`

4.4.2 Resposta alínea d.ii)

No sentido AfonsoHenriques - Teresa:

```

1 > traceroute 192.168.0.194
2 traceroute to 192.168.0.194 (192.168.0.194), 30 hops max, 60 byte packets
3 1 192.168.0.225 (192.168.0.225) 0.050 ms 0.014 ms 0.011 ms
4 2 172.16.143.1 (172.16.143.1) 0.028 ms 0.017 ms 0.016 ms
5 3 10.0.0.29 (10.0.0.29) 0.036 ms 0.021 ms 0.022 ms
6 4 10.0.0.25 (10.0.0.25) 0.038 ms 0.028 ms 0.026 ms
7 5 10.0.0.13 (10.0.0.13) 0.044 ms 0.033 ms 0.033 ms
8 6 10.0.0.17 (10.0.0.17) 0.057 ms 0.064 ms 0.042 ms
9 7 10.0.0.5 (10.0.0.5) 0.058 ms 0.064 ms 0.051 ms
10 8 10.0.0.1 (10.0.0.1) 0.062 ms 0.050 ms 0.049 ms
11 9 172.16.142.2 (172.16.142.2) 0.067 ms 0.057 ms 0.055 ms
12 10 192.168.0.194 (192.168.0.194) 0.074 ms 0.064 ms 0.062 ms

```

No sentido Teresa - AfonsoHenriques:

```

1 > traceroute 192.168.0.226
2 traceroute to 192.168.0.226 (192.168.0.226), 30 hops max, 60 byte packets
3 1 192.168.0.193 (192.168.0.193) 0.092 ms 0.011 ms 0.009 ms
4 2 172.16.142.1 (172.16.142.1) 0.032 ms 0.016 ms 0.037 ms

```



```

5 3 10.0.0.2 (10.0.0.2) 0.027 ms 0.018 ms 0.017 ms
6 4 10.0.0.6 (10.0.0.6) 0.030 ms 0.022 ms 0.024 ms
7 5 10.0.0.18 (10.0.0.18) 0.058 ms 0.041 ms 0.092 ms
8 6 10.0.0.14 (10.0.0.14) 0.070 ms 0.050 ms 0.033 ms
9 7 10.0.0.26 (10.0.0.26) 0.046 ms 0.036 ms 0.037 ms
10 8 10.0.0.30 (10.0.0.30) 0.049 ms 0.041 ms 0.061 ms
11 9 172.16.143.2 (172.16.143.2) 0.058 ms 0.047 ms 0.045 ms
12 10 192.168.0.226 (192.168.0.226) 0.059 ms 0.051 ms 0.050 ms

```

Sim, pode-se afirmar que a rota seguida é exatamente a mesma nos dois sentidos. Os IPs diferem sempre num bit (pois existe um IP para saída e um para entrada).

4.5 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.

4.5.1 Resposta alínea e)

As subredes Galiza e CDN têm entradas na tabela de encaminhamento com "matches" mais longos. Assim sendo, o router irá optar por essas rotas. Caso essas entradas não existissem, o tráfego não iria ser encaminhado corretamente, levando a problemas na rede.

4.6 Alínea f)

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

4.6.1 Resposta alínea f)

Os endereços são todos privados. Isto pois, fazem todos parte do "grupo" de endereços privados reservados - 192.168.0.0/24 ; 172.16.0.0/12 ; 10.0.0.0/8

4.7 Alínea g)

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

4.7.1 Resposta alínea g)

Os switches são dispositivos que operam no nível 2 (nível de dados) e são responsáveis por encaminhar o tráfego entre diferentes dispositivos conectados à rede. Ao contrário de dispositivos de nível 3, como os routers, os switches não precisam de endereços IP. Eles utilizam endereços MAC para identificar dispositivos conectados a eles e encaminhar o tráfego pela rede.

5 Parte II - Exercício 2

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

5.1 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.

5.1.1 Resposta alínea a)

Primeiramente, apagamos a rota default do Castelo, executando o comando: `ip route del default`

Após isso, foram adicionadas as seguintes rotas:

```
1 > netstat -rn
2 Kernel IP routing table
3 Destination      Gateway            Genmask           Flags      MSS Window  irtt Iface
4 192.168.0.192     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
5 192.168.0.200     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
6 192.168.0.208     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
7 192.168.0.216     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
8 192.168.0.224     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
9 192.168.0.224     0.0.0.0           255.255.255.248  U           0  0        0 eth0
10 192.168.0.232     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
11 192.168.0.240     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
12 192.168.0.248     192.168.0.225     255.255.255.248  UG          0  0        0 eth0
13
```

Por último foram feitos os testes de maneira a descobrir se este realmente estava conectado a todos os polos.

```
root@Castelo:/tmp/pycore.35085/Castelo.conf# ping 192.168.0.196
PING 192.168.0.196 (192.168.0.196) 56(84) bytes of data.
64 bytes from 192.168.0.196: icmp_seq=1 ttl=55 time=0.222 ms
64 bytes from 192.168.0.196: icmp_seq=2 ttl=55 time=0.159 ms
64 bytes from 192.168.0.196: icmp_seq=3 ttl=55 time=0.101 ms
^C
--- 192.168.0.196 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 0.101/0.160/0.222/0.049 ms
```

Figura 5: Conexão Galiza

```

root@Castelo:/tmp/pycore.35085/Castelo.conf# ping 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=0.166 ms
64 bytes from 192.168.0.210: icmp_seq=2 ttl=55 time=0.138 ms
64 bytes from 192.168.0.210: icmp_seq=3 ttl=55 time=0.142 ms
64 bytes from 192.168.0.210: icmp_seq=4 ttl=55 time=0.133 ms
^C
--- 192.168.0.210 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3076ms
rtt min/avg/max/mdev = 0.133/0.144/0.166/0.012 ms

```

Figura 6: Conexão CDN

```

root@Castelo:/tmp/pycore.35085/Castelo.conf# ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=61 time=0.143 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=61 time=0.083 ms
64 bytes from 192.168.0.234: icmp_seq=3 ttl=61 time=0.075 ms
^C
--- 192.168.0.234 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
rtt min/avg/max/mdev = 0.075/0.100/0.143/0.030 ms

```

Figura 7: Conexão Instucional

Retirando a *default* route necessitamos de reconectar o *host* Castelo a todos os outros polos manualmente. Desta maneira no futuro, limitando nos a não ter a *default* route, teremos problemas de conexão com futuras redes que sejam adicionadas.

5.2 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.

5.2.1 Resposta alínea b)

Para o endereço IP 172.16.12.128/26 temos um total de 64 endereços, como queremos pelo menos 3 redes precisamos de pelo menos 2 bits de maneira a representá-las, assim a máscara deixará de ser /26 e passará a ser /28, e por consequência de utilizarmos 2 bits acabamos com 4 redes. Dividindo o número de endereços por cada rede:

$$64/4 = 16$$

Concluimos que cada rede terá 16 endereços, 14 usáveis.

- 1ª Rede: 172.16.12.128/28
Range: 172.16.12.129 - 172.16.12.142
Broadcast: 172.16.12.143
- 2ª Rede: 172.16.12.144/28

Range: 172.16.12.145 - 172.16.12.158

Broadcast: 172.16.12.159

- 3ª Rede: 172.16.12.160/28

Range: 172.16.12.161 - 172.16.12.173

Broadcast: 172.16.12.175

- 4ª Rede: 172.16.12.176/28

Range: 172.16.12.177 - 172.16.12.190

Broadcast: 172.16.12.191

5.3 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?

5.3.1 Resposta alínea c)

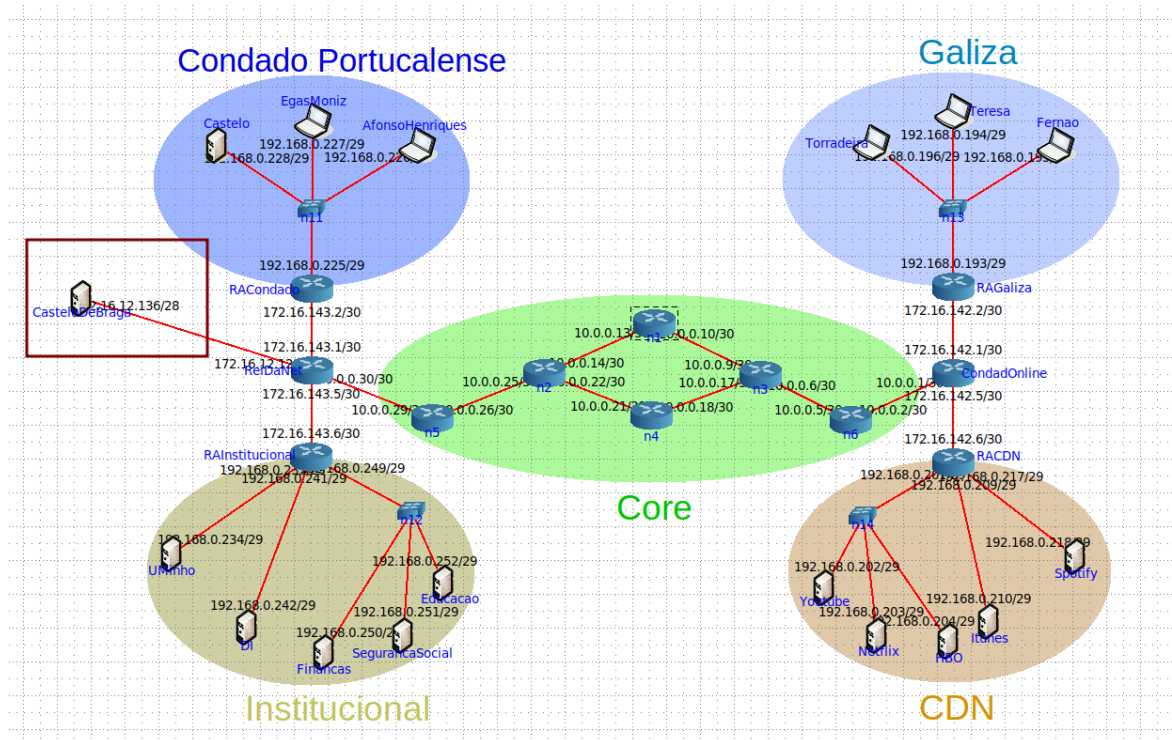


Figura 8: Topologia do exercício

Ligamos um host ao ReiDaNet, associando-lhe o endereço 172.16.12.129/28. Em baixo, segue a demonstração da ligação, com sucesso, a todos os polos.

```

root@CasteloDeBraga:/tmp/pycore.35085/CasteloDeBraga.conf# ping 192.168.0.227
PING 192.168.0.227 (192.168.0.227) 56(84) bytes of data.
64 bytes from 192.168.0.227: icmp_seq=1 ttl=62 time=0.072 ms
64 bytes from 192.168.0.227: icmp_seq=2 ttl=62 time=0.049 ms
64 bytes from 192.168.0.227: icmp_seq=3 ttl=62 time=0.069 ms
^C
--- 192.168.0.227 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2030ms
rtt min/avg/max/mdev = 0.049/0.063/0.072/0.010 ms
root@CasteloDeBraga:/tmp/pycore.35085/CasteloDeBraga.conf# ping 192.168.0.202
PING 192.168.0.202 (192.168.0.202) 56(84) bytes of data.
64 bytes from 192.168.0.202: icmp_seq=1 ttl=56 time=0.208 ms
64 bytes from 192.168.0.202: icmp_seq=2 ttl=56 time=0.220 ms
64 bytes from 192.168.0.202: icmp_seq=3 ttl=56 time=0.208 ms
^C
--- 192.168.0.202 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2056ms
rtt min/avg/max/mdev = 0.208/0.212/0.220/0.005 ms
root@CasteloDeBraga:/tmp/pycore.35085/CasteloDeBraga.conf# ping 192.168.0.242
PING 192.168.0.242 (192.168.0.242) 56(84) bytes of data.
64 bytes from 192.168.0.242: icmp_seq=1 ttl=62 time=0.053 ms
64 bytes from 192.168.0.242: icmp_seq=2 ttl=62 time=0.102 ms
64 bytes from 192.168.0.242: icmp_seq=3 ttl=62 time=0.108 ms
^C
--- 192.168.0.242 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
rtt min/avg/max/mdev = 0.053/0.087/0.108/0.024 ms
root@CasteloDeBraga:/tmp/pycore.35085/CasteloDeBraga.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=56 time=0.192 ms
64 bytes from 192.168.0.194: icmp_seq=2 ttl=56 time=0.115 ms
64 bytes from 192.168.0.194: icmp_seq=3 ttl=56 time=0.207 ms
^C
--- 192.168.0.194 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.115/0.171/0.207/0.040 ms

```

Figura 9: Conexão, respetivamente, com o Condado, o CDN, o Instituto e a Galiza

6 Parte II - Exercício 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.

6.1 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

6.1.1 Resposta alínea a)

Executando os comandos no dispositivo n6:

```

1 > ip route del 192.168.0.192/29
2 > ip route del 192.168.0.200/29
3 > ip route del 192.168.0.208/29
4 > ip route del 192.168.0.216/29

```

Conseguimos remover as rotas referentes a Galiza e CDN.

Temos agora que encontrar o IP novo que irá agregar todos os IPs anteriores num só.
Listando-os todos e convertendo para binário o padrão que é incomum nas subredes:

```
1 192.168.0.11000000/29
2 192.168.0.11001000/29
3 192.168.0.11010000/29
4 192.168.0.11011000/29
```

Determinar onde a primeira mudança ocorre:

```
1 192.168.0.110|00|000/29
2 192.168.0.110|01|000/29
3 192.168.0.110|10|000/29
4 192.168.0.110|11|000/29
```

Converter IP:

```
1 192.168.0.11000000/27
```

ou seja

```
1 192.168.0.192/27
```

Para finalizar, basta adicionar uma rota no IP 192.168.0.192/27 via 10.0.0.1, e então conseguimos fazer com que exista apenas uma rota para ambos os polos.

```
1 > ip route add 192.168.0.192/27 via 10.0.0.1
```

```
root@n6:/tmp/pycore.1/n6.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=62 time=0.046 ms
64 bytes from 192.168.0.194: icmp_seq=2 ttl=62 time=0.073 ms
```

Figura 10: Conectividade a Teresa, Galiza.

```
root@n6:/tmp/pycore.1/n6.conf# ping 192.168.0.204
PING 192.168.0.204 (192.168.0.204) 56(84) bytes of data.
64 bytes from 192.168.0.204: icmp_seq=1 ttl=62 time=0.149 ms
64 bytes from 192.168.0.204: icmp_seq=2 ttl=62 time=0.034 ms
```

Figura 11: Conectividade a HBO, CDN.


```

root@n6:/tmp/pycore.1/n6.conf# ip route
10.0.0.0/30 dev eth0 proto kernel scope link src 10.0.0.2
10.0.0.4/30 dev eth1 proto kernel scope link src 10.0.0.5
10.0.0.8/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.12/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.16/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.20/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.24/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.28/30 via 10.0.0.6 dev eth1 proto zebra
172.0.0.0/8 via 10.0.0.6 dev eth1 proto zebra
172.16.142.0/30 via 10.0.0.1 dev eth0 proto zebra
172.16.142.4/30 via 10.0.0.1 dev eth0 proto zebra
172.16.143.0/30 via 10.0.0.6 dev eth1 proto zebra
172.16.143.4/30 via 10.0.0.6 dev eth1 proto zebra
192.168.0.192/27 via 10.0.0.1 dev eth0
192.168.0.224/29 via 10.0.0.6 dev eth1 proto zebra
192.168.0.232/29 via 10.0.0.6 dev eth1 proto zebra
192.168.0.240/29 via 10.0.0.6 dev eth1 proto zebra
192.168.0.248/29 via 10.0.0.6 dev eth1 proto zebra

```

Figura 12: Route Table

6.2 Alínea b)

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

6.2.1 Resposta alínea b)

Fazendo o mesmo processo da alínea anterior só que para os IPs do Condado e do Institucional:

```

1 > ip route del 192.168.0.224/29
2 > ip route del 192.168.0.232/29
3 > ip route del 192.168.0.240/29
4 > ip route del 192.168.0.248/29

```

Executamos estes comandos para eliminar as rotas presentes. Aplicando o mesmo algoritmo para descobrir o novo IP da rota temos como resultado:

```

1 192.168.0.224/27

```

e então basta adicioná-lo como nova rota:

```

1 > ip route add 192.168.0.224/27 via 10.0.0.6

```

```

root@n6:/tmp/pycore.1/n6.conf# ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=58 time=0.122 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=58 time=0.175 ms

```

Figura 13: Dispositivo UMinho Instituto

```

root@n6:/tmp/pycore.1/n6.conf# ping 192.168.0.226
PING 192.168.0.226 (192.168.0.226) 56(84) bytes of data.
64 bytes from 192.168.0.226: icmp_seq=1 ttl=58 time=0.246 ms
64 bytes from 192.168.0.226: icmp_seq=2 ttl=58 time=0.126 ms
64 bytes from 192.168.0.226: icmp_seq=3 ttl=58 time=0.107 ms

```

Figura 14: Dispositivo Afonso Henriques Condado

```

root@n6:/tmp/pycore.1/n6.conf# ip route
10.0.0.0/30 dev eth0 proto kernel scope link src 10.0.0.2
10.0.0.4/30 dev eth1 proto kernel scope link src 10.0.0.5
10.0.0.8/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.12/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.16/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.20/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.24/30 via 10.0.0.6 dev eth1 proto zebra
10.0.0.28/30 via 10.0.0.6 dev eth1 proto zebra
172.0.0.0/8 via 10.0.0.6 dev eth1 proto zebra
172.16.142.0/30 via 10.0.0.1 dev eth0 proto zebra
172.16.142.4/30 via 10.0.0.1 dev eth0 proto zebra
172.16.143.0/30 via 10.0.0.6 dev eth1 proto zebra
172.16.143.4/30 via 10.0.0.6 dev eth1 proto zebra
192.168.0.192/27 via 10.0.0.1 dev eth0
192.168.0.224/27 via 10.0.0.6 dev eth1

```

Figura 15: Route Table

6.3 Alínea c)

Comente os aspetos positivos e negativos do uso do *Supernetting*.

6.3.1 Resposta alínea c)

Aspetos positivos:

- O tamanho da lista das rotas é menor, poupando tempo no CPU e memória.
- Abstrai topologias de outros routers, fazendo com que, quando essas alteradas, as rotas do nosso router não precisem de ser alteradas.

Aspetos negativos:

- Quanto maior a gama de IPs que uma entrada nas rotas do nosso router cobre, mais IPs podem não estar a ser usados, sendo assim desperdiçados