

Relatório TP2 | Redes de Computadores

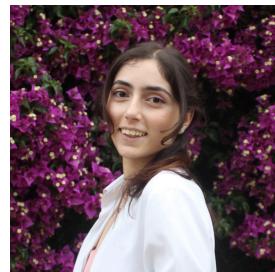
Grupo 31 | 2023/2024



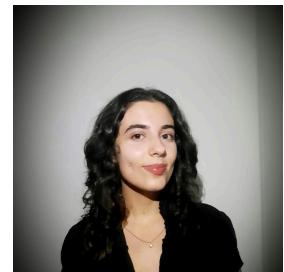
João Lobo
(A104356)



Mariana Rocha
(A90817)



Rita Camacho
(A104439)



1^a Parte

1.

Prepare uma topologia CORE para verificar o comportamento do **traceroute**. Na topologia deve existir: um *host* (pc) cliente designado **Jasmine**, cujo *router* de acesso é RA1; o *router* RA1 está simultaneamente ligado a dois *routers* no *core* da rede RCx e RCy (cada grupo de trabalho deve personalizar a rede de *core* fazendo *x* e *y* corresponder ao seu identificador de grupo PLxy); estes estão conectados a um *router* de acesso RA2, que por sua vez, se liga a um *host* (servidor) designado **Aladdin**. 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 20 ms. Após ativar a topologia, note que pode não existir conectividade IP imediata entre **Jasmine** e **Aladdin** pois é necessário que o anúncio de rotas entre *routers* se efetue e estabilize.

Documente e justifique todas as respostas às seguintes alíneas:

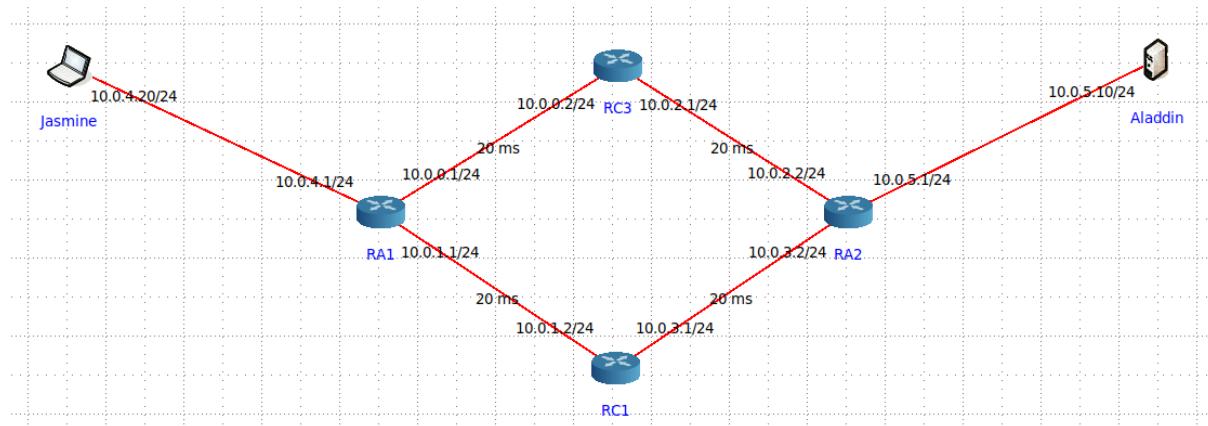


Figura 1: Topologia

1. a)

Active o wireshark no *host* **Jasmine**. Numa *shell* de **Jasmine** execute o comando **traceroute -I** para o endereço IP do **Aladdin**. Registe e analise o tráfego ICMP enviado pelo sistema **Jasmine** e o tráfego ICMP recebido como resposta. Explique os resultados obtidos tendo em conta o princípio de funcionamento do **traceroute**.

Ao executar o comando **traceroute -I** numa *shell* de **Jasmine** para o endereço do **Aladdin**, obtivemos os seguintes resultados:

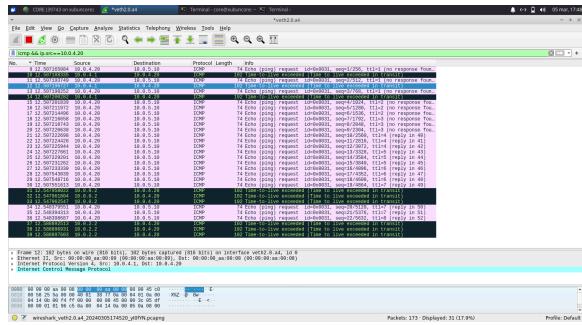


Figura 2: Tráfego ICMP enviado pelo sistema **Jasmine**

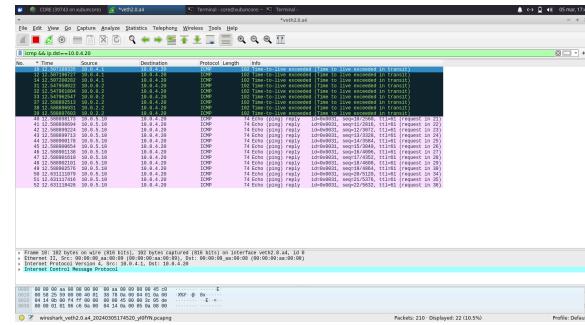


Figura 3: Tráfego ICMP recebido como resposta

O comando `traceroute -I 10.0.5.10` foi utilizado para acompanhar o caminho dos pacotes IP enviados pelo sistema **Jasmine** (**10.0.4.20**) para o **Aladdin** (**10.0.5.10**).

Inicialmente, **Jasmine** enviou um pedido de *echo (ping)* para **Aladdin** com TTL de 1. No entanto, este pacote teve *Time-to-live exceeded*, pois expirou no *router RA1*. Também podemos observar o sucedido no nº 31 com o *router RC3*.

À medida que o TTL foi incrementado, os pacotes alcançaram **Aladdin** e **Jasmine** começou a receber respostas de *Echo reply* de **Aladdin**, indicando que os pacotes alcançaram com sucesso o destino. Podemos observar que o `traceroute` mostra os *routers* usados ao longo do caminho, onde os pacotes ICMP são recebidos e onde os TTLs expiram, podendo ajudar a identificar possíveis falhas na rede.

Concluindo, estes resultados mostram como o `traceroute` utiliza o TTL e o tráfego ICMP para compreender a rota dos pacotes.

1. b)

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

Através da análise do *output* da execução do comando `traceroute -I <ip-aladdin>` no terminal do sistema **Jasmine**, concluímos que o valor mínimo do campo TTL deve ser 4. Comprovámos este valor através de sucessivas execuções do `ping`, incrementando em cada iteração o valor de *hops* máximo, onde se observa que, chegando a 4, o `ping` passa a receber uma resposta.

```
root@Jasmine:/tmp/pycore.39743/Jasmine.conf# traceroute -I 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1  10.0.4.1 (10.0.4.1)  0.041 ms  0.005 ms  0.004 ms
 2  10.0.0.2 (10.0.0.2)  41.033 ms  41.025 ms  41.021 ms
 3  10.0.2.2 (10.0.2.2)  81.910 ms  81.909 ms  81.907 ms
 4  10.0.5.10 (10.0.5.10)  81.905 ms  81.904 ms  81.902 ms
```

Figura 4: Execução do `traceroute`

```

CORE (39743 on xubuncore) *veth2.0.4 Terminal - core@xubuncore: ~ Terminal - 05 mar, 17:52
File Edit View Terminal Tabs Help Terminal -
From 10.0.4.1 icmp seq=5 Time to live exceeded
From 10.0.4.1 icmp seq=6 Time to live exceeded
From 10.0.4.1 icmp seq=7 Time to live exceeded
^C
-- 10.0.5.10 ping statistics ...
7 packets transmitted, 0 received, +7 errors, 100% packet loss, time 6134ms
root@Jasmine:/tmp/pycore.39743/Jasmine.conf# ping 10.0.5.10 -t 2
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
From 10.0.0.2 icmp seq=1 Time to live exceeded
From 10.0.0.2 icmp seq=2 Time to live exceeded
From 10.0.0.2 icmp seq=3 Time to live exceeded
^C
-- 10.0.5.10 ping statistics ...
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2004ms
root@Jasmine:/tmp/pycore.39743/Jasmine.conf# ping 10.0.5.10 -t 3
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
From 10.0.2.2 icmp seq=1 Time to live exceeded
From 10.0.2.2 icmp seq=2 Time to live exceeded
From 10.0.2.2 icmp seq=3 Time to live exceeded
From 10.0.2.2 icmp seq=4 Time to live exceeded
^C
-- 10.0.5.10 ping statistics ...
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3005ms
root@Jasmine:/tmp/pycore.39743/Jasmine.conf# ping 10.0.5.10 -t 4
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=1 ttl=61 time=82.0 ms
64 bytes from 10.0.5.10: icmp_seq=2 ttl=61 time=81.6 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=61 time=82.1 ms
64 bytes from 10.0.5.10: icmp_seq=4 ttl=61 time=81.9 ms
64 bytes from 10.0.5.10: icmp_seq=5 ttl=61 time=81.6 ms
^C
-- 10.0.5.10 ping statistics ...
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 81.584/81.836/82.122/0.211 ms
root@Jasmine:/tmp/pycore.39743/Jasmine.conf#

```

Figura 5: Sucessivas execuções do comando `ping`

1. 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`.

Valor Médio RTT utilizando 10 pacotes = **82.9532 ms**

```

Terminal -
File Edit View Terminal Tabs Help Terminal -
root@Jasmine:/tmp/pycore.39743/Jasmine.conf# traceroute -I 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
1 10.0.4.1 (10.0.4.1) 0.041 ms 0.005 ms 0.004 ms
2 10.0.0.2 (10.0.0.2) 41.033 ms 41.025 ms 41.021 ms
3 10.0.2.2 (10.0.2.2) 81.910 ms 81.909 ms 81.907 ms
4 10.0.5.10 (10.0.5.10) 81.995 ms 81.994 ms 81.902 ms
root@Jasmine:/tmp/pycore.39743/Jasmine.conf# traceroute -I 10.0.5.10 -q 10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
1 10.0.4.1 (10.0.4.1) 0.042 ms 0.006 ms 0.005 ms 0.004 ms 0.004 ms 0.004 ms * * * *
2 10.0.0.2 (10.0.0.2) 41.872 ms 41.865 ms 41.863 ms 41.861 ms 41.861 ms 41.859 ms * * * *
3 10.0.2.2 (10.0.2.2) 83.090 ms 83.089 ms 81.913 ms 81.886 ms 81.884 ms 81.882 ms * * * *
4 10.0.5.10 (10.0.5.10) 82.589 ms 82.586 ms 84.316 ms 84.295 ms 82.285 ms 82.265 ms 82.260 ms 82.258 ms 83.350 ms 83.328 ms
root@Jasmine:/tmp/pycore.39743/Jasmine.conf#

```

Figura 6: Valores utilizados no cálculo presentes no índice 4

1. 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?

Não é possível calcular com precisão o valor médio do atraso num sentido dividindo o RTT por dois. Isto ocorre devido à complexidade das redes reais, onde os atrasos podem variar devido a diversos fatores, tais como congestionamento da rede, processamento nos *routers* e assimetria no encaminhamento dos pacotes. Além disso, diversos componentes - atraso de propagação, processamento, fila e codificação - podem variar ao longo do tempo, aumentando a dificuldade no cálculo do atraso num sentido de forma precisa.

De modo que, enquanto o cálculo do atraso num sentido usado RTT dividido por dois pode ser uma abordagem simplificada, esta não tem em conta a complexidade de uma rede real, sendo necessárias abordagens mais avançadas.

2.

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

Windows. O programa *tracert* disponibilizado no Windows não permite mudar o tamanho das mensagens a enviar. Como alternativa, o programa *pingplotter* (ou equivalente) na sua versão livre ou shareware (<http://www.pingplotter.com>) permite maior flexibilidade para efetuar **traceroute**. Descarregue, instale e experimente o *pingplotter* face ao objectivo pretendido.

O tamanho da mensagem a enviar (*ICMP Echo Request*) pode ser estabelecido no *pingplotter* no menu *Edit -> Options -> Default Settings -> Engine*. Uma vez enviado um conjunto de pacotes com valores crescentes de TTL, o programa recomeça com TTL=1, após um determinado intervalo. Tanto o valor do intervalo de tempo como o número de intervalos podem ser configurados.

Linux/Unix. O comando **traceroute** permite indicar o tamanho do pacote ICMP (opção **-I**) através da linha de comando, a seguir ao *host* de destino (ver man **traceroute**).

Documente as suas respostas com a impressão do(s) output(s) (e.g. pacote(s)) que as suportam.

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 (expanda o *tab* correspondente na janela de detalhe do *wireshark*).

2. a)

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

O endereço IP da interface ativa é o 172.26.104.182, no entanto nas capturas de ecrã que serão apresentadas, o endereço IP exibido é 10.0.2.15, uma vez que estamos a utilizar um ambiente de virtualização *VirtualBox*.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\lal040439\Desktop> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet 3:
  Connection-specific DNS Suffix . .
  Link-local IPv6 Address . . . . . fe80::a2d8:a8ae%2025:818%0
  IPv4 Address . . . . . 172.26.254.182
  Subnet Mask . . . . . 255.255.255.0
  Default Gateway . . . . .

Wireless LAN adapter Ligação de Área Local* 1:
  Media State . . . . . Media disconnected
  Connection-specific DNS Suffix . .
  Wireless LAN adapter Ligação de Área Local* 2:
  Media State . . . . . Media disconnected
  Connection-specific DNS Suffix . .
  Wireless LAN adapter Wi-Fi:
  Connection-specific DNS Suffix . . eduvnas.unisnho.pt
  Link-local IPv6 Address . . . . . fe80::6d07:280e%5f8e:aaeh%16
  IPv4 Address . . . . . 172.26.254.8
  Subnet Mask . . . . . 255.255.255.0
  Default Gateway . . . . . 172.26.254.254

Ethernet adapter Ligação de Rede Bluetooth 2:
  Media State . . . . . Media disconnected
  Connection-specific DNS Suffix . .

```

Figura 7: IP da interface ativa do computador

2. b)

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

Através da leitura dos campos no cabeçalho IP, é possível identificar o protocolo como sendo o **ICMP** (*Internet Control Message Protocol*), sendo que este tem valor 1 (**0x01**).

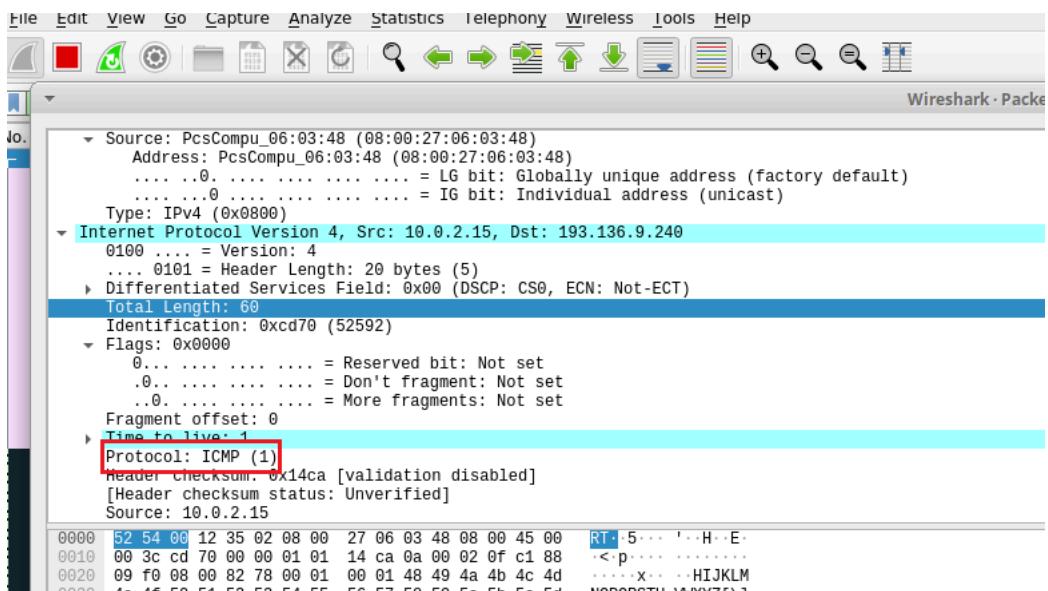


Figura 8: Protocolo ICMP

2. c)

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

O cabeçalho IPv4 possui **20 bytes**; já o campo de dados (*payload*) possui **40 bytes**, tendo sido calculado através de ***total length - cabeçalho = 60 - 20 = 40 bytes***.

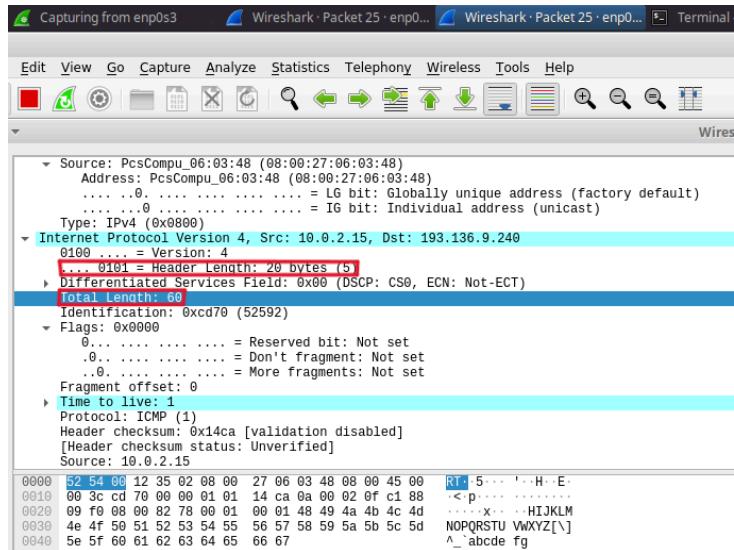


Figura 9: Tamanho em *bytes* do cabeçalho IPv4 e da *total length*

2. d)

O datagrama IP foi fragmentado? Justifique.

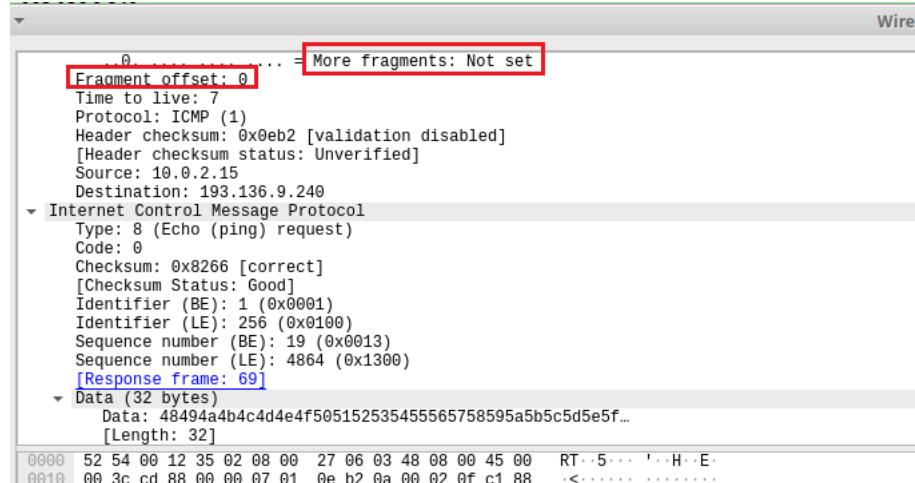


Figura 10: Parâmetros que permitiram concluir o questionado

O datagrama IP não foi fragmentado, pois, no cabeçalho do pacote, o *fragment offset* de valor 0, indica-nos que estamos a receber o início do *payload* e, como a *flag More fragments* não está definida, não faltam dados. Sendo assim, a informação conseguiu ser enviada sem recorrer a fragmentação, com apenas um pacote.

2. 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.

Ao ordenar os pacotes capturados de acordo com o endereço IP fonte, observámos que os campos do cabeçalho IP que variam de pacote para pacote são o TTL, o *checksum* e o *identification*. O TTL varia de 3 em 3 para limitar o número de saltos na rede garantindo a eficiência na entrega do pacote. O *checksum* varia devido à inclusão de diferentes informações nos pacotes, como o número de sequência ou *timestamp*, sendo necessário recalcular para cada pacote enviado para garantir a integridade dos dados. O *identification* é utilizado para identificar os diversos pacotes, tendo cada um deles um identificador diferente, dando a possibilidade ao utilizador de reconstruir a sequência correta dos pacotes recebidos.

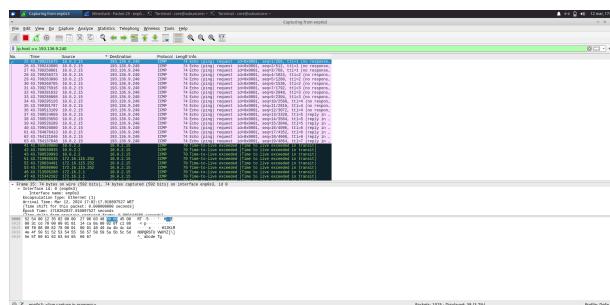


Figura 11: Tráfego ICMP gerado a partir do endereço IP ordenado pelo endereço IP fonte

2. f)

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

Concluímos que o IP se mantinha, mas o TTL incrementava até obter resposta.

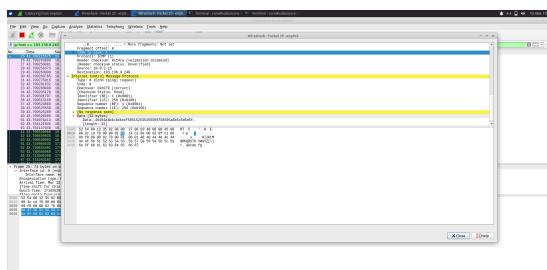


Figura 12: 1º Pacote

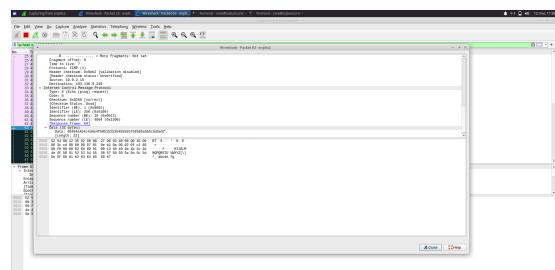


Figura 13: Último Pacote

2. g)

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

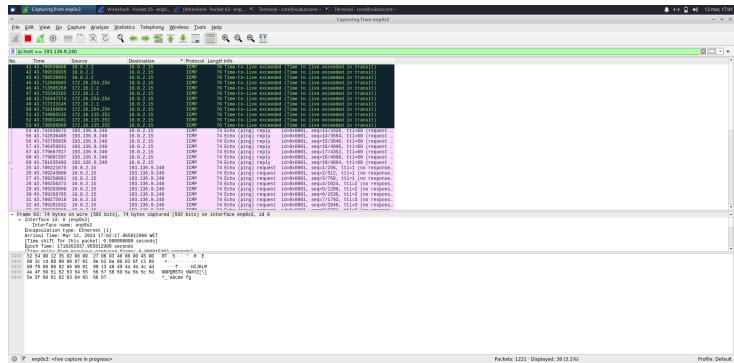


Figura 14: Tráfego ICMP gerado a partir do endereço IP ordenado pelo endereço IP destino

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 valor recebido no campo TTL varia entre os valores 253, 254 e 255, porque o TTL é decrementado por cada *router* ao longo do caminho e as respostas *ICMP TTL exceeded* podem seguir caminhos de retorno diferentes dos pacotes que as originaram, levando a valores diferentes de TTL.

ii)

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

As mensagens de resposta *ICMP TTL exceeded* são enviadas com um valor TTL relativamente alto na sua origem para garantir que possam ser devolvidas ao *host* de origem, independentemente do número de saltos que o pacote original fez na rede. Com isto, há uma grande garantia que a informação de expiração do TTL seja comunicada ao *host* de origem de forma eficaz.

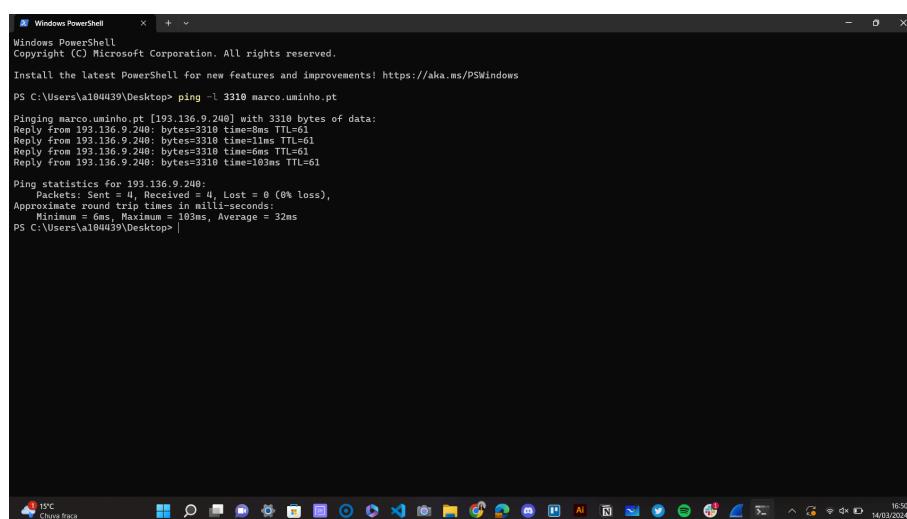
2. h)

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?

Teoricamente, é possível incluir a informação contida no cabeçalho ICMP no cabeçalho IPv4. No entanto, esse cenário terá várias implicações. Ao incluir informações do ICMP no cabeçalho IPv4, reduz-se a sobrecarga de cabeçalhos, uma vez que não seria necessário um cabeçalho adicional para o ICMP em cada pacote. Isto seria benéfico para redes com largura de banda limitada, ou em situações onde a eficiência é fraca. Ao integrar informações do ICMP diretamente no cabeçalho IPv4, otimizar-se-ia o processamento de pacotes, reduzindo a necessidade de análise adicional para lidar com informações de controlo. No entanto, como o cabeçalho IPv4 tem um tamanho fixo (20 bytes), ao adicionar informações do ICMP, este valor aumentaria (28 bytes). Os pacotes seriam maiores, podendo trazer implicações nas redes com limitações de MTU. Além disso, manter o ICMP separado do IPv4 oferece flexibilidade para a evolução independente de ambos os protocolos. Integrá-los pode limitar esta flexibilidade, tornando mais difícil para futuras atualizações e extensões.

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 (3xy0) bytes, em que xy é o número do seu grupo de trabalho (e.g., o grupo PL19 deve usar um tamanho de pacote de 3190 bytes). De modo a poder visualizar os fragmentos, aceda a *Edit → Preferences → Protocols* e em IPv4 desative a opção “*Reassemble fragmented IPv4 datagrams*”.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\al04439\Desktop> ping -l 3310 marco.uminho.pt

Pinging marco.uminho.pt [193.136.9.240] with 3310 bytes of data:
Reply from 193.136.9.240: bytes=3310 time=3ms TTL=61
Reply from 193.136.9.240: bytes=3310 time=11ms TTL=61
Reply from 193.136.9.240: bytes=3310 time=6ms TTL=61
Reply from 193.136.9.240: bytes=3310 time=10ms 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 = 6ms, Maximum = 10ms, Average = 32ms
PS C:\Users\al04439\Desktop>
```

Figura 15: ping -l 3310 marco.uminho.pt

3. a)

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

O envio do `ping` com tamanho de pacote 3310 *bytes* para `marco.uminho.pt` ultrapassa o tamanho máximo permitido pela rede, sendo este 1500 *bytes*. Neste caso, foi necessário fragmentar o pacote inicial em pedaços mais pequenos para que pudesse ser transmitido com sucesso pela rede.

3. 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?

```
Wireshark · Packet 205 · Wi-Fi
Internet Protocol Version 4, Src: 172.26.104.182, 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: 0x188e (6286)
    > 001. .... = Flags: 0x1, More fragments
        ...0 0000 0000 0000 = Fragment Offset: 0
        Time to Live: 128
        Protocol: ICMP (1)
        Header Checksum: 0x0000 [validation disabled]
        [Header checksum status: Unverified]
```

Figura 16: Primeiro fragmento do datagrama original

Verifica-se que o datagrama foi fragmentado pois, nas *flags* de valor `001`, o 3º bit tem valor `1` (`More Fragments`), indicando assim que o datagrama é composto por vários fragmentos. Este pacote trata-se do primeiro fragmento, pois têm o *Fragment Offset* a `0`. O tamanho deste datagrama é 1500 *bytes* (campo *Total Length*).

3. 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?

Não se trata do 1º fragmento, pois o *offset* é 1480, sendo diferente de 0 (no entanto, o valor do TTL é igual ao do primeiro fragmento, permitindo concluir que se trata do mesmo pacote embora fragmentado). Também através das *flags* é possível afirmar o mesmo, mas também que existem mais fragmentos após este 2º.

Figura 17: Parâmetros que permitiram concluir o questionado

3. 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*.

O tamanho máximo MTU é 1500 *bytes* e o tamanho total do pacote, representado por L, é 3310 *bytes*.

1º Passo: Verificar quantos dados podemos enviar (MTU - Cabeçalho):

$$1500 - 20 = 1480 \text{ bytes}$$

2º Passo: Verificar se é necessário fragmentar o pacote:

$$\text{nº fragmentos} = \text{floor}(L / \text{MTU}) + 1 = \text{floor}(3310 / 1500) + 1 = 2 + 1 = 3$$

É necessário fragmentar o pacote em 3 pacotes, todos com:

- Identificador: 0x188e (6286)
- TTL: 128

3º passo: Calcular o número de *bytes* transportados no último fragmento:

3.1º passo: 1º fragmento:

Temos que enviar 3310 *bytes* de dados e podemos enviar, no máximo, por pacote 1480 *bytes*.

$$L = 3310 - 1480 = 1830 \text{ bytes}$$

$$L_1 = 1480 + 20 = 1500 \text{ bytes}$$

$$\text{Fragment offset} = 0 \text{ bytes}$$

3.2º passo: 2º fragmento:

$$L = 1830 - 1480 = 350 \text{ bytes}$$

$$L_2 = 1480 + 20 = 1500 \text{ bytes}$$

3.3º passo: 3º fragmento (último):

$$L3^{\circ} = 350 + 20 = 370 \text{ bytes}$$

$$\text{Fragment offset} = 1480 \times 2 = 2960 \text{ bytes}$$

O número de pacotes fragmentados teórico correspondeu ao número de pacotes fragmentados observado pelo *wireshark* é o mesmo, indicando que a fragmentação ocorreu conforme o esperado e que a rede lidou corretamente com o tamanho dos pacotes. No entanto, o valor teórico do número de bytes transportados no último fragmento é 370 bytes e o valor visto no *wireshark* é de 378 bytes. Este acréscimo de 8 bytes deve-se ao cabeçalho ICMP.

3. 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.

Através das *flags*, concluímos que já não existem mais fragmentos (*bit a 0*), logo trata-se do último fragmento. Além disso, o valor do *offset* também nos permite afirmar o mesmo.

Utilizou-se o filtro (`ip.flags.mf == False`) && (`ip.flag.offset == 370`) de forma a listar apenas o último fragmento.

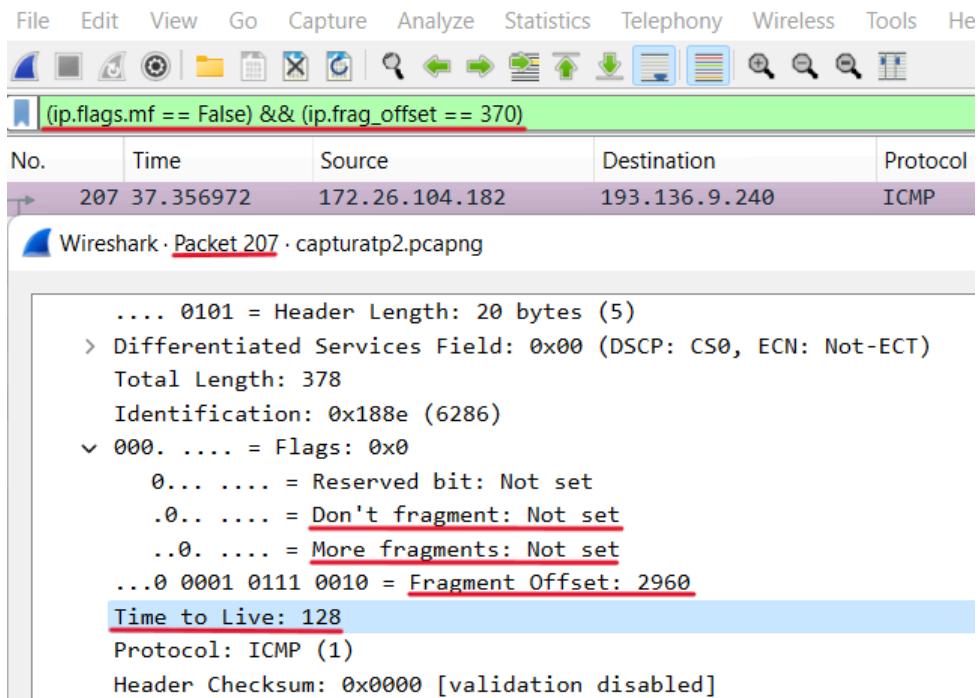


Figura 18: Parâmetros que permitiram concluir o questionado

3. 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 equipamento onde o datagrama IP original é reconstruído a partir dos fragmentos é o *host* do destino final. A reconstrução não pode ocorrer noutro equipamento diferente do identificado, porque apenas o destinatário final possui todas as informações necessárias para reconstruir o pacote original. Isso acontece, pois apenas o destinatário possui o conhecimento completo do tamanho e da estrutura do datagrama original, além de poder identificar corretamente a sequência dos fragmentos.

De modo que, equipamentos intermédios não têm a capacidade de reconstruir os fragmentos num datagrama IP original, porque não possuem todas as informações necessárias e não são o destino final do pacote, de modo que, apenas reecaminham os fragmentos com base nas informações do cabeçalho IP, sem alterar o conteúdo.

3. 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.

Entre os diversos fragmentos, variam o resultado das *flags*, o tamanho em *bytes* dos fragmentos (entre o último e os restantes), bem como o *offset*, mas o valor TTL mantiém-se.

A informação utilizada na reconstrução do datagrama original é o IP e os diversos *offsets* presentes nos vários fragmentos. O IP permite identificar que os fragmentos pertencem ao mesmo datagrama inicial a ser reconstruído, e o *offset* orienta-os na ordem de reconstrução do mesmo.

3. h)

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

O primeiro fragmento de cada pacote é o único a ser identificado como um pacote ICMP, pois, na transmissão de pacotes fragmentados, somente primeiro cabeçalho é que leva o cabeçalho ICMP original que originou a fragmentação. De modo que, como os fragmentos seguintes são apenas partes do pacote original dividido, não contêm o cabeçalho ICMP original.

Concluindo, o cabeçalho ICMP original apenas se encontra no primeiro fragmento, enquanto os fragmentos seguintes não contêm essa informação, porque a função deles é apenas transportar parte dos dados do pacote original, sem carregar informações adicionais do cabeçalho ICMP.

3. 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?

A partir de 1480 *bytes*, o datagrama deve ser fragmentado. Caso este valor diminuisse, seria necessário enviar mais pacotes para se conseguir transmitir o datagrama inteiro.

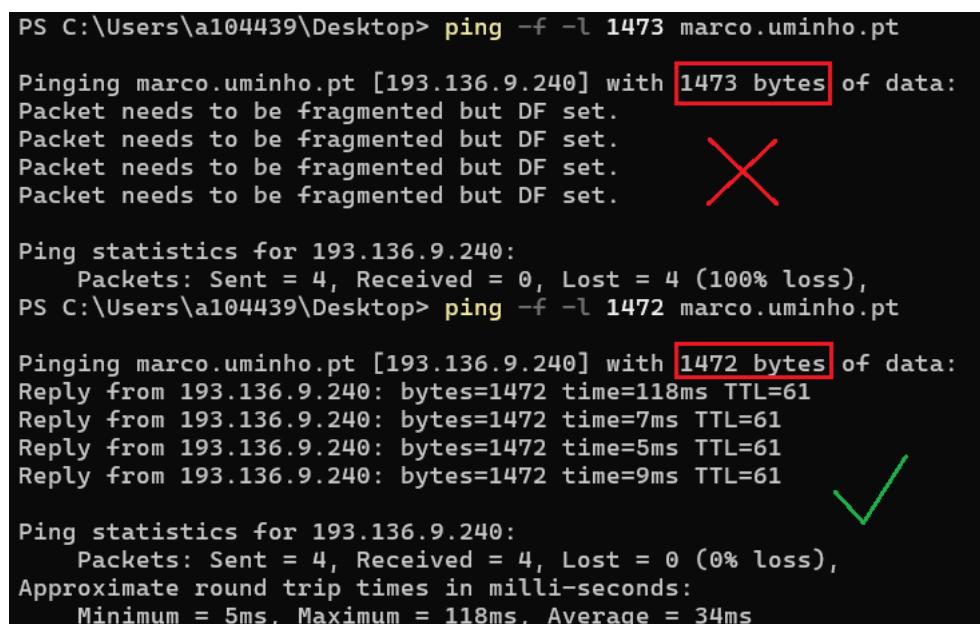
3. j)

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.

O valor máximo de `SIZE` para o pacote não ser fragmentado deve ser **1472 bytes**.

$$\begin{aligned} 1500 \text{ bytes (Ethernet MTU*)} - 20 \text{ bytes (Cabeçalho IP)} - 8 \text{ byte (Cabeçalho ICMP)} \\ = 1472 \text{ bytes} \end{aligned}$$

*Ethernet MTU → Unidade máxima de transmissão em *bytes* para *Ethernet v2*



```
PS C:\Users\al04439\Desktop> 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. X

Ping statistics for 193.136.9.240:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PS C:\Users\al04439\Desktop> 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=118ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=7ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=5ms TTL=61
Reply from 193.136.9.240: bytes=1472 time=9ms 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 = 5ms, Maximum = 118ms, Average = 34ms
```

Figura 19: Prova prática do valor obtido

2^a Parte

1.

Com os avanços da Inteligência Artificial, D. Afonso Henriques termina todas as suas tarefas mais cedo e vê-se com algum tempo livre. Decide então fazer remodelações no reino:

1. a)

De modo a garantir uma posição estrategicamente mais vantajosa e ter casa de férias para relaxar entre batalhas, ordena a construção de um segundo Castelo, em Braga. Não tendo qualquer queixa do serviço prestado, recorre aos serviços do ISP ReiDaNet, que já utiliza no condado, para ter acesso à rede no segundo Castelo. O ISP atribuiu-lhe o endereço de rede IP 172.XX.33.128/26 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 12 ou mais *hosts*. Assuma que todos os endereços de sub-redes são utilizáveis.

O ISP atribuiu o endereço 172.31.33.128/26 ao segundo Castelo. Exigindo que haja no mínimo 3 redes e que cada uma tenha no mínimo 12 *hosts*. De modo que:

#subredes ≥ 3 ; #hosts ≥ 12 ;

IP Inicial: 172.31.33.128/26

-> (passando para binário)

10101100.00011111.00100001.10000000

#subredes $\geq 3 \Rightarrow 2^n \geq 3 \Leftrightarrow n \geq 1.584\dots \Rightarrow n = 2$ bits para definir subredes

Sobram assim 4 *bits* para definir os *hosts*, confirmando-se abaixo:

$$2^n - 2 = 2^4 - 2 = 16 - 2 = 14 \geq 12$$

Assim sendo, podemos definir as 3 redes, o tamanho da máscara aumenta 2 *bits* que ficam no momento reservados para a rede:

Rede 0: IP: 10101100.00011111.00100001.10000000 -> 172.31.33.128/28

Rede 1: IP: 10101100.00011111.00100001.10010000 -> 172.31.33.144/28

Rede 2: TP: 10101100.00011111.00100001.10100000 -> 172.31.33.160/28

1. b)

Ligue um novo *host* Castelo2 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 dispositivos do Condado Portucalense.

Associámos o 1º endereço da 2ª sub-rede ao ReiDaNet, sendo este 172.31.33.145/28 e o endereço associado ao *host* Castelo2 foi 172.31.33.147/28 também pertencente à segunda sub-rede calculada anteriormente.

Como podemos ver, existe conectividade com o Condado Portucalense, tendo sido realizado um `ping` ao AfonsoHenriques.

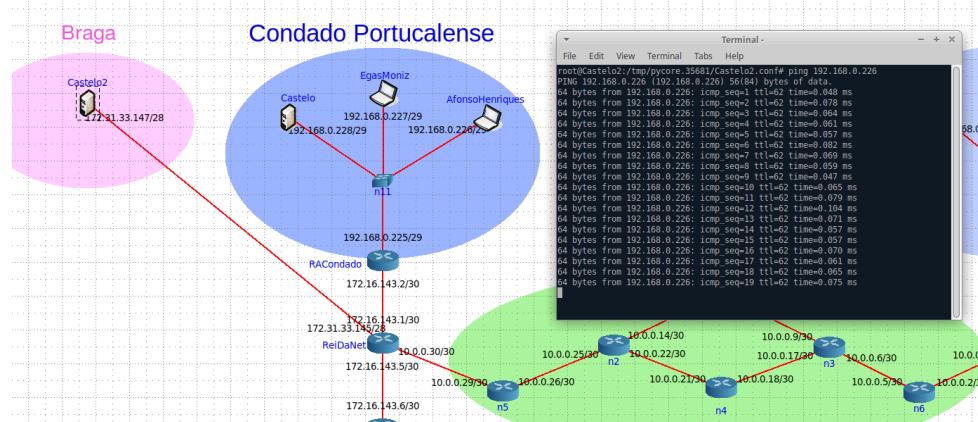


Figura 20: Verificação de conectividade através de um `ping`

1. c)

Não estando satisfeito com a decoração deste novo Castelo, opta por eliminar a sua rota *default*. Adicione as rotas necessárias para que o Castelo2 continue a ter acesso ao Condado Portucalense e à rede Institucional. Mostre que a conectividade é restabelecida, assim como a tabela de encaminhamento resultante. Explicite ainda a utilidade de uma rota *default*.

Para o Condado Portucalense, temos o endereço IP 192.168.0.225/29, passando este valor para binário ficámos com o endereço IP: 11000000.10101000.00000000.11100001

De forma a abranger todos os endereços presentes no Condado Portucalense, definimos o destino como 11000000.10101000.00000000.11100000/29 -> 192.168.0.224/29. Adicionando assim o comando `ip route add 192.168.0.224/29 via 172.31.33.145`.

Para a rede Institucional, optando por ver algo mais abrangente para funcionar para todos os endereços. De modo que, todos os endereços iniciam por 192.168.0. optámos por colocar cada último valor em binário:

```

240 (dec) = 11110000 (bin)
232 (dec) = 11101000 (bin)
248 (dec) = 11111000 (bin)

```

Observando os valores em binário, os primeiros 3 1's são comuns, ou seja, esse valor será o usado: 11100000 (bin) = 224 (dec)

A máscara neste caso diminui 2 *bits*, passando para 27, porque apenas os primeiro 27 *bits* são iguais. O comando adicionado foi `ip route add 192.168.0.224/27 via 172.31.33.145`.

Nota: Invés deste processo e de adicionar apenas uma linha para todos os dispositivos do Institucional, poderíamos ter colocado os seguintes comandos:

```

ip route add 192.168.0.240/27 via 172.31.33.145
ip route add 192.168.0.232/27 via 172.31.33.145
ip route add 192.168.0.248/27 via 172.31.33.145

```

Como podemos verificar pela imagem de seguida, a tabela de encaminhamento encontra-se com os dois comandos e existe conexão com o Institucional e com o Condado Portucalense.

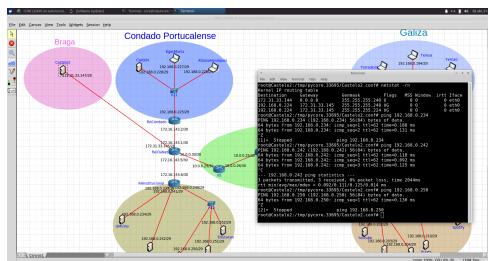


Figura 21: Prova de conectividade com o
Institucional

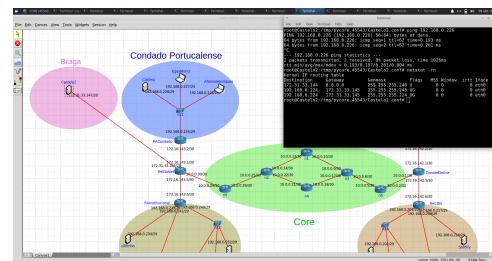


Figura 22: Prova de conectividade com o
Condado Portucalense

Haver uma rota *default* é bastante útil para evitar colocar todas as rotas possíveis manualmente, caso contrário pode haver algum tráfego a não ser encaminhado, pois não está abrangido nas rotas colocadas manualmente.

2.

D.Afonso Henriques quer enviar fotos do novo Castelo à sua mãe, D.Teresa, mas está a ter alguns problemas de comunicação. 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. a)

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

Foram executados os seguintes comandos `ping` para averiguar a conectividade de AfonsoHenriques com o servidor Financas e com os servidores CDN:

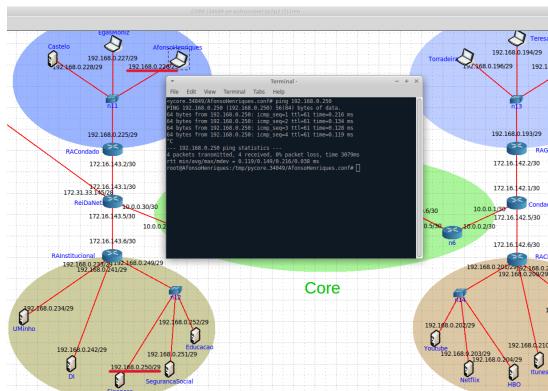


Figura 23: Conectividade com o servidor
Finanças

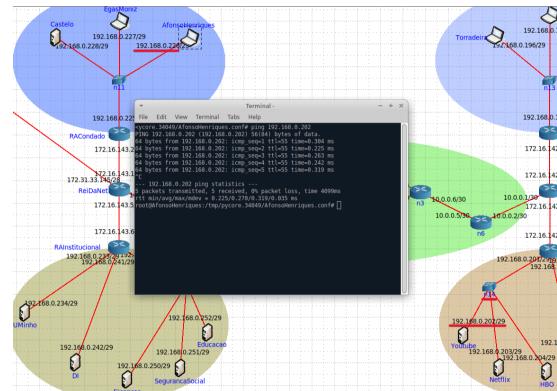


Figura 24: Conectividade com o servidor CDN (*Switch n14 - Youtube*

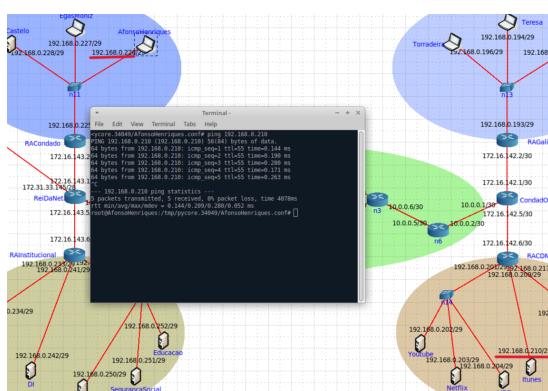


Figura 25: Conectividade com o servidor
CDN (Itunes)

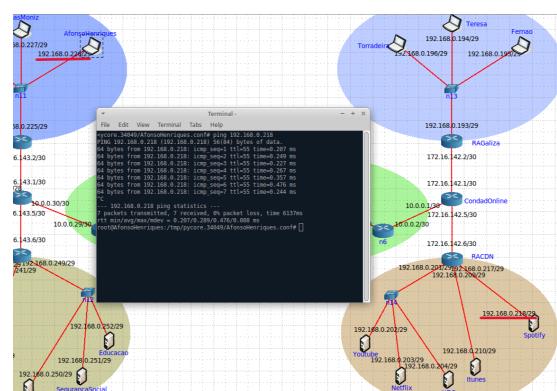


Figura 26: Conectividade com o servidor
CDN (Spotify)

2. 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*.

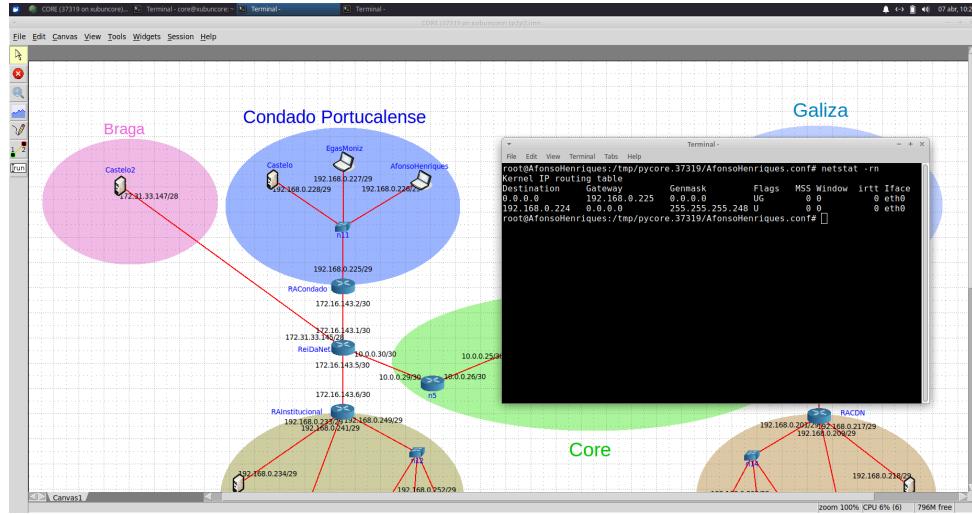


Figura 27: Tabela de Encaminhamento do AfonsoHenriques

Na tabela de encaminhamento do AfonsoHenriques, a primeira entrada `0.0.0.0` indica que o tráfego destinado a um endereço IP não especificado deve ser encaminhado para o *gateway* `192.168.0.225`, sendo este o *router* do RACondado através da interface `eth0`. Esta entrada é o “percurso” padrão para alcançar redes externas. A segunda entrada `192.168.0.224` define uma rota específica para a sub-rede em que se encontra. Isto indica que o tráfego destinado à sub-rede `192.168.0.224/29`, que inclui o próprio dispositivo AfonsoHenriques, pode ser alcançado sem nenhum *gateway* através da interface `eth0`.

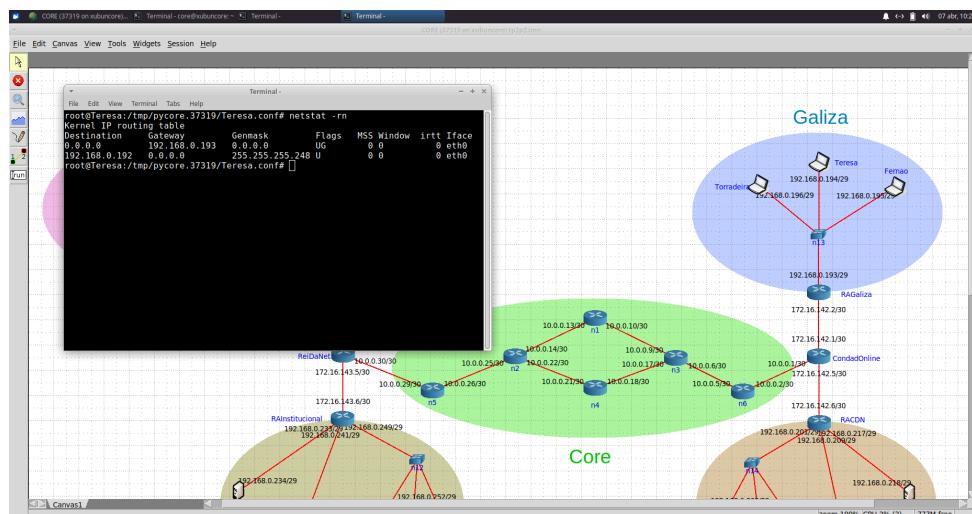


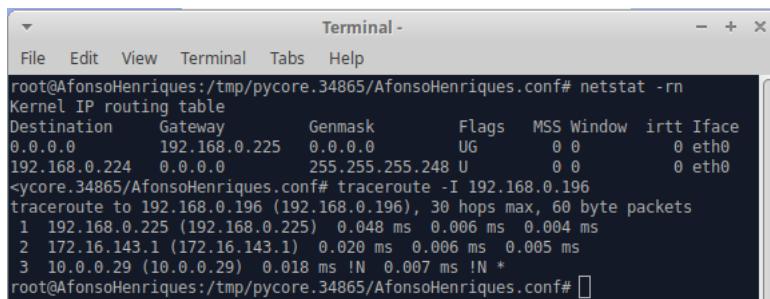
Figura 28: Tabela de Encaminhamento da Teresa

Na tabela de encaminhamento da Teresa, a primeira entrada **0.0.0.0** indica que o tráfego destinado a um endereço IP não especificado deve ser encaminhado para o *gateway* **192.168.0.193**, sendo este o *router* do RACondado através da interface **eth0**. Esta entrada é o “percurso” padrão para alcançar redes externas. A segunda entrada **192.168.0.192** define uma rota específica para a sub-rede em que se encontra. Isto indica que o tráfego destinado à sub-rede **192.168.0.192/29**, que inclui o próprio dispositivo Teresa, pode ser alcançado sem nenhum *gateway* através da interface **eth0**.

Como ambos os dispositivos têm uma entrada “padrão” (**0.0.0.0**) direcionando para os seus respetivos *gateways* (**192.168.0.225** - AfonsoHenriques e **192.168.0.193** - Teresa), indica que o tráfego destinado a alguma rede fora das suas sub-redes locais deve ser encaminhado para esses *gateways*. De modo que, não parece haver qualquer problema nas suas entradas.

2. c)

Analise 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.



```
Terminal -
File Edit View Terminal Tabs Help
root@AfonsoHenriques:/tmp/pycore.34865/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
<core.34865/AfonsoHenriques.conf# traceroute -I 192.168.0.196
traceroute to 192.168.0.196 (192.168.0.196), 30 hops max, 60 byte packets
 1  192.168.0.225 (192.168.0.225)  0.048 ms  0.006 ms  0.004 ms
 2  172.16.143.1 (172.16.143.1)  0.020 ms  0.006 ms  0.005 ms
 3  10.0.0.29 (10.0.0.29)  0.018 ms !N  0.007 ms !N *
root@AfonsoHenriques:/tmp/pycore.34865/AfonsoHenriques.conf#
```

Figura 29: Análise inicial da Tabela de Endereçamento de Afonso Henriques e do estado do caminho nó a nó até Teresa

Através de uma série de execuções **ping 192.168.0.194** (para D. Teresa) enviados por cada *router* da rede *core*, verifica-se que os únicos que falham no envio de pacotes são o n1, n2 e n5. Com essa informação analisámos a tabela de endereçamento do *router* n2.

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.0.8	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

Figura 30: Tabela de endereçamento do *router* n2

Realçado na imagem, verifica-se que todos os pacotes a serem enviados para o endereço da D. Teresa, são roteados pelo *router* n5. Para corrigir este problema, deve-se remover então esta entrada e adicionar uma que reencaminhe estes pacotes para o *router* n4.

```
root@n2:/tmp/pycore.45463/n2.conf# ip route del 192.168.0.194/31 via 10.0.0.25
<2.conf# ip route add 192.168.0.194/31 via 10.0.0.21
root@n2:/tmp/pycore.45463/n2.conf#
```

Figura 31: Correção da entrada errada

```
root@AfonsoHenriques:/tmp/pycore.33619/AfonsoHenriques.conf# ping 172.16.142.1
PING 172.16.142.1 (172.16.142.1) 56(84) bytes of data.
64 bytes from 172.16.142.1: icmp_seq=1 ttl=57 time=0.180 ms
64 bytes from 172.16.142.1: icmp_seq=2 ttl=57 time=0.150 ms
64 bytes from 172.16.142.1: icmp_seq=3 ttl=57 time=0.104 ms
^C
--- 172.16.142.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.104/0.144/0.180/0.031 ms
root@AfonsoHenriques:/tmp/pycore.33619/AfonsoHenriques.conf#
```

Figura 32: ping Afonso Henriques ao ISP CondadOnline pós alterações

Verifica-se então que após as alterações, a partir de D. Afonso Henriques é possível establecer a comunicação com o ISP CondadOnline.

2. d)

Uma vez que o *core* da rede esteja a encaminhar corretamente os pacotes enviados por Afonso Henriques, 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.

1º Inicialmente verificámos que não havia conectividade entre AfonsoHenriques e Teresa:

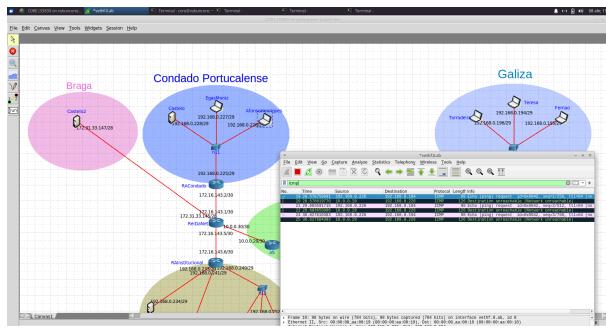


Figura 33: Verificação da falta de conectividade

2º Estabelecer conectividade de AfonsoHenriques para Teresa:

Apesar de haver tráfego a chegar ao ISP CondadOnline, não estava a haver conectividade na mesma entre os dois, de modo que analisando as tabelas de encaminhamento dos *routers* n5 e n2 verificámos que faltavam mais alguns passos.

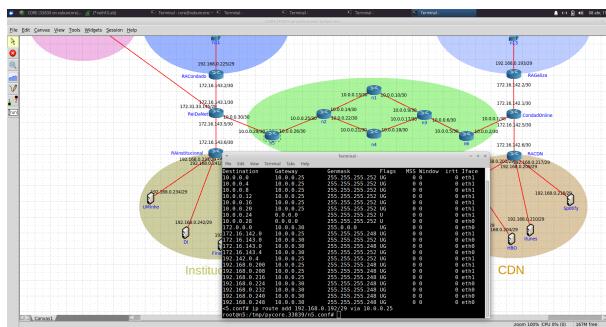


Figura 34: Conexão no n5

No *router* n5 era necessário adicionar `ip route add 192.168.0.192/29 via 10.0.0.25` para haver conectividade para RAGaliza indo para o *router* n2.

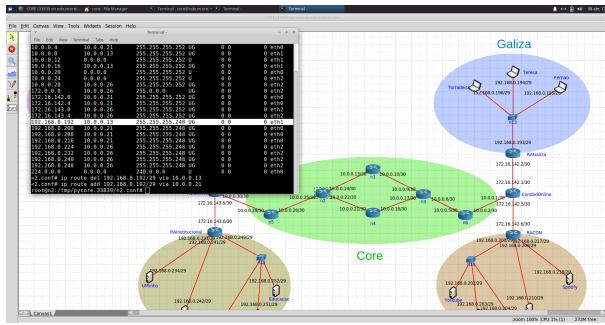


Figura 35: Conexão no n2

No *router* n2, verificámos que o destino do RAGaliza estava a ir pelo *router* n1, quando já existe um caminho pelo *router* n4, de modo que retirámos esse caminho pelo *router* n1 `ip route del 192.168.0.192/29 via 10.0.0.13` e adicionámos pelo *router* n4 `ip route add 192.168.0.192/29 via 10.0.0.21`.

3º Estabelecer conectividade de Teresa para AfonsoHenriques:

Apesar de já haver conectividade de AfonsoHenriques para Teresa, ainda não há de Teresa para AfonsoHenriques.

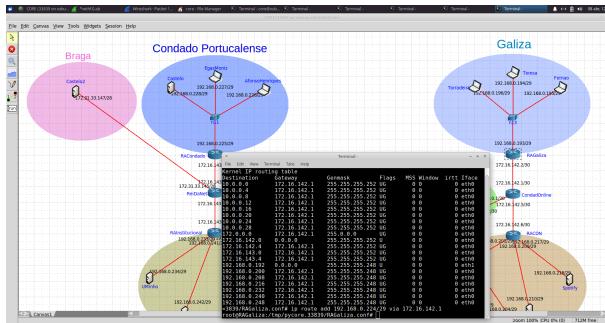


Figura 36: Conexão no RAGaliza

Observando o *router* RAGaliza, verificámos que falta adicionar a rota para o RACondado, de modo que adicionámos `ip route add 192.168.0.224/29 via 172.16.142.1`.

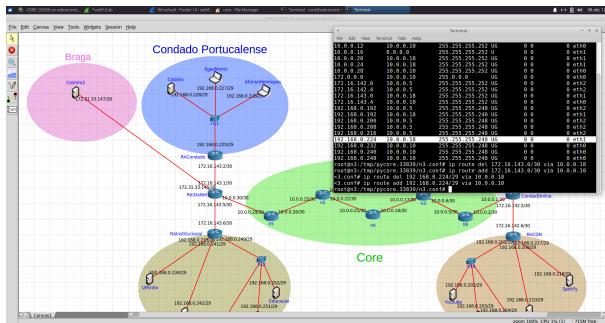


Figura 37: Conexão no n3

Já tendo RAGaliza com as rotas corretas na sua tabela de encaminhamento, continua a não haver conexão pois o *router* n3 invés de fazer a rota pelo n1 (*router* que já tem o

caminho para RACondado correto) está a enviar pelo *router* n4, não estando este com a tabela de encaminhamento correta. Assim eliminámos a rota pelo *router* n4 ip route del 172.16.143.0/30 via 10.0.0.18 e ip route del 192.168.0.224/29 via 10.0.0.18; adicionámos pelo n1 ip route add 172.16.143.0/30 via 10.0.0.10 e ip route add 192.168.0.224/29 via 10.0.0.10.

4º verificação da conectividade:

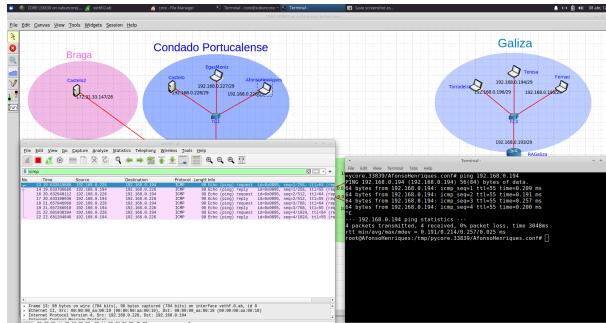


Figura 38: Verificação final da conectividade

Após esta análise e adição e eliminação de rotas, podemos verificar que já existe conectividade entre AfonsoHenriques e Teresa, como podemos ver pelo `ping` realizado, onde não há perda de informação.

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 `traceroute`). Mostre graficamente a rota seguida nos dois sentidos por esses pacotes ICMP.

As rotas dos pacotes *ICMP echo reply* não são iguais em sentido inverso que as rotas dos pacotes *ICMP echo request* enviados entre AfonsoHenriques e Teresa, pois estes pacotes realizam percursos diferentes.

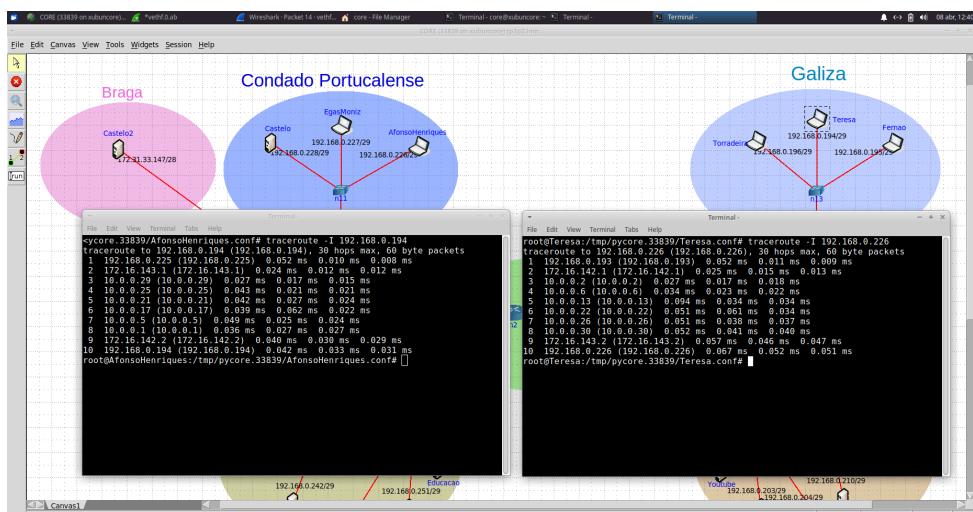


Figura 39: traceroute de ambos os dispositivos

De seguida, observa-se pelas imagens que a rota realizada pelos pacotes a partir de AfonsoHenriques é diferente da rota realizada pelos pacotes a partir de Teresa.

Rotas:

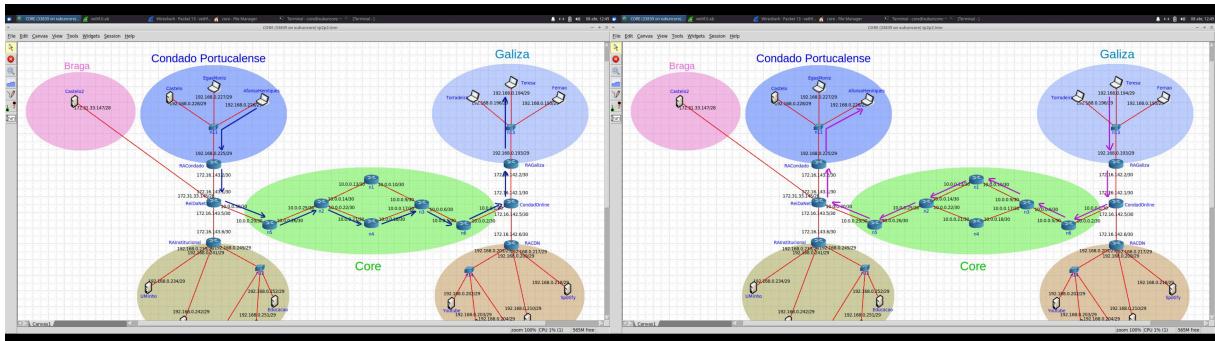


Figura 40: Rota de AfonsoHenriques para Figura 41: Rota de Teresa para AfonsoHenriques

2. 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	10.0.0.3	255.255.255.240	00	0 0	0 brnz
192.168.0.192	10.0.0.18	255.255.255.240	UG	0 0	0 eth1
192.168.0.192	10.0.0.5	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.

As subredes Galiza e CDN têm entradas na tabela de encaminhamento com correspondências (*matches*) mais compridas. Para evitar problemas de redes, o *router* optará pelas mesmas, já que, caso não existissem, o tráfego não iria ser encaminhado corretamente.

2. f)

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

Visto que, todos os endereços na topologia se encontram nos intervalos definidos especificamente para redes privadas na arquitetura de endereçamento da *Internet* (formulada pela IETF e IANA), estes são todos privados.

2. g)

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

Os *switches* não têm nenhum endereço IP atribuído. Estes agem como pontes dentro de uma rede, ligando diferentes dispositivos para facilitar a comunicação. Estes operam com base em endereços físicos dos dispositivos, invés de usarem endereços IP. De modo que, não é necessário atribuir nenhum endereço IP a um *switch*, visto que ele não comunica ao nível de rede, mas sim na camada de ligação de dados.

3.

Ao ver as fotos no CondadoGram, D. Teresa não ficou convencida com as novas alterações e ordena que Afonso Henriques vá arrumar o castelo. Inconformado, este decide planear um novo ataque, mas constata que o seu exército não só perde bastante tempo a decidir que direção tomar a cada salto como, por vezes, inclusivamente se perde.

3. a)

De modo a facilitar a travessia, elimine as rotas referentes a Galiza e CDN no dispositivo nº 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.

Todos os endereços IP do CondadoPortucalense e Institucional iniciam por 192.168.0., de modo que só é necessário verificar os últimos 8 *bits* de cada endereço:

```
192 (dec) = 11000000 (bin)
200 (dec) = 11001000 (bin)
208 (dec) = 11010000 (bin)
216 (dec) = 11011000 (bin)
```

Como podemos verificar, apenas 3 *bits* são comuns nos últimos 8 *bits*, de modo que o endereço IP usado para Supernetting será o número 11000000, correspondente a 192 em decimal, e a máscara passa a ser 27, pois apenas 27 *bits* permanecem iguais. Assim, adicionando a linha de comando `ip route add 192.168.0.192/27 via 10.0.0.1`:

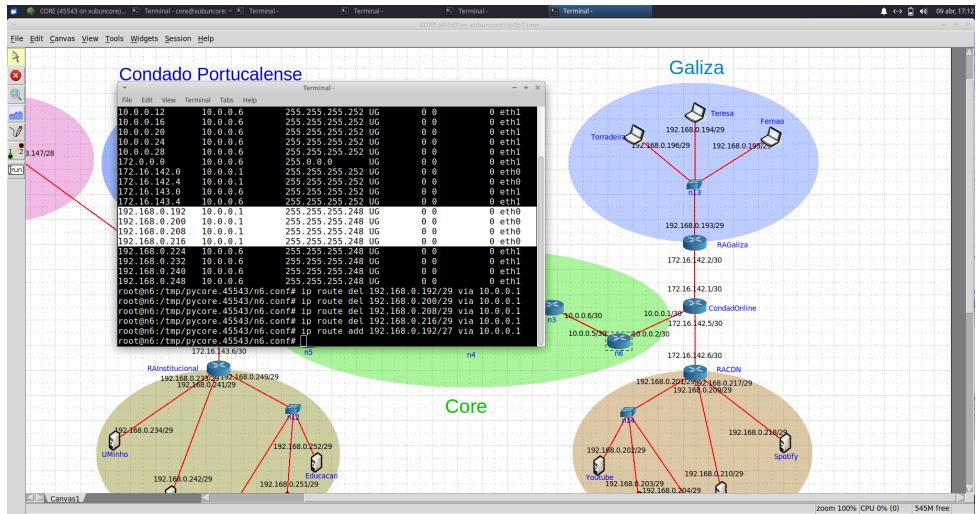


Figura 43: Eliminação das rotas referentes ao CondadoPortucalense e Institucional e adição de *Supernetting*

Nesta imagem verifica-se a conexão.

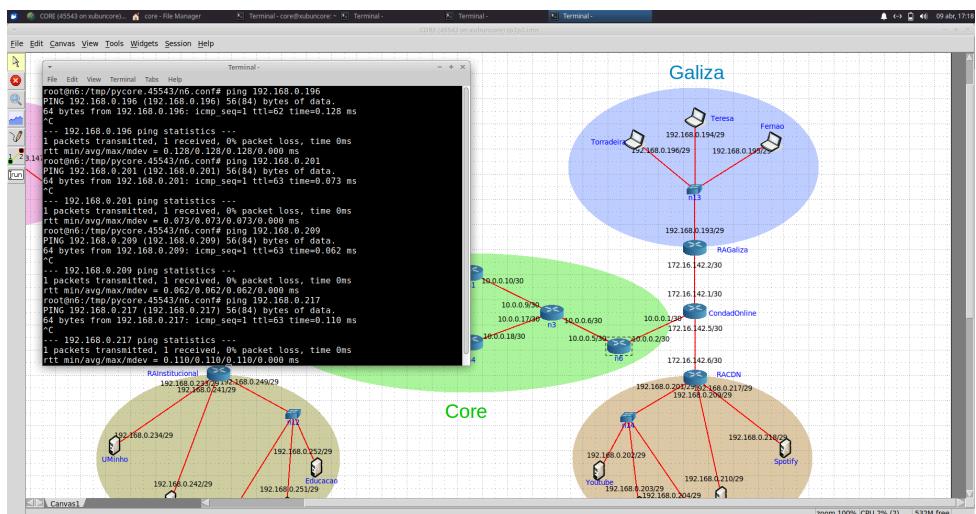


Figura 44: Verificação da conectividade

3. b)

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

Todos os endereços IP do CondadoPortucalense e Institucional iniciam por **192.168.0.**, de modo que só é necessário verificar os últimos 8 *bits* de cada endereço:

```
224 (dec) = 11100000 (bin)
232 (dec) = 11101000 (bin)
240 (dec) = 11110000 (bin)
248 (dec) = 11111000 (bin)
```

Como podemos verificar, apenas 3 bits são comuns nos últimos 8 bits, de modo que o endereço IP usado para *Supernetting* será o número 11100000, correspondente a 224 em decimal, e a máscara passa a ser 27, pois apenas 27 bits permanecem iguais. Assim, adicionando a linha de comando `ip route add 192.168.0.224/27 via 10.0.0.6`:

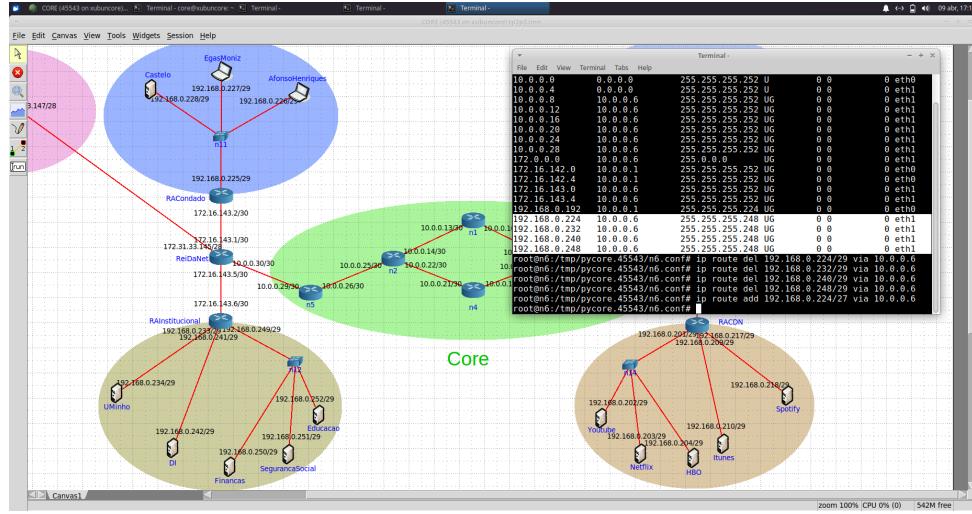


Figura 45: Eliminação das rotas referentes ao CondadoPortucalense e Institucional e adição de *Supernetting*

Nesta imagem verifica-se a conexão.

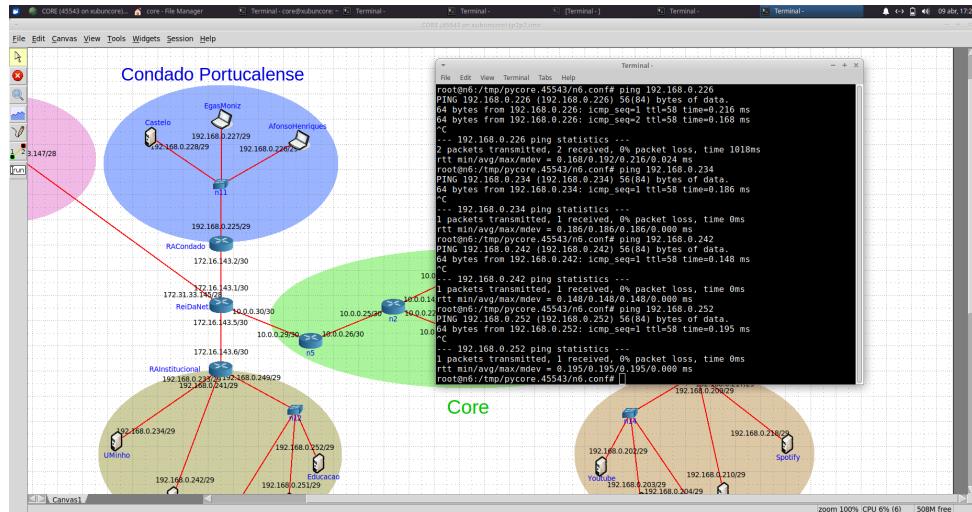


Figura 46: Verificação da conectividade

3. c)

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

Os aspetos positivos do uso do *Supernetting* são:

- reduz o tráfego da rede pela *internet*;
- aumenta a velocidade da pesquisa pela tabela de encaminhamento, aumentando a performance da rede;
- com o uso de *supernetting*, o número de entradas de informações de rotas diminui, pois é resumido numa única entrada, de modo que o tamanho da tabela de memória do *router* diminui, economizando espaço de memória.

Os aspetos negativos do uso do *Supernetting* são:

- toda a rede encontra-se na mesma “classe”;
- apesar de abranger muitas entradas numa, falha em cobrir algumas áreas diferentes;
- risco de esgotamento do endereço IP, caso não seja usado corretamente.